HPCC Systems®

HPCC Client Tools

Boca Raton Documentation Team

LexisNexis® Risk Solutions

HPCC Client Tools

Boca Raton Documentation Team

2017 Version 6.4.8-1

Overview 4
- Documentation Conventions 5

ECL Plus 8
- Command Line Interface 8

ECL Command Line Interface 14
- The ECL Command Syntax 14

ECL Compiler 56
- Using the ECL Compiler as a Stand Alone option 57
- Compiled Options: 60
- Examples 61

Command Line DFU 63
- Command Line Interface 63

ESDL Command Line Interface 75
- The ESDL Command Syntax 75

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.

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\6.2.0\clienttools\bin where the number (6.2.0 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\6.2.0\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 Plus

Command Line Interface

eclplus.exe

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

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

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

eclplus.ini

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

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

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

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

Running queries in batch mode

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

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

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

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

Then if you run:

```
eclplus @dataset.txt
```

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

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

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

then run:

```
eclplus @datasetquery.txt
```

You'll get:

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

Workunit manipulation

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

List all work units

To list all work units:

```
eclplus action=list
```

The output looks like:

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

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

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

View the result of a certain workunit

You can look at specific workunit results, like this:

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

The output will look like:

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

Dump a workunit

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

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

See the Workunit Dump section below for the result.

See the thor graph of a workunit:

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

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

Graph name must be supplied in the graph= parameter.

Aborting a workunit

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

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

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

Timeout

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

For example:

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

eclplus will return immediately.

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

eclplus will return in 2 seconds.

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

Output format

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

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

Also, you may specify the following output formats:

csv

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

csvh

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

raw

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

runecl

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

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

bin(ary)

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

Workunit Dump

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

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

```
<W20110615-160604 agentPID="4162" agentSession="4296042782" cloneable="1"
 clusterName="thor" codeVersion="138"  isClone="1" scope="hpccdemo"
 state="completed" submitID="hpccdemo"
 token="X1lUMJ6oacON/1anTHTQW1JVHr1bbY8EWTSJhlDOrtYxmD13Z5ly4Qd26sEYVtxhW">
  <Action>run</Action>
  <Debug>
    <applyinstantecltransformations>1</applyinstantecltransformations>
    <applyinstantecltransformationslimit>100</applyinstantecltransformationslimit>
    <created_by>ws_workunits</created_by>
    <created_for>hpccdemo</created_for>
    <eclagentlog>//192.168.237.132/var/log/HPCCSystems/myeclagent/eclagent.06_15_11.log
    </eclagentlog>
    <targetclustertype>hthor</targetclustertype>
  </Debug>
  <Query fetchEntire="1">
    <Associated>
      <File crc="701142319" filename="libW20110615-160604.so" type="dll"/>
    </Associated>
    <Text>
      <Archive build="community_3.0.0" eclVersion="3.0.0">  <Query
       originalFilename="C:\DOCUME~1\Hpccdemo\LOCALS~1\Temp\TFR2CE.tmp">
       OUTPUT(&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///8BYXNjaWkAAWFzY2lpAAAYAAAAAA==
      </SchemaRaw>
      <totalRowCount>1</totalRowCount>
      <Value xsi:type="SOAP-ENC:base64"> CwAAAEhlbGxvIFdvcmxk </Value>
    </Result>
  </Results>
  <TimeStamps>
    <TimeStamp application="workunit">
      <Created ts="1308153964"> 2011-06-15T16:06:04Z </Created>
    </TimeStamp>
    <TimeStamp application="EclAgent" instance="localhost.localdom">
      <Started ts="1308153971"> 2011-06-15T16:06:11Z </Started>
    </TimeStamp>
    <TimeStamp application="EclAgent" instance="localhost.localdom">
      <Finished ts="1308153971"> 2011-06-15T16:06:11Z </Finished>
    </TimeStamp>
  </TimeStamps>
  <Timings>
    <Timing count="1" duration="1" max="1308040" name="WorkUnit_lockRemote"/>
    <Timing count="1" duration="6" max="6577412" name="SDS_Initialize"/>
    <Timing count="1" duration="0" max="704338" name="Environment_Initialize"/>
    <Timing count="1" duration="16" max="16414003" name="Process"/>
  </Timings>
  <Workflow>
    <Item mode="normal" state="done" type="normal" wfid="1">
      <Schedule/>
    </Item>
  </Workflow>
</W20110615-160604>
```

ECL Command Line Interface

The ECL Command Syntax

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

--version	displays version info.
Arguments	
deploy	Create a workunit from an ecl file, archive, or dll
publish	Add a workunit to a query set
unpublish	Remove a query from a query set
run	Run the given ecl file, archive, dll, wuid, or query
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
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
ECLCC_PATH
```

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

ecl deploy

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

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

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

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

Examples:

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

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

ecl deploy	Creates a workunit on the HPCC system from the given ECL text, file, archive, shared object, or dll. The workunit is created in the *compiled* state.
Arguments	
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
Options	
-n, --name	The published 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 query to HPCC as text rather than as a generated 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
-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.

ecl publish

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

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

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

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

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

Examples:

```
ecl publish roxie findperson.ecl --name=FindPersonService -A
ecl publish roxie W20150914-125557 --name=FindPersonService -A
ecl publish roxie libW20150914-125557.so --name=FindPersonService -A
ecl publish roxie ArchiveQuery.xml --name=FindPersonService -A
ecl publish roxie - --name=FindPersonService --activate
ecl publish roxie findperson.ecl --name=FindPersonService --no-activate
ecl publish roxie ArchiveQuery.xml --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.
Arguments	
target	The target cluster to which to publish
wuid	The workunit id to publish
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
Options	
-n, --name	The published query 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.
--daliip=	IP address or hostname of the remote Dali to use for remote logical file lookups.
--update-dfs	Update local DFS info if remote DALI has changed
---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=<sec>	Maximum time to wait for cluster finish updating
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--main	The definition to use from legacy ECL repository
--ecl-only	Send ECL query to HPCC as text rather than as a generated archive
--limit=<limit>	Sets the result limit for the query, defaults to 100
-f<option>[=value]	Set an ECL option (equivalent to #option)
-Dname=value	Override the definition of a global attribute 'name'
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
-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.

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> <file> [--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> [--name=<val>][--input=<file|xml>][--wait=<i>]

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

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

Examples:

```
ecl run thor findperson.ecl --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
Arguments	
target	The target cluster to which to publish
wuid	The workunit id to run
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
Options	
-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.
--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.
-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 query to HPCC as text rather than as a generated archive
--limit	Sets the result limit for the query, defaults to 100
-f<option>[=value]	set an ECL option (equivalent to #OPTION in ECL)
eclcc Options	
-I <path>	Add path to locations to search for ECL imports (e.g., -I C:\MyECL\ )
-L <path>	Add path to locations to search for system libraries
--manifest	Specify path to manifest file
-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.
-f-xxx	Will pass the option -xxx to eclcc

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.
Arguments	
wuid	The workunit from which to return results.
Options	
--noroot	Suppresses the <Result> root tag in the XML returned.
--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)

ecl activate

ecl activate <queryset> <query_id>

Example:

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

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

ecl deactivate

ecl deactivate <queryset> <active_alias>

Example:

```
ecl deactivate Roxie FindpersonService
```

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

ecl queries list

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

Examples:

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

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

ecl queries copy

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

Examples:

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

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

ecl queries copy-set

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

Examples:

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

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

ecl queries config

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

Examples:

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

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

ecl packagemap add

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

Examples:

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

ecl packagemap add	Calls the packagemap add command
Actions	
add	Adds a package map to the target cluster
Arguments	
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
Options	
-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
-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 package map
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 package map and make the spec package map active.
Arguments	
target	The target containing the package map to activate
packagemap	name of package map 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 package map.
Arguments	
target	The target containing the package map to deactivate
packagemap	Name of package map 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 package map names
Arguments	
target	The target containing the package map 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 package map info
Arguments	
target	The target containing the package map 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 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.
Actions	
add-part	Adds additional package map content to an existing package map
Arguments	
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
Options	
--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
-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 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.
Actions	
get-part	Fetches the given part from the given package map
Arguments	
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
Options	
--global-scope	The specified package map is sharable across multiple targets
-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 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.
Actions	
remove-part	Removes the given part from the given package map
Arguments	
target	Name of target to use
packagemap	Name of the package map containing the part
partname	Name of the part to remove
Options	
--global-scope	The specified package map is sharable across multiple targets
-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 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.
Actions	
validate	Validates package map info
Arguments	
filename	The filename containing the package map info to validate
target	The target containing the package map 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 package map that is active for the given target
--pmid=<packagemapid>	Validates the given package map
--queryid	Validate the files for the given queryid if they are mapped in the package map
--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 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
Actions	
copy	Copies a package map from one target to another
Arguments	
path	Path to the source package map to copy.
target	The target to copy the package map to
Options	
-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
-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)

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

ecl bundle info	Lists information about a bundle
Options	
<bundleName>	A bundle filename, a bundle folder, a bundle name, or a URL.
--version	Specify a version of the bundle
-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 --keepprior
```

ecl bundle install	Installs a bundle
Options	
<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
--keepprior	Do not remove any previous versions of the bundle
--update	Update an existing installed bundle
-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).

ecl bundle uninstall

ecl bundle uninstall <bundleName>

Examples:

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

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

ecl bundle list

ecl bundle list <pattern>

Examples:

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

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

ecl bundle use

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

Example:

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

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

ecl roxie unused-files

ecl roxie unused-files <processName>

Examples:

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

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

ecl abort

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

Examples:

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

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

ecl status

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

Examples:

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

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

ecl getwuid

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

Examples:

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

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

ecl getname

ecl getname -wu <WUID>

Examples:

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

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

ECL Compiler

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

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

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

Using the ECL Compiler as a Stand Alone option

The ECL Compiler is normally used through the ECL IDE or Eclipse using the ECL plugin 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
-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.
-foption[=value]	Set an ecl option. See #OPTION in the *ECL Language Reference* for details.
-main <*ref*>	Compile definition <ref> from the source collection
-syntax	Perform a syntax check of the ECL
-platform=hthor	Generate code for hthor executable (default)
-platform=roxie	Generate code for roxie cluster
-platform=thor	Generate code for thor cluster

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

Output control options:

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

C++ options:

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

Other options:

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

Compiled Options:

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

Usage: a.out <options>

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

Examples

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

Running a basic ECL program using the command line compiler

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

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

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

  (including the quotes) into the file.

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

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

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

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

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

  ```
  eclcc hello.ecl
  ```

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

  on Linux systems:

  ```
     ./a.out
  ```

  on Windows systems:

  ```
     a.out
  ```

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

Compile with Options

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

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

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

on Linux systems:

```
   ./Hello
```

on Windows systems:

```
   Hello
```

This will result in the output of the following.

```
Hello world
```

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

on Linux systems:

```
   ./Hello -xml
```

on Windows systems:

```
   Hello -xml
```

This would result in the output of the following:

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

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

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

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

This would result in the output of the following:

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

Command Line DFU

Command Line Interface

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

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

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

Evaluation of options follows this order of precedence:

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

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

General Options:

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

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

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

dfuplus.ini

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

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

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

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

Spray Operations:

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

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

srcip	Optional. The IP address of the source machine. If omitted, the information must be supplied by the *srcxml* parameter.
srcfile	Optional. The path to the source file. This may contain wildcard characters (* and ?) to include multiple source files in the spray to a single *dstname*. If omitted, the information must be supplied by the *srcxml* parameter.
srcxml	The name of the XML file containing the information required for the *srcip* and *srcfile* parameters. This file may have been obtained by previous use of the savexml *operation*. This option provides the feature of combining multiple source files into a single resulting logical file in the HPCC.
dstname	The logical name of the destination file.
dstcluster	The name of the destination cluster.
prefix	Optional. Both of the following (separated by a comma):
filename{:*length***}**	Prepends the filename (optionally limited to *length* characters) to the data.
filesize{:[B|L][1-8]}	Prepends the size of the file to the data. Optionally, you can specify the format of that integer (**B** specifies big endian, **L** specifies little endian) and the size of integer to contain it (**1** to **8** bytes). If format and size are omitted, the default is L4. 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.
expireDays	Optional. A integer value indicating the number of days before automatically removing the file. If omitted, the default is -1 (never expires).
format	Optional. One of the following values: **fixed csv delimited xml recfmv recfmb** If omitted, the default is fixed.
fixed format options:	

recordsize	The fixed size of each record, in bytes.
csv/delimited options:	
encoding	Optional. One of the following: ascii, utf8, utf8n, utf16, utf16le, utf16be, utf32, utf32le, utf32be ; If omitted, the default is ascii.
maxrecordsize	Optional. The maximum size of each record, in bytes. If omitted, the default is 8192.
separator	Optional. The field delimiter. If omitted, the default is a comma (\,).
terminator	Optional. The record delimiter. If omitted, the default is line feed or carriage return line feed (\r,\r\n).
quote	Optional. The string quote character. If omitted, the default is single quote (').
xml format options:	
rowtag	The XML tag identifying each record. Required.
encoding	Optional. One of the following: utf8 utf8n utf16 utf16le utf16be utf32 utf32le utf32beIf omitted, the default is utf8.
maxrecordsize	Optional. The maximum size of each record, in bytes. If omitted, the default is 8192.

Examples:

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

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

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

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

//fixed spray example using the above spray.xml file to
          combine
// multiple source files into a single logical file
// in this case, zip._1_of_3, zip._2_of_3, and zip._3_of_3
```

```
          into zip1:
dfuplus action=spray srcxml=spray.xml
          dstcluster=thordstname=RTTEMP::myzip1 recordsize=5

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

//Multiple spray all .JPG and .BMP files under
// c:\import on 10.150.51.26 to single logical file
          LE::imagedb:
dfuplus action=spray srcip=10.150.51.26
          srcfile=c:\import\*.jpg,c:\import\*.bmp

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

Despray Operations:

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

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

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

Examples:

```
dfuplus action=despray dstip=10.150.50.14
   dstfile=c:\import\despray\timezones.txt srcname=RTTEMP::timezones.txt
//the spray.xml file contains:
```

```
<File directory="c:\import\"
   group="thor"
   modified="2004-04-27T14:58:38"
   name="zip"
   numparts="2"
   partmask="zip._$P$_of_$N$">
<Attr job="zip1"
   owner="rtaylor"
   recordSize="5"
   replicated="1"
   workunit="D20040427-111857"/>
<Part modified="2004-04-27T14:58:40"
   node="10.150.51.29"
   num="1"
   size="165"/>
<Part modified="2004-04-27T14:58:40"
   node="10.150.51.29"
   num="2"
size="165"/>
</File>
//despray example using the above spray.xml file to split a single
// logical file into multiple destination files
// in this case, zip._1_of_3, zip._2_of_3, and zip._3_of_3 from zip1:
dfuplus action=despray dstxml=spray.xml dstcluster=thor
          srcname=RTTEMP::myzip1


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

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

Copy Operations:

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

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

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

srcname	The logical name of the source file.
dstname	The logical name of the destination file.
dstcluster	The name of the destination cluster.
srcdali	Optional. The IP address of the source Dali server, if different from the destination Dali (associated with the ESP Server specified in the *server* option).
srcusername	Optional. The username to use to access the *srcdali*. If omitted, the General Options *username* is used.
srcpassword	Optional. The password to use to access the *srcdali*. If omitted, the General Options *password* is used.

preservecompression	Optional. A boolean flag (0 | 1) indicating whether to preserve the compression of the source file. If omitted, the default is 1.

Example:

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

Remove Operations:

The **remove** operation deletes a logical file from the system data store, optionally 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.

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 *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
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*:

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

Example:

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

Removesuper Operations:

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

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

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

Example:

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

Listsuper Operations:

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

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

superfile	The logical name of the superfile.

Example:

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

Status Operations:

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

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

wuid	The workunit identifier of the workunit.

Example:

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

Abort Operations:

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

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

wuid	The workunit identifier of the workunit.

Example:

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

Resubmit Operations:

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

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

wuid	The workunit identifier of the workunit.

Example:

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

Savexml Operations:

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

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.

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

Example:

```
dfuplus action=savexml srcname=RTTEMP::timezones.txt
          dstxml=flattimezones.xml
    // this results in the following XML file:
    <File directory="c:\thordata\rttemp"
          group="thor"
          modified="2004-06-18T14:17:16"
          name="timezones.txt"
          numparts="3"
          partmask="timezones.txt._$P$_of_$N$">
    <Attr job="timezones.txt"
          owner="rtaylor"
          recordSize="155"
          replicated="1"
          size="51305"
          workunit="D20040618-101716"/>
    <OrigName>rttemp::timezones.txt</OrigName>
    <Part modified="2004-06-18T14:17:18"
          node="10.150.50.15"
          num="1"
          size="17050"/>
    <Part modified="2004-06-18T14:17:17"
          node="10.150.50.18"
          num="2"
          size="17050"/>
    <Part modified="2004-06-18T14:17:17"
          node="10.150.50.16"
          num="3"
          size="17205"/>
    </File>
```

Monitor Operations:

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

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

event	The name of the user-defined event to trigger. This is used as the first parameter of the ECL EVENT function.
lfn	Optional. The name of the logical file in the DFU to look for. Using this option precludes using the *ip*, *file*, and *sub* options.
ip	Optional. The IP address or name of the server on which the physical file will reside. This may be omitted if the *file* option contains a full URL.
file	Optional. The fully qualified path of the physical file to look for. This may contain wildcard characters (* and ?).
sub	Optional. Specifies searching subdirectories for the physical file if the *file* option contains wild-card characters (* and ?).

shotlimit	Optional. The number of arrival events to generate before marking the DFU workunit as complete. A value of negative one (-1) indicates continuing until the workunit is manually aborted. If omitted, the default value is one (1).

Note the following caveats and restrictions:

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

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

3) Events are only generated on the polling interval.

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

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

Example:

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

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.

lfn	The logical file name of the source file
outformat	Optional. The format of the result. Valid options are: csv, xml, json, or ascii. The default is xml.
csvheader	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.

lfn	The logical file name of the source file
backup	Optional. A boolean flag (0 | 1) indicating whether to write the history to a file before erasing it. Default is 1 (enable).
dstxml	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>]

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

esdl xml

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

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

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

Examples:

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

esdl ecl

esdl ecl sourcePath outputPath [options].

sourcePath	The absolute path to the ESDL Definition file containing the EsdlService definition for the service.
outputPath	The absolute path to the location where ECL output is to be written.
-x, --expandedxml	Output expanded XML files.
--includes	If present, process all included files.
--rollup	If present, rollup all processed includes to a single ECL output file.
-cde	Specifies the HPCC Component files directory (location of xslt files).
--ecl-imports	Comma-delimited import list to be attached to the output ECL. Each entry generates a corresponding IMPORT statement.
--ecl-header	Text to include in header or target (generated) file (must be valid ECL).
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]

sourcePath	The absolute path to the ESDL Definition file containing the EsdlService definition for the service.
serviceName	Name of ESDL Service defined in the given ESDL file.
--version <version number>	Constrain to interface version
*--method <method name>[;<method name>]**	Constrain to list of specific method(s)
--xslt <xslt file path>	Path to '/xslt/esxdl2xsd.xslt' file to transform EsdlDef to XSD
--preprocess-output <raw output directory> :	Output preprocessed XML file to specified directory before applying XSLT transform
--annotate <all | none>	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 al' enables additional annotations such as collapsed, cols, form_ui, html_head and rows.
--noopt	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.
-opt,--optional <param value>	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
-tns,--target-namespace <target namespace>	The target namespace passed to the transform via the parameter 'tnsParam' used for the final output of the XSD.
-n <int> .	Number of times to run transform after loading XSLT. Defaults to 1
--show-inheritance	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
--no-arrayof	Supresses the use of the arrrayof 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.
-v|--verbose	display verbose information
-?/-h/--help	show usage page
Output	(srcdir|<outdir>)/filename.ecl

This generates XSD from the ESDL definition.

Examples:

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

esdl wsdl

esdl wsdl sourcePath serviceName [options]

sourcePath	The absolute path to the ESDL Definition file containing the EsdlService definition for the service.
serviceName	Name of ESDL Service defined in the given ESDL file.
--version <version number>	Constrain to interface version
*--method <method name>[;<method name>]**	Constrain to list of specific method(s)
--xslt <xslt file path>	Path to '/xslt/esxdl2xsd.xslt' file to transform EsdlDef to XSD
--preprocess-output <raw output directory> :	Output preprocessed XML file to specified directory before applying XSLT transform
--annotate <all | none>	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 al' enables additional annotations such as collapsed, cols, form_ui, html_head and rows.
--noopt	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.
-opt,--optional <param value>	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
-tns,--target-namespace <target namespace>	The target namespace passed to the transform via the parameter 'tnsParam' used for the final output of the XSD.
-n <int> .	Number of times to run transform after loading XSLT. Defaults to 1
--show-inheritance	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
--no-arrayof	Supresses the use of the arrrayof 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.
--wsdladdress	Defines the output WSDL file's location address
-v|--verbose	display verbose information
-?/-h/--help	show usage page
Output	(srcdir|<outdir>)/filename.ecl

This generates WSDL from ESDL definition.

Examples:

```
esdl wsdl MathService.ecm 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
-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.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 <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 <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) 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 MathSvc.1 MathSvc --config MathSvcCfg.xml
                  -s nnn.nnn.nnn.nnn -p 8010
```

Configuring ESDL binding methods

The DESDL 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>sometrxid</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.

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 <ESPProcessName> <ESPBindingName> [options]

ESPProcessName	The ESP Process name
ESPBindingName	The ESP Binding 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 unbind ESDL service based bindings.

To unbind a given ESDL binding, provide the ESP process name and the ESP binding which make up this ESDL binding.

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

Example:

```
esdl unbind-service myesp myServiceBinding
```

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://<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-service myesp 8003 MathSvc.1 MathSvc --config MathSvcCfg.xml -s nnn.nnn.nnn.nnn -p 8010
```

esdl unbind-method

esdl unbind-method <ESPProcessName> <ESPBindingName> <ESDLServiceName> <MethodName> [options]

ESPProcessName	The target ESP Process name
ESPBindingName	The target ESP binding name associated with this service
ESDLServiceName	The name of the ESDLService associated with the target method.
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 binding0.

To unbind a method, provide the target ESP process name (the ESP which hosts the service.)

You must also provide the ESP binding on which this service is configured to run, the name of the ESDL service, and the name of the method you are unbinding.

Example:

```
esdl unbind-method myesp myespbinding WsMyService mymethod
```

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
```