



HPCC Client Tools

Boca Raton Documentation Team

HPCC Client Tools

Boca Raton Documentation Team

Copyright © 2015 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.

2015 Version 5.4.2-1

Overview	4
Documentation Conventions	5
ECL Plus	8
Command Line Interface	8
ECL Command Line Interface	14
The ECL Command Syntax	14
ECL Compiler	51
<i>Using the ECL Compiler as a Stand Alone option</i>	52
Compiled Options:	54
Examples	55
Command Line DFU	57
Command Line Interface	57
ESDL Command Line Interface	68
The ESDL Command Syntax	68

Overview

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

ECLPlus	Command line ECL execution tool to facilitate automation of ECL Code execution.
ECL	Command line ECL tool.
ECL Compiler	Command line ECL Compiler
DFUPlus	Command line Distributed File Utility management tool, facilitate automation of data file spray, despray, and other common file handling tasks.
ESDL	Command line ESDL management tool.

Documentation Conventions

ECL Language

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

Example Code

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

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

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

Actions

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

- Keyboard and mouse actions are shown in 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 ECLPlus, DFUPlus, and the ECL Command line tools.

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

Download the appropriate Client Tools for your Operating System. (available for 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-4.X.X-XWindows-i386.exe on your machine. Follow the prompts to complete the installation.

RPM-Based Systems (CentOS/RedHat):

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

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

Debian-Based Systems (Ubuntu):

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

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

Mac OSX:

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:

<http://hpccsystems.com/download/free-community-edition/client-tools>

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 4.0.x, 4.2.x, 4.4.x, etc.

HPCC Client Tools

Overview

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 eclplus:

```
eclplus action=view wuid=W12345678
```

To run eclplus for an older or another version of client tools, for instance 4.0.x:

```
/opt/HPCCSystems/4.0.x/clienttools/bin/eclplus action=view wuid=W12345678
```

Windows

Client tools for Windows installs in a directory such as: C:\Program Files (x86)\HPCCSystems\4.2.0\clienttools where the number (4.2.0 for example) corresponds to the version of the client tools.

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 Plus

Command Line Interface

ecplus.exe

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

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

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

ecplus.ini

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

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

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



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

Running queries in batch mode

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

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

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

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

Then if you run:

```
ecplus @dataset.txt
```

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

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

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

then run:

```
ecplus @datasetquery.txt
```

You'll get:

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

Workunit manipulation

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

List all work units

To list all work units:

```
eclplus action=list
```

The output looks like:

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

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

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

View the result of a certain workunit

You can look at specific workunit results, like this:

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

The output will look like:

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

Dump a workunit

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

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

See the Workunit Dump section below for the result.

Return graph data for a workunit:

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

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

Graph name must be supplied in the graph= parameter.

Aborting a workunit

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

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

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

Timeout

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

For example:

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

eclplus will return immediately.

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

eclplus will return in 2 seconds.

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

Output format

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

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

Also, you may specify the following output formats:

CSV

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

csvh

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

raw

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

runectl

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

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

bin(ary)

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

Workunit Dump

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

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

```
<W20110615-160604 agentPID="4162" agentSession="4296042782" cloneable="1"
clusterName="thor" codeVersion="138" isClone="1" scope="hpccdemo"
state="completed" submitID="hpccdemo"
token="X1lUMJ6oacON/1anTHTQW1JVHr1bbY8EWTSJh1DOrtYxmD13Z5ly4Qd26sEYVtxhw">
  <Action>run</Action>
  <Debug>
    <applyinstantecltrtransformations>1</applyinstantecltrtransformations>
    <applyinstantecltrtransformationslimit>100</applyinstantecltrtransformationslimit>
    <created_by>ws_workunits</created_by>
    <created_for>hpccdemo</created_for>
    <eclagentlog>//192.168.237.132/var/log/HPCCSystems/myeclagent/eclagent.06_15_11.log
    </eclagentlog>
    <targetclustertype>hthor</targetclustertype>
  </Debug>
  <Query fetchEntire="1">
    <Associated>
      <File crc="701142319" filename="libW20110615-160604.so" type="dll"/>
    </Associated>
    <Text>
      <Archive build="community_3.0.0" eclVersion="3.0.0"> <Query
        originalFilename="C:\DOCUME~1\Hpccdemo\LOCALS~1\Temp\TFR2CE.tmp">
          OUTPUT(&apos;Hello World&apos;); </Query> </Archive>
      </Text>
    </Query>
  </resultLimit>100</resultLimit>
  <Results>
    <Result fetchEntire="1" name="Result 1" sequence="0" status="calculated">
      <rowCount>1</rowCount>
      <SchemaRaw xsi:type="SOAP-ENC:base64"> UmVzdWx0XzEABPH///8BYXNjaWkAAWFzY2lpAAAYAAAAA==
      </SchemaRaw>
      <totalRowCount>1</totalRowCount>
      <Value xsi:type="SOAP-ENC:base64"> CwAAAEhlbGxvIFdvcmxk </Value>
    </Result>
  </Results>
  <TimeStamps>
    <TimeStamp application="workunit">
      <Created ts="1308153964"> 2011-06-15T16:06:04Z </Created>
    </TimeStamp>
    <TimeStamp application="EclAgent" instance="localhost.localdom">
      <Started ts="1308153971"> 2011-06-15T16:06:11Z </Started>
    </TimeStamp>
  </TimeStamps>
</W>
```

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

ECL Command Line Interface

The ECL Command Syntax

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

--version displays version info.

Arguments

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

ecl.ini

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

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

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



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

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

Evaluation of options follows this order of precedence:

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

Environment Variables

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

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



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

ecl deploy

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

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

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

Examples:

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

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

ecl deploy	Creates a workunit on the HPCC system from the given ECL text, file, archive, shared object, or dll. The workunit is created in the <i>compiled</i> state.
------------	--

Arguments

ecl_file	The ECL text file to deploy
archive	The ECL archive to deploy
so dll	The workunit dynamic linked library or shared object to deploy

Options

-t, --target	The target cluster to associate workunit with
-n, --name	The workunit query name
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--main	The definition to use from legacy ECL repository
--ecl-only	Send ecl text to hpcc without generating archive

ecclcc Options

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

ecl publish

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

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

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

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

Examples:

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

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

ecl publish Publishes a query into a queryset. The query is created by adding a workunit to a queryset and assigning it a query name.

Arguments

wuid	The workunit id to publish
ecl_file	The ECL text file to deploy
archive	The ECL archive to deploy
so dll	The workunit dynamic linked library or shared object to deploy

Options

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

HPCC Client Tools
ECL Command Line Interface

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

ecl unpublish

ecl unpublish <queryset> <query_id>

Example:

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

ecl unpublish executes the supplied ecl unpublish command

Arguments

queryset The name of queryset containing query to unpublish

query_id The query to remove from query set. Wildcards allowed, but must be in quotes (e.g., "MyQuery*").

Options

-v, --verbose Output additional tracing information

-s, --server The IP Address or hostname of ESP server running ECL Watch services

--port The ECL Watch services port (Default is 8010)

-ssl Use SSL to secure the connection to the server.

-u, --username The username (if necessary)

-pw, --password The password (if necessary)

ecl run

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

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

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

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

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

Examples:

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

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

ecl run executes the supplied ecl run command

Arguments

wuid	The workunit id to publish
ecl_file	The ECL text file to deploy
archive	The ECL archive to deploy
so dll	The workunit dynamic linked library or shared object to deploy

Options

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

ecldata Options

-Ipath	Add path to locations to search for ecl imports
--------	---

HPCC Client Tools
ECL Command Line Interface

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

ecl activate

ecl activate <queryset> <query_id>

Example:

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

ecl activate	Activates a published query. This assigns a query to the active alias with the same name as the query.
--------------	--

Arguments

queryset	The name of queryset containing query to activate
query_id	The query to activate

Options

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

ecl deactivate

ecl deactivate <queryset> <active_alias>

Example:

```
ecl deactivate Roxie FindpersonService
```

ecl deactivate Deactivates a published query by removing an active query alias from the given query-set.

Arguments

queryset The name of queryset containing alias to deactivate
active_alias The active alias to be removed from the queryset

Options

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

ecl queries list

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

Examples:

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

ecl queries list	Displays a list of the queries in one or more querysets. If a cluster is provided the querysets associated with that cluster will be shown. If no queryset or cluster is specified all querysets are shown.
------------------	---

Actions

list	List queries in queryset(s)
------	-----------------------------

Options

queryset	The name of queryset from which to list queries
-t, --target	The target cluster associated with the queries to list
-A, --activate	Activates query when published
--show=<flags>	Show only queries with matching flags

Flags

A	Active
S	Suspended
U	No Flags

Options

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

ecl queries copy

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

Examples:

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

ecl queries copy Copies a query from one queryset to another. A query can be copied from one HPCC environment to another by using a path which begins with '/' followed by the IP or hostname and Port of the source ECL Watch and then followed by the source queryset and query.

Actions

copy Copy a query from one queryset to another

Options

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

ecl queries copy-set

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

Examples:

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

ecl queries copy-set Copies a set of queries from one target to another.

Actions

copy-set Copy a set of queries from one target to another.

Options

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

ecl queries config

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

Examples:

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

ecl queries config Updates query configuration values

Actions

config Set or update query configuration values

Options

target The name of the target queryset

queryid The name of the query

--no-reload Specifies to not request a reload of the Roxie cluster

--wait=<sec> Maximum time to wait for cluster finish updating (in ms)

--timeLimit=<sec> Value to set for query timeLimit configuration

--warnTimeLimit=<sec> Value to set for query warnTimeLimit configuration

--memoryLimit=<mem> Value to set for query memoryLimit configuration. Format <mem> as 500000B, 550K, 100M, 10G, or 1T, etc.

-v, --verbose Output additional tracing information

-s, --server The IP Address or hostname of ESP server running ECL Watch services

--port The ECL Watch services port (Default is 8010)

-ssl Use SSL to secure the connection to the server.

-u, --username The username (if necessary)

-pw, --password The password (if necessary)

ecl packagemap add

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

Examples:

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

ecl packagemap add Calls the packagemap add command

Actions

add Adds a packagemap to the target cluster

Arguments

target The target to associate the packagemap with

filename The name of the file containing packagemap information.

--daliip= IP address or hostname of the remote Dali to use for logical file lookups

Options

-O, --overwrite Whether to overwrite existing information - true if present

-A, --activate Activates packagemap

--allow-foreign Specifies to allow the use of foreign files. If a packagemap references foreign files and this is not enabled, packagemap add will fail.

--pmid=<packagemapid> id of package map - defaults to filename if not specified

-v, --verbose Output additional tracing information

-s, --server The IP Address or hostname of ESP server running ECL Watch services

--port The ECL Watch services port (Default is 8010)

-ssl Use SSL to secure the connection to the server.

-u, --username The username (if necessary)

-pw, --password The password (if necessary)

ecl packagemap delete

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

Examples:

```
ecl packagemap delete roxie mypackagemap
```

ecl packagemap delete

Calls the packagemap delete command

Actions

delete

Deletes a packagemap

Options

-v, --verbose

Output additional tracing information

-s, --server

The IP Address or hostname of ESP server running ECL Watch services

--port

The ECL Watch services port (Default is 8010)

-ssl

Use SSL to secure the connection to the server.

-u, --username

The username (if necessary)

-pw, --password

The password (if necessary)

ecl packagemap activate

ecl packagemap activate <target> <packagemap>

Example:

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

ecl packagemap activate

The activate command will deactivate the currently active packagemap and make the specified packagemap active.

Arguments

target

The target containing the packagemap to activate

packagemap

name of packagemap to update

Options

-v, --verbose

Output additional tracing information

-s, --server

The IP Address or hostname of ESP server running ECL Watch services

--port

The ECL Watch services port (Default is 8010)

-ssl

Use SSL to secure the connection to the server.

-u, --username

The username (if necessary)

-pw, --password

The password (if necessary)

ecl packagemap deactivate

ecl packagemap deactivate <target> <packagemap>

Example:

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

ecl packagemap deactivate

The deactivate command will deactivate the currently active packagemap.

Arguments

target

The target containing the packagemap to deactivate

packagemap

Name of packagemap to deactivate

Options

-v, --verbose

Output additional tracing information

-s, --server

The IP Address or hostname of ESP server running ECL Watch services

--port

The ECL Watch services port (Default is 8010)

-ssl

Use SSL to secure the connection to the server.

-u, --username

The username (if necessary)

-pw, --password

The password (if necessary)

ecl packagemap list

ecl packagemap list <target>

Examples:

```
ecl packagemap list roxie
```

ecl packagemap list Calls the packagemap list command

Actions

list Lists loaded packagemap names

Arguments

target The target containing the packagemap to list

Options

-v, --verbose Output additional tracing information

-s, --server The IP Address or hostname of ESP server running ECL Watch services

--port The ECL Watch services port (Default is 8010)

-ssl Use SSL to secure the connection to the server.

-u, --username The username (if necessary)

-pw, --password The password (if necessary)

ecl packagemap info

ecl packagemap info [options] <target>

Examples:

```
ecl packagemap info roxie
```

ecl packagemap info Calls the packagemap info command

Actions

info returns packagemap info

Arguments

target The target containing the packagemap to retrieve

Options

-v, --verbose Output additional tracing information

-s, --server The IP Address or hostname of ESP server running ECL Watch services

--port The ECL Watch services port (Default is 8010)

-ssl Use SSL to secure the connection to the server.

-u, --username The username (if necessary)

-pw, --password The password (if necessary)

ecl packagemap validate

ecl packagemap validate <target> [<filename>]

Examples:

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

The packagemap validate command verifies that :

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

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

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

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

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

ecl roxie attach

ecl roxie attach <processName>

Examples:

```
ecl roxie attach myroxie
```

ecl roxie attach Attach the roxie to Dali

Options

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

ecl roxie detach

ecl roxie detach <processName>

Examples:

```
ecl roxie detach myroxie
```

ecl roxie detach

Detach the roxie from Dali

Options

-v, --verbose

Output additional tracing information

-s, --server

The IP Address or hostname of ESP server running ECL Watch services

--port

The ECL Watch services port (Default is 8010)

-ssl

Use SSL to secure the connection to the server.

-u, --username

The username (if necessary)

-pw, --password

The password (if necessary)

ecl roxie reload

ecl roxie reload <processName>

Examples:

```
ecl roxie reload myroxie
```

ecl roxie reload Reloads the roxie info from Dali

Options

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

ecl roxie check

ecl roxie check <processName>

Examples:

```
ecl roxie check myroxie
```

ecl roxie check Checks the state of the roxie process

Options

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

ecl bundle depends

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

Examples:

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

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

ecl bundle info

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

Examples:

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

ecl bundle info Lists information about a bundle

Options

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

ecl bundle install

ecl bundle install <bundleName>

Examples:

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

ecl bundle install Installs a bundle

Options

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

ecl bundle uninstall

ecl bundle uninstall <bundleName>

Examples:

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

ecl bundle install Installs a bundle

Options

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

ecl bundle list

ecl bundle list <pattern>

Examples:

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

ecl bundle list

Lists bundles matching specified pattern

Options

<pattern>

A pattern specifying bundles to list. If omitted, all bundles are listed

--details

Report details of each installed bundle

-v, --verbose

Output additional tracing information

ecl bundle use

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

Example:

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

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

ecl roxie unused-files

ecl roxie unused-files <processName>

Examples:

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

ecl roxie unused-files	Finds files in the DFS for the given roxie process that are not currently used by queries on that roxie.
------------------------	--

Options

--check-pagemaps	Exclude files referenced in active pagemaps
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
-ssl	Use SSL to secure the connection to the server.
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl abort

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

Examples:

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

ecl abort aborts one or more Workunits from the given WUID or job name

Options

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

ecl status

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

Examples:

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

ecl status returns the status of a given workunit or job name. If more than one is found, a CSV list returns.

Options

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

ecl getwuid

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

Examples:

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

ecl getwuid returns the WUID(s) for a given job name. If more than one is found, a list returns.

Options

-n	The job name
--limit= <i>nn</i>	Integer to set result limit, default is 100
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
-ssl,--ssl	Use SSL to secure the connection to the server.
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl getname

ecl getname -wu <WUID>

Examples:

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

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

ECL Compiler

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

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



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

Using the ECL Compiler as a Stand Alone option

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

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

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

Usage: eclcc <options> ECL_file.ecl

General options:

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



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

Output control options:

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

C++ options:

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

Other options:

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

Compiled Options:

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

Usage: a.out <options>

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

Examples

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

Running a basic ECL program using the command line compiler

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

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

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

(including the quotes) into the file.

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

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

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

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

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

```
eclcc hello.ecl
```

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

on Linux systems:

```
./a.out
```

on Windows systems:

```
a.out
```

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

Compile with Options

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

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

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

on Linux systems:

```
./Hello
```

on Windows systems:

```
Hello
```

This will result in the output of the following.

```
Hello world
```

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

on Linux systems:

```
./Hello -xml
```

on Windows systems:

```
Hello -xml
```

This would result in the output of the following:

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

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

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

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

This would result in the output of the following:

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

Command Line DFU

Command Line Interface

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

<i>--version</i>	displays version info
<i>operation</i>	One of the following actions: spray, despray, copy, remove, rename, list, add, addsuper, removesuper, listsuper, savexml, status, abort, resubmit, monitor
<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.

This option is only available on systems where replication has been enabled.

HPCC Client Tools
Command Line DFU

<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>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=<password></i>	Optional. Specifies to encrypt the target filename using the supplied password.
<i>decrypt=<password></i>	Optional. Specifies to decrypt the source filename using the supplied password.
<i>jobname=<jobname></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:

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

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



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

Spray Operations:

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

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

srcip	Optional. The IP address of the source machine. If omitted, the information must be supplied by the <i>srcxml</i> parameter.
srcfile	Optional. The path to the source file. This may contain wildcard characters (* and ?) to include multiple source files in the spray to a single <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.
dstname	The logical name of the destination file.
dstcluster	The name of the destination cluster.
prefix	Optional. Both of the following (separated by a comma):
filename{:length}	Prepends the filename (optionally limited to <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 specifies big endian, L specifies little endian) and the size of integer to contain it (1 to 8 bytes). If format and size are omitted, the default is L4.
<i>format</i>	Optional. One of the following values: fixed csv delimited xml recfmv recfmb . If omitted, the default is fixed.
fixed <i>format</i> options:	
<i>recordsize</i>	The fixed size of each record, in bytes.
csv/delimited <i>format</i> options:	
<i>encoding</i>	Optional. One of the following: ascii , utf8 , utf8n , utf16 , utf16le , utf16be , utf32 , utf32le , utf32be ; If omitted, the default is ascii .
<i>maxrecordsize</i>	Optional. The maximum size of each record, in bytes. If omitted, the default is 8192.
<i>separator</i>	Optional. The field delimiter. If omitted, the default is a comma (,).

<i>terminator</i>	Optional. The record delimiter. If omitted, the default is line feed or carriage return line feed (\r,\r\n).
<i>quote</i>	Optional. The string quote character. If omitted, the default is single quote (').
xml format options:	
<i>rowtag</i>	The XML tag identifying each record.
<i>encoding</i>	Optional. One of the following: utf8 utf8n utf16 utf16le utf16be utf32 utf32le utf32belf omitted, the default is utf8.
<i>maxrecordsize</i>	Optional. The maximum size of each record, in bytes. If omitted, the default is 8192.

Examples:

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

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

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

//the spray.xml file contains:
<File directory="c:\import\"
  group="thor"
  modified="2004-04-27T14:58:38"
  name="zip"
  numparts="2"
  partmask="zip._P$_of_$_$">
<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=thor dstname=RTTEMP::myzip1 recordsize=5

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

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

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

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

Despray Operations:

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

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

<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> parameter.
<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 <i>savexml 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):
filename{[:length]}	Uses the prepended filename (see the <i>prefix</i> option to the <i>spray operation</i>) to split out the data into separate files.
filesize{:[B L][1-8]}	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.

Examples:

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

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

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

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

Copy Operations:

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

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

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

<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.
<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.

Example:

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

Remove Operations:

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

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

name The logical name of the file to remove.

Example:

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

Rename Operations:

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

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

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

Example:

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

List Operations:

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

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

name The mask defining the logical file names to list.

Example:

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

Add Operations:

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

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

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

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

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

Example:

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

Addsuper Operations:

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

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

<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
```

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:

wuid The workunit identifier of the workunit.

Example:

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

Abort Operations:

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

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

wuid The workunit identifier of the workunit.

Example:

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

Resubmit Operations:

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

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

wuid The workunit identifier of the workunit.

Example:

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

Savexml Operations:

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

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

srcname The logical name of the source file.

srcname The logical name of the source file.

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

Example:

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

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

Monitor Operations:

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

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

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

ESDL Command Line Interface

The ESDL Command Syntax

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

<i>--version</i>	displays version info.
<i>help <command></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>get-binding</i>	Get ESDL binding.

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>-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 [options] filename.ecm [<outdir>]

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

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

Examples:

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

esdl xsd

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

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

This generates XSD from the ESDL definition.

Examples:

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

esdl wsd

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

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

This generates WSDL from ESDL definition.

Examples:

```
esdl wsd MathService.ecm .
```

esdl publish

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

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

Publishes an ESDL service definition to the system datastore.

Examples:

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

esdl list-definitions

esdl list-definitions [options]

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

This command lists published definitions

Example:

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

esdl delete

esdl delete <ESDLServiceDefinitionName> <ESDLServiceDefinitionVersion> [options]

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

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

Example:

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

esdl bind-service

esdl bind-service

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

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

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

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

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

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

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

Example:

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

esdl list-bindings

esdl list-bindings [options]

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

Use this command to list bindings on a server.

Example:

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

esdl unbind-service

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

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

Use this command to unpublish ESDL Service based bindings.

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

Example:

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

esdl bind-method

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

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

Use this command to publish ESDL Service based bindings.

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

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

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

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

Example:

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

esdl get-binding

esdl get-binding <ESDLBindingId> [options]

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

Use this command to get DESDL Service based bindings.

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

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

Example:

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