HPCC Systems®

HPCC Client Tools

Boca Raton Documentation Team

HPCC Client Tools

Boca Raton Documentation Team

We welcome your comments and feedback about this document via email to <docfeedback@hpccsystems.com> Please include **Documentation Feedback** in the subject line and reference the document name, page numbers, and current Version Number in the text of the message.

2013 Version 3.10.8.2

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.

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 to differentiate action steps from explanatory text, as shown here:

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

Installation

The installation program installs all client tools, including the ECLPlus, DFUPlus, and RoxieConfig. The installation program provides the option to select which tools to install.

- From the CD, your Build Server, or your download:
- Run **setup.msi** , then follow the prompts

System Requirements

The recommended system configuration is:

A Pentium III processor (or higher) and at least 128 megabytes (MB) of RAM.

Microsoft Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows 7, or higher.

A full installation requires less than 16 MB of disk space.

The optional Tutorial Data File is 100 MB.

In addition, you will need Internet Explorer 8 and have Active Scripting support enabled (Javascript). If browser security is set to High, you should add the ECLWatch IP (or DNS name) as a Trusted Site.

Adobe Reader 5 (or higher) is required to view the documentation PDF files.

GVC Viewer is used to display graphs in ECL Watch, and RoxieConfig GUI. It is installed automatically.

ECL Plus

Command Line Interface

ECLPlus.exe

eclplus *action= owner= user= password= cluster= server= queue= 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 URL or IP address of the ECL Watch server.
queue=	The name of the ECL server queue.
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, or any combination.

ECLPLUS.INI

All the options can be put directly on the command line, or placed in a file called ECLPLUS.INI. Options that do not change very often should be put in the ini file. For example:

```
server=10.150.50.12
cluster=training
queue=eclserver_training
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:

```
C:\>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:

```
C:\>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:

```
C:\>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 eclserver. It contains realtime 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:

```
C:\>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:

```
C:\>Eclplus jobname=myquery1 @datasetquery.txt
```

View the result of a certain workunit

You can look at specific workunit results, like this:

```
C:\>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:

```
C:\>Eclplus action=dump wuid= W20090226-100958-85552898
```

See the Workunit Dump section below for the result.

See the thor graph of a workunit:

The thor graph

```
C:\>Eclplus action=graph wuid=W20090226-100958-85552898
```

Aborting a workunit

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

```
C:\>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:

```
C:\>Eclplus @datasetquery.txt timeout=0
```

Eclplus will return immediately.

```
C:\>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:

```
C:\>Eclplus @datasetquery.txt output=o1.txt
C:\>type o1.txt
firstname lastname
Yanrui Ma
Richard Taylor
Richard Chapman
```

Also, you may specify the following output formats:

csv

```
C:\>Eclplus @datasetquery.txt format=csv
[QUERY 0]
"Yanrui ","Ma "
"Richard ","Taylor "
"Richard ","Chapman "
```

csvh

```
C:\>Eclplus @datasetquery.txt format=csvh
[QUERY 0]
"firstname","lastname"
"Yanrui ","Ma "
"Richard ","Taylor "
"Richard ","Chapman "
```

raw

```
C:\>Eclplus @datasetquery.txt format=raw
Yanrui     Ma
Richard    Taylor
Richard    Chapman
```

runecl

```
C:\>Eclplus @datasetquery.txt format=runecl
[QUERY 0]
[0]
firstname -> Yanrui
```

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

bin(ary)

```
C:\>Eclplus @datasetquery.txt format=bin
Yanrui Ma Richard Taylor Richard Chapman
```

Running queries in Interactive mode

Assuming you have the ECLPLUS.INI file as listed above, when you run eclplus without any options you will enter interactive mode:

```
C:\>Eclplus
Connected to:
SDS=172.16.20.13 Queue=eclserver_linux_7way Cluster=linux_7way
ECL>
```

Type in ? for a list of the available commands:

```
ECL> ?
```

Supported commands:

<ecl>	executes the supplied ecl
@file.ecl	executes the ecl in file.ecl
conn[ect] SDSserver	connects to the specified SDS server
disc[onnect]	disconnects from the active SDS server
l[ist] {line}	list command history, or retrieve one line
sta[rt] file.ecl	executes the ecl in file.ecl
setparam name=value	set global variable name to value
sav[e] file	save ecl to a local file
spo[ol] {file|off}	send query output to a file, or the console if 'off'
exit	exit ECLPlus
quit	exit ECLPlus
ver[sion]	display version information
help	displays this message
?	displays this message

Most of the commands are self-explanatory. You can run a query by typing in the ecl directly:

```
ECL> 1+1
Result
2
```

Or by running a file that contains the query:

```
ECL> @datasetquery.txt
firstname lastname
Yanrui  Ma
```

```
Richard Taylor
Richard Chapman
```

Workunit Dump

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

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

```
<W20110615-160604 agentPID="4162" agentSession="4296042782" cloneable="1"
 clusterName="thor" codeVersion="138"  isClone="1" scope="hpccdemo"
 state="completed" submitID="hpccdemo"
 token="X1lUMJ6oacON/1anTHTQW1JVHr1bbY8EWTSJhlDOrtYxmD13Z5ly4Qd26sEYVtxhW">
  <Action>run</Action>
  <Debug>
    <applyinstantecltransformations>1</applyinstantecltransformations>
    <applyinstantecltransformationslimit>100</applyinstantecltransformationslimit>
    <created_by>ws_workunits</created_by>
    <created_for>hpccdemo</created_for>
    <eclagentlog>//192.168.237.132/var/log/HPCCSystems/myeclagent/eclagent.06_15_11.log
    </eclagentlog>
    <targetclustertype>hthor</targetclustertype>
  </Debug>
  <Query fetchEntire="1">
    <Associated>
      <File crc="701142319" filename="libW20110615-160604.so" type="dll"/>
    </Associated>
    <Text>
      <Archive build="community_3.0.0" eclVersion="3.0.0">  <Query
       originalFilename="C:\DOCUME~1\Hpccdemo\LOCALS~1\Temp\TFR2CE.tmp">
       OUTPUT(&amp;apos;Hello World&amp;apos;); </Query> </Archive>
    </Text>
  </Query>
  <resultLimit>100</resultLimit>
  <Results>
    <Result fetchEntire="1" name="Result 1" sequence="0" status="calculated">
      <rowCount>1</rowCount>
      <SchemaRaw xsi:type="SOAP-ENC:base64"> UmVzdWx0XzEABPH///8BYXNjaWkAAWFzY2lpAAAYAAAAAA==
      </SchemaRaw>
      <totalRowCount>1</totalRowCount>
      <Value xsi:type="SOAP-ENC:base64"> CwAAAEhlbGxvIFdvcmxk </Value>
    </Result>
  </Results>
  <TimeStamps>
    <TimeStamp application="workunit">
      <Created ts="1308153964"> 2011-06-15T16:06:04Z </Created>
    </TimeStamp>
    <TimeStamp application="EclAgent" instance="localhost.localdom">
      <Started ts="1308153971"> 2011-06-15T16:06:11Z </Started>
    </TimeStamp>
    <TimeStamp application="EclAgent" instance="localhost.localdom">
      <Finished ts="1308153971"> 2011-06-15T16:06:11Z </Finished>
    </TimeStamp>
  </TimeStamps>
  <Timings>
    <Timing count="1" duration="1" max="1308040" name="WorkUnit_lockRemote"/>
    <Timing count="1" duration="6" max="6577412" name="SDS_Initialize"/>
    <Timing count="1" duration="0" max="704338" name="Environment_Initialize"/>
    <Timing count="1" duration="16" max="16414003" name="Process"/>
  </Timings>
  <Workflow>
    <Item mode="normal" state="done" type="normal" wfid="1">
      <Schedule/>
```

```
    </Item>
  </Workflow>
</W20110615-160604>
```

ECL Command Line Interface

The ECL Command Syntax

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

--version=	displays version info.
Arguments	
deploy	Create a workunit from an ecl file, archive, or dll
publish	Add a workunit to a query set
unpublish	Remove a query from a query set
run	Run the given ecl file, archive, dll, wuid, or query
activate	Activate a published query
deactivate	Deactivate the given query alias name
queries	List or manipulate queries and querysets
packagemap	execute package commands (for Roxie)

ecl.ini

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

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

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

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

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

Evaluation of options follows this order of precedence:

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

Environment Variables

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

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

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

ecl deploy

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

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

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

Examples:

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

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

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

ecl publish

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

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

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

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

Examples:

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

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

ecl publish	Publishes a query into a queryset. The query is created by adding a workunit to a queryset and assigning it a query name.
Arguments	
wuid	The workunit id to publish
ecl_file	The ECL text file to deploy
archive	TheECL archive to deploy
so | dll	The workunit dynamic linked library or shared object to deploy
Options	
-t, --target	The target cluster to associate workunit with
-n, --name	The workunit job name
-A, --activate	Activates query when published (default)
-A-, --no-activate	Does not activate query when published
--no-reload	Does not reload the queryset
--no-reload	Specifies to not request a reload of the Roxie cluster
--timeLimit=<sec>	Value to set for query timeLimit configuration
--warnTimeLimit=<sec>	Value to set for query warnTimeLimit configuration
--memoryLimit=<mem>	Value to set for query memoryLimit configuration. Format <mem> as 500000B, 550K, 100M, 10G, or 1T, etc.
--wait=<sec>	Maximum time to wait for cluster finish updating
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running eclwatch services
--port	The eclwatch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--main	The definition to use from legacy ECL repository
--ecl-only	Send ecl text to hpcc without generating archive
eclcc Options	

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

ecl unpublish

ecl unpublish <queryset> <query_id>

Example:

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

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

ecl run

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

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

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

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

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

Examples:

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

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

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

ecl activate

ecl activate <queryset> <query_id>

Example:

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

ecl activate	Activates a published query. This assigns a query to the active alias with the same name as the query.
Arguments	
queryset	The name of queryset containing query to activate
query_id	The query to activate
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running eclwatch services
--port	The eclwatch services port (Default is 8010)
-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 eclwatch services
--port	The eclwatch services port (Default is 8010)
-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 eclwatch services
--port	The eclwatch services port (Default is 8010)
-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 EclWatch 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 files referenced by the query
-A, --activate	Activates query when copied
--no-reload	Specifies to not request a reload of the Roxie cluster
-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 eclwatch services
--port	The eclwatch services port (Default is 8010)
-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 eclwatch services
--port	The eclwatch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl packagemap add

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

Examples:

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

ecl packagemap add	Calls the packagemap add command
Actions	
add	Adds a packagemap to the target cluster
Arguments	
target	The target to associate the packagemap with
filename	The name of the file containing packagemap information.
--daliip=	IP address or hostname of the remote Dali to use for logical file lookups
Options	
-O, --overwrite	Whether to overwrite existing information - true if present
-A, --activate	Activates packagemap
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running eclwatch services
--port	The eclwatch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl packagemap delete

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

Examples:

```
ecl packagemap delete roxie mypackagemap
```

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

ecl packagemap activate

ecl packagemap activate <target> <packagemap>

Example:

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

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

ecl packagemap deactivate

ecl packagemap deactivate <target> <packagemap>

Example:

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

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

ecl packagemap list

ecl packagemap list <target>

Examples:

```
ecl packagemap list roxie
```

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

ecl packagemap info

ecl packagemap info [options] <target>

Examples:

```
ecl packagemap info roxie
```

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

ecl packagemap validate

ecl packagemap validate <target> [<filename>]

Examples:

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

The packagemap validate command verifies that :

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

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

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

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

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

ecl roxie attach

ecl roxie attach <processName>

Examples:

```
ecl roxie attach myroxie
```

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

ecl roxie detach

ecl roxie attach <processName>

Examples:

```
ecl roxie detach myroxie
```

ecl roxie attach	Detach the roxie from Dali
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running eclwatch services
--port	The eclwatch services port (Default is 8010)
-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 eclwatch services
--port	The eclwatch services port (Default is 8010)
-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 roxie process state
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running eclwatch services
--port	The eclwatch 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, however, you can use the ECL Compiler in a stand alone manner, to create stand alone programs, or workunits. The ECL Compiler can read ECL code from standard input, or can read it from a specified input file. It compiles the code into an executable program (Such as an 'EXE' file in Windows). The resulting program, when executed, runs the job, writing any output to standard output. Alternatively, you could redirect the output to a file or pipe into another process. With the ECL Compiler, you do not need a supercomputer cluster to develop and run ECL code.

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

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

Usage: eclcc <options> ECL_file.ecl

General options:

-I <*path*>	Add path to locations to search for ecl imports
-L <*path*>	Add path to locations to search for system libraries
-o <*file*>	Specify name of output file (default a.out if linking to executable, or stdout)
-manifest	Specify path to manifest file listing resources to add
-foption[=value]	Set an ecl option (#option)
-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
-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
-g	Set an ecl option (#option)
-Wc,xx	Pass option xx to the c++ compiler
-shared	Generate workunit shared object instead of a stand-alone exe

Other options:

-b	Batch mode. Each source file is processed in turn. Output name depends on the input filename
-c	compile only (don't link)
-help, --help	Display help message
--help -v	Display verbose help message
--logfile <*file*>	Write log to specified file
-specs <*file*>	Read eclcc configuration from specified file
-v --verbose	Output additional tracing information while compiling
--version	Output version 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>
```

Command Line DFU

Command Line Interface

DFUPlus.exe

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

operation	One of the following actions: spray, despray, copy, remove, rename, list, add, addsuper, removesuper, listsuper, savexml, status, abort, resubmit, monitor
@filename	Optional. The name of a file containing necessary *options*. If omitted and no command line *options* are specified, the appropriate *options* must be in the DFUPLUS.INI file.
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.

The **DFUPlus.exe** application 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, or any combination. Command line *options* override any in the nominated *@filename*, which in turn override any in the DFUPLUS.INI file.

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. Options that rarely change should be put in the DFUPLUS.INI file. For example:

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

In all the examples below, we'll assume DFUPLUS.INI has the above content.

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

Spray Operations:

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

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

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

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

Examples:

```
//fixed spray example:
C:\>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:
C:\>dfuplus action=spray srcxml=c:\import\flattimezones.xml
          dstname=RTTEMP::timezones.txt dstcluster=thor recordsize=155

//csv spray example:
C:\>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:
C:\>dfuplus action=spray srcxml=spray.xml
          dstcluster=thordstname=RTTEMP::myzip1 recordsize=5

//xml spray example:
C:\>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:
C:\>dfuplus action=spray srcip=10.150.51.26
```

```
          srcfile=c:\import\*.jpg,c:\import\*.bmp

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

Despray Operations:

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

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

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

Examples:

```
C:\>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:
C:\>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:
C:\>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. It may also be used to copy files from other LexisNexis HPCC environments (using the *srcdali* 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.

Example:

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

Remove Operations:

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

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

name	The logical name of the file to remove.
nodelete	A boolean flag (0 | 1) indicating whether to physically delete the file in addition to removing its listing from the DFU. If omitted, the default is 0—physically delete the file.

Example:

```
C:\>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:

```
C:\>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:

```
C:\>dfuplus action=list name=*
```

Add Operations:

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

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

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

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

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

Example:

```
C:\>dfuplus action=add srcxml=flattimezones.xml dstname=flattimezones.txt
```

Addsuper Operations:

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

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

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

Example:

```
C:\>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:

```
C:\>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:

```
C:\>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:

```
C:\>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:

```
C:\>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:

```
C:\>dfuplus action=resubmit wuid=W20050309-093020
```

Savexml Operations:

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

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

srcname The logical name of the source file.

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

Example:

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

```
      node="10.150.50.15"
      num="1"
      size="17050"/>
<Part modified="2004-06-18T14:17:17"
      node="10.150.50.18"
      num="2"
      size="17050"/>
<Part modified="2004-06-18T14:17:17"
      node="10.150.50.16"
      num="3"
      size="17205"/>
</File>
```

Monitor Operations:

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

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

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

Note the following caveats and restrictions:

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

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

3) Events are only generated on the polling interval.

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

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

Example:

```
C:\>dfuplus action=monitor event=MyEvent ip=edata10 file=/dz/arr.txt
```

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