DataCutter Reference Manual

2.1

Generated by Doxygen 1.2.8.1

Wed Dec 4 17:38:23 2002

Contents

Chapter 1

DataCutter Hierarchical Index

1.1 DataCutter Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

DataCutter Compound Index

2.1 DataCutter Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

DataCutter File Index

3.1 DataCutter File List

Here is a list of all documented files with brief descriptions:

Chapter 4

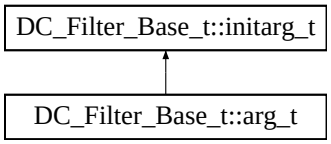
DataCutter Class Documentation

4.1 DC_Filter_Base_t::arg_t Class Reference

Argument passed to the process function containing input and output streams.

`#include <dc_filterservice.h>`

Inheritance diagram for DC_Filter_Base_t::arg_t::



Public Methods

- **arg_t** (void)
- ∼**arg_t** (void)
- int **insIndex** (char ∗sbName)

  Returns an index to ins based on the stream name provided, or -1 if not found.

- int **outsIndex** (char ∗sbName)

  Returns an index to outs based on the stream name provided, or -1 if not found.

- **DC_PipeInStream_t**∗ **insLookup** (char ∗sbName)

  Returns an input stream based on the stream name provided, or NULL if not found.

- **DC_PipeOutStream_t**∗ **outsLookup** (char ∗sbName)

  Returns an output stream based on the stream name provided, or NULL if not found.

- **DC_Buffer_t**∗ **insReadAny** (int ∗piFound=NULL, bool ∗rgStreamSet=NULL)

  Blocks until a buffer is available for any input stream.

- int **insNumEOW** (void)

  Provides the number of streams in ins that already hit EOW.

- bool **insAllEOW** (void)

  Returns true if all the streams in ins are at EOW.

- int **CloseOutputWaitInput** (int wWorkNum, bool fEndOfWork=false)

  Closes all outgoing streams and waits for all input streams to see EOS.

Public Attributes

- int **nins**
- **DC_PipeInStream_t∗ ins**
- int **nouts**
- **DC_PipeOutStream_t∗ outs**

Protected Methods

- int **_init** (DC_RemoteFilterCopy ∗pFilter_in, **DC_FilterService_t ∗pDC**)
- int **_resetInsEOW** (void)
- int **DrainInput** (int wWorkNum)

4.1.1 Detailed Description

Argument passed to the process function containing input and output streams.

The documentation for this class was generated from the following file:

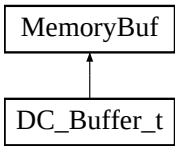
- **dc_filterservice.h**

4.2 DC_Buffer_t Class Reference

The buffer class.

`#include <dc_buffer.h>`

Inheritance diagram for DC_Buffer_t::



Public Methods

- **DC_Buffer_t** (void)
- **DC_Buffer_t** (int wMax)
- **DC_Buffer_t** (char *pBuf, int wMax, int wSize)
- **DC_Buffer_t** (const DC_Buffer_t &buf)
- void **consume** (void)

  Release a buffer.

- bool **getConsume** (void)

  Get the consume state.

- void **setConsume** (bool fConsume_in)

  Set the consume state.

Public Attributes

- DC_Buffer_t* **pNext**

  Optionally used by application (ex: list of buffers).

4.2.1 Detailed Description

The buffer class.

A DC_Buffer_t is the unit of transfer for streams. This class is simply a container for a chunk of contiguous memory, which can be statically or dynamically allocated. These buffers have a maximum size, and keep track of how much of that maximum is occupied. Data that will fit into the unused portion of the buffer can be appended. Extraction occurs from the head of the buffer, maintaining a extraction pointer for multiple extraction operations. An application can also just get a pointer and length of the data in the buffer, and use it in place.

When finished with a buffer, the **consume**() (p. 10) method should be called, which may free the space. Heap allocated memory is handled correctly based on the constructor used. Generally avoiding static DC_Buffer_t objects with the use of **consume**() (p. 10) is advised, unless **setConsume**() (p. 9) with the value true is explicitly called.

4.2.2 Member Function Documentation

4.2.2.1 void DC_Buffer_t::consume (void) [inline]

Release a buffer.

Called on a buffer returned from read that a filter will not need any longer. (Depending how memory was allocated for the buffer data area, this may or may not free the actual buffer space.)

The documentation for this class was generated from the following files:

- **dc_buffer.h**
- dc_buffer.cpp

4.3 DC_Filter_Base_t Class Reference

Base filter class.

`#include <dc_filterservice.h>`

Public Methods

- **DC_Filter_Base_t** (void)
- virtual ~**DC_Filter_Base_t** (void)
- char* **FindAppFilterConfig** (const char *sbName, int wId=-1)

  Searches the console config file for the section [AppName.FilterName].

- long **FindAppFilterConfigInt** (const char *sbName, int wId=-1, bool fInterpretUnits=true)
- double **FindAppFilterConfigDbl** (const char *sbName, int wId=-1)
- void **FilterLock** (void)

  Filter service provided method.

- void **FilterUnlock** (void)

  Filter service provided method.

- virtual DC_RTN_t **init** (**initarg_t** &arg)=0

  User-defined interface method.

- virtual DC_RTN_t **process** (**arg_t** &arg)=0

  User-defined interface method.

- virtual DC_RTN_t **finalize** (void)=0

  User-defined interface method.

Public Attributes

- **DC_FilterService_t*** **pDC**

  The service which instantiates the filter in a remote process.

- const char* **sbFilterName**

  The name of the filter.

4.3.1 Detailed Description

Base filter class.

Applications should subclass this abstract base class for every filter, then provide definitions for the virtual functions init, process, and finalize.

The documentation for this class was generated from the following file:

- **dc_filterservice.h**

4.4 DC_FilterInstance_t Class Reference

`#include <dc_instance.h>`

Public Methods

- **DC_FilterInstance_t** (void)
- **~DC_FilterInstance_t** (void)
- DC_WorkHandle_t **AppendWork** (**DC_Work_t** &work, bool fWait=true)

  appends the given work to a running instance, returns handle.

Public Attributes

- **DC_Filter_Base_t::arg_t arg**

  fDetached == F -> used by console process to recv results.

Protected Types

- enum { **INIT**, **WAIT**, **WORK**, **STOP**, **DONE** }

Protected Methods

- Types::uint **find_filter** (const char *sbName)
- DC_ConsoleStreamSpec* **find_stream** (const char *sbName, Types::uint &iStream)
- bool **_isDetached** (void)
- void **_Delete** (void)
- int **_Reset** (**DC_FilterService_t** *pDC_in, DC_ConsoleProcessState *pConsole_in)
- int **_ProcessLayout** (**DC_FilterLayout_t** &layout)
- int **_ProcessPlacement** (**DC_Placement_t** &placement)
- int **_StartInstance** (void)
- int **_StopInstance** (void)
- int **_WaitUntilInstanceDone** (void)
- int **_InstanceExit_Reader** (void)

Protected Attributes

- **DC_FilterService_t*** **pDC**
- DC_ConsoleProcessState* **pConsole**
- int **wInstanceNum**
- char* **sbLayoutName**
- bool **fDetached**
- enum DC_FilterInstance_t:: { ... } **eState**
- int **wNextWorkNum**
- int **wRemoteInstances**

  total remote processes used for this instance.

- int **cRemoteInstanceExit**

  count of remote processes that finished.

- DArray<DC_ConsoleFilterSpec*> **daFilterSpec**
- DArray<DC_ConsoleStreamSpec*> **daStreamSpec**
- DC_Mutex **mutex**

4.4.1 Detailed Description

Encapsulates the state of a set of filters being executed, and work being directed at them. This is of use within the Console Process called by the application main() or front end.

Multiple instances all exist within a single remote process. That remote process is assumed to exist before this class can do anything.

4.4.2 Member Enumeration Documentation

4.4.2.1 anonymous enum [protected]

INIT - after ctor called WAIT - filters instantiated on remote libs, no work to do WORK - processing work STOP - currently being stopped DONE - instance is finished

4.4.3 Constructor & Destructor Documentation

4.4.3.1 DC_FilterInstance_t::DC_FilterInstance_t (void)

ctor

4.4.3.2 DC_FilterInstance_t::~DC_FilterInstance_t (void)

dtor

4.4.4 Member Function Documentation

4.4.4.1 DC_WorkHandle_t DC_FilterInstance_t::AppendWork (DC_Work_t & *work*, bool *fWait* = true)

appends the given work to a running instance, returns handle.

Append the given work buffer to this running instance.

Inputs: work - uow to add to the new instance fWait - T=indicates if user will eventually call DC.WaitWork() F=automatic reap of work state when finished Return: int - 0+=work handle, <0=failure

4.4.4.2 int DC_FilterInstance_t::_ProcessLayout (DC_FilterLayout_t & *layout*) [protected]

parse and validate the configinfo based layout, and fill out the specs in our instance.

case "thru": Indicated by a single filter with the name <console>. Can only have ins (we may remove this restiction in the future if needed). This filter is virtual, and no thread is created, as done for a normal remote process filter. The thread of the console process main is used instead. All the remote process structures are created but the ins[] struct necessarily needs to be different, because instances are dynamically created and stopped. The console's filterinstance obj is where the ins will be parked.

4.4.4.3 int DC_FilterInstance_t::_ProcessPlacement (DC_Placement_t & *placement*) [protected]

parse and validate the placement information, check which remote processes are not in daPlacementSpec and add them.

4.4.4.4 int DC_FilterInstance_t::_StartInstance (void) [protected]

generate and send the start instance command for each remote process

4.4.4.5 int DC_FilterInstance_t::_StopInstance (void) [protected]

Sends stop messages to all remote processes for this instance. The reader will later recv the INSTANCEEXIT packets and adjust the instance state accordingly.

Return: int - number of INSTANCEEXIT packets to expect for this instance

4.4.4.6 DC_ConsoleStreamSpec * DC_FilterInstance_t::find_stream (const char * *sbName*, Types::uint & *iStream*) [protected]

look for a stream by name, and if not found, create a new one

4.4.5 Member Data Documentation

4.4.5.1 enum { ... } DC_FilterInstance_t::eState [protected]

INIT - after ctor called WAIT - filters instantiated on remote libs, no work to do WORK - processing work STOP - currently being stopped DONE - instance is finished

4.4.5.2 bool DC_FilterInstance_t::fDetached [protected]

false if [filter.<console>] is detected in filterlayout, which causes a pseudo-filter to be created for when results are funnelled back through the main console thread.

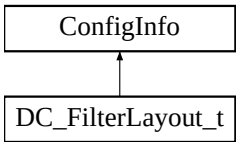
The documentation for this class was generated from the following files:

- **dc_instance.h**
- dc_instance.cpp

4.5 DC_FilterLayout_t Class Reference

`#include <dc_filterlayout.h>`

Inheritance diagram for DC_FilterLayout_t::



Public Methods

- **DC_FilterLayout_t** (void)
- ~**DC_FilterLayout_t** (void)
- char* **getName** (void)

  gets the symbolic name for the layout.

- void **setName** (char *sbName_in)

  sets the symbolic name for the layout.

- int **Add** (char *sbFilterName, char *sbInsList, char *sbOutsList, bool fAllowCopies=true)

Protected Attributes

- char* **sbName**

4.5.1 Detailed Description

allows for in-code creation of placement for a set of filters, instead of reading it from an external file.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 DC_FilterLayout_t::DC_FilterLayout_t (void)

ctor

4.5.2.2 DC_FilterLayout_t::~DC_FilterLayout_t (void)

dtor

The documentation for this class was generated from the following files:

- **dc_filterlayout.h**
- dc_filterlayout.cpp

4.6 DC_FilterService_t Class Reference

Main controlling class.

`#include <dc_filterservice.h>`

Public Types

- typedef **DC_Filter_Base_t*** (* **FilterFactory_f**)(char *)
- typedef **DC_Policy_Base_t*** (* **PolicyFactory_f**)(char *)

Public Methods

- **DC_FilterService_t** (void)
- **~DC_FilterService_t** (void)
- int **init** (char *sbAppName_in, FilterFactory_f fcnFilterFactory_in, int *pargc, char ***pargv, PerfCallback_f fcnPerfCallback_in=NULL)
- void **setPolicyFactory** (PolicyFactory_f fcnPolicyFactory_in)
- bool **isRemoteProcess** (void)
- int **RemoteProcess** (void)
- char* **getAppName** (void) const
- int **PlacementPlanning** (**DC_FilterLayout_t** &layout, **DC_Work_t** &work, **DC_Placement_t** &placement)
- int **ReuseFilterInstance** (**DC_FilterLayout_t** &layout, **DC_Work_t** &work, **DC_Placement_t** &placement, **DC_FilterInstance_t** *pinstance)
- int **NewFilterInstance** (**DC_FilterLayout_t** &layout, **DC_Work_t** &work, **DC_Placement_t** &placement, **DC_FilterInstance_t** *&pinstance)
- int **GetFilterInstance** (**DC_FilterLayout_t** &layout, **DC_Work_t** &work, **DC_Placement_t** &placement, **DC_FilterInstance_t** *&pinstance)
- int **StopFilterInstance** (**DC_FilterInstance_t** *&pinstance, bool fWait=true)
- int **ProbeWork** (DC_WorkHandle_t wh, int *status)
- int **WaitWork** (DC_WorkHandle_t wh)

  Wait for a unit of work to finish.

- DC_WorkHandle_t **WaitAnyWork** (void)

  Wait for any appended work to finish.

- int **ExitStatus** (DC_WorkHandle_t wh)
- void **FilterLock** (void)
- void **FilterUnlock** (void)
- char* **FindAppConfig** (const char *sbName, const char *sbFilterName=NULL, int wId=-1)

  Examines the section [AppName] of the console configuration file.

- long **FindAppConfigInt** (const char *sbName, const char *sbFilterName=NULL, int wId=-1, bool fInterpretUnits=true)

  Examines the section [AppName] of the console configuration file.

- double **FindAppConfigDbl** (const char *sbName, const char *sbFilterName=NULL, int wId=-1)

  Examines the section [AppName] of the console configuration file.

Protected Attributes

- char* **sbAppName**
- int* **pargc**
- char*** **pargv**
- FilterFactory_f **fcnFilterFactory**
- PolicyFactory_f **fcnPolicyFactory**
- DC_ConsoleProcessState* **pConsole**
- DC_RemoteProcessState* **pRemote**
- DC_ReaderServerLoop* **pReader**
- InetAddress **addrReader**

  For localhost communication with reader.

- ClientSocket **sockReader**
- InetAddress **addrLocal**

  Listen socket.

- ServerSocket **sockLocal**
- DC_Mutex **mutexFilterInternal**

  Currently used for ConsoleOutput/ConsolePrintf and >1 filter fd writes and FindAppConfig-XXX().

- bool **fCalledRun**

  Runtime state.

4.6.1 Detailed Description

Main controlling class.

Applications should instantiate one of these.

4.6.2 Member Function Documentation

4.6.2.1 int DC_FilterService_t::ExitStatus (DC_WorkHandle_t *wh*)

Reap the work entry, and return the exit status.

Inputs: wh - valid work handle from an AppendWork() or **WaitAnyWork**() (p. 18) call Return: int - <0 == DC_ERR_xxx, >=0 = exit status

4.6.2.2 char * DC_FilterService_t::FindAppConfig (const char * *sbName*, const char * *sbFilterName* = NULL, int *wId* = -1)

Examines the section [AppName] of the console configuration file.

Searches in the console config info for an entry in the seciton with the applications name (as specified in the **init**() (p. 19) call).

Inputs: sbName - entry to find wId - sub section value (ie: [AppName.wId]) -1 = just use the AppName as the section name sbFilterName - NULL=don't use, else [AppName.FilterName] or [AppName.FilterName.wId] Return: char * - NULL=not found, !NULL=value (must be freed by caller!)

4.6.2.3 int DC_FilterService_t::GetFilterInstance (DC_FilterLayout_t & *layout*, DC_Work_t & *work*, DC_Placement_t & *placement*, DC_FilterInstance_t *& *pinstance*)

Convenience routine to first try to reuse an existing filter instance, and if that fails, create a new one.

Inputs: layout - describes set of filters and constraints to instantiate work - uow to add to the new instance placement - describes where filters should be started Output: pinstance - ptr to context of the instance to use Return: int - DC_ERR_OK=success, DC_ERR_xxx=failure

4.6.2.4 int DC_FilterService_t::NewFilterInstance (DC_FilterLayout_t & *layout*, DC_Work_t & *work*, DC_Placement_t & *placement*, DC_FilterInstance_t *& *pinstance*)

Creates a new filter instance context for use by the application.

Inputs: layout - describes set of filters and constraints to instantiate work - uow to add to the new instance placement - describes where filters should be started Output: pinstance - ptr to a new instance context to use Return: int - DC_ERR_OK=success, DC_ERR_xxx=failure

4.6.2.5 int DC_FilterService_t::RemoteProcess (void)

called by remote process main thread

4.6.2.6 int DC_FilterService_t::StopFilterInstance (DC_FilterInstance_t *& *pinstance*, bool *fWait* = true)

Attempts a controlled shutdown of the given filter instance.

Inputs: pinstance - ptr to context of the instance to use fWait - true=wait for instance to stop, false=async stop Output: pinstance - NULL Return: int - DC_ERR_OK=success, DC_ERR_xxx=failure

4.6.2.7 DC_WorkHandle_t DC_FilterService_t::WaitAnyWork (void)

Wait for any appended work to finish.

Block until any work is finished, and return this work handle.

Return: int - <0 == DC_ERR_xxx, >=0 = work handle to use in **ExitStatus**() (p. 17)

4.6.2.8 int DC_FilterService_t::WaitWork (DC_WorkHandle_t *wh*)

Wait for a unit of work to finish.

Block until the given piece of work is finish, reap the entry, and return the exit status.

Inputs: wh - valid work handle from an AppendWork() call Return: int - <0 == DC_ERR_xxx, >=0 = exit status

4.6.2.9 int DC_FilterService_t::init (char * *sbAppName_in*, FilterFactory_f *fcnFilterFactory_in*, int * *pargc*, char *** *pargv*, PerfCallback_f *fcnPerfCallback_in* = NULL)

Initializes system and determines if we are the console.

Inputs: sbAppName_in - name of the application fcnFilterFactory_in - user fcn ptr that creates filter objects pargc, pargv - command line args Return: int - 0=ok, !0=error

The documentation for this class was generated from the following files:

- **dc_filterservice.h**
- dc_filterservice.cpp

4.7 DC_PipeInStream_t Class Reference

Input stream.

`#include <dc_filterservice.h>`

Public Methods

- char* **getName** (void)
- bool **isEndOfWork** (void)
- bool **isEndOfStream** (void)
- **DC_Buffer_t*** **read** (void)
- int **close** (void)

4.7.1 Detailed Description

Input stream.

This is passed into filter objects for accessing the streams, and are local to each filter thread (other than the pointers to potentially shared (colocated) DC_Remote objects).

The documentation for this class was generated from the following file:

- **dc_filterservice.h**

4.8 DC_PipeOutStream_t Class Reference

Output stream.

`#include <dc_filterservice.h>`

Public Methods

- char* **getName** (void)
- bool **isEndOfWork** (void)
- bool **isEndOfStream** (void)
- int **write** (**DC_Buffer_t** *pbuf, void *pUserArg=NULL)
- int **write** (**DC_Buffer_t** &buf, void *pUserArg=NULL)
- int **write_nocopy** (**DC_Buffer_t** *pbuf, void *pUserArg=NULL)
- int **write_nocopy** (**DC_Buffer_t** &buf, void *pUserArg=NULL)
- int **close** (bool fEndOfWork=false)

4.8.1 Detailed Description

Output stream.

This is passed into filter objects for accessing the streams, and are local to each filter thread (other than the pointers to potentially shared (colocated) DC_Remote objects).

write() does a deep copy of the buffer object and memory region as needed to allow the caller to modify the buffer immediately after this call returns. For example, a colocated sink will cause the buffer object and memory region to be duplicated and placed directly in the consumer's queue.

pUserArg is used for when there is a user-defined stream write policy in use. This extra arg is passed to the policy.write_copyset() callback to help decide which copyset the buffer should be directed to. A typical example would be to more simply pass along metadata, such as spatial location information, along with a data chunk to avoid needing to decode the whole data buffer to decide where it should be sent.

write_nocopy() (p. 21) uses the given buffer. If colocated with the sink this will be deposited directly in that Queue. Use of stack allocated **DC_Buffer_t** (p. 9) objects for this call is not recommended due to undesirable results if the stack frame is removed before the object is dequeued/used by a colocated consumer.

pUserArg is used for when there is a user-defined stream write policy in use. This extra arg is passed to the policy.write_copyset() callback to help decide which copyset the buffer should be directed to. A typical example would be to more simply pass along metadata, such as spatial location information, along with a data chunk to avoid needing to decode the whole data buffer to decide where it should be sent.

4.8.2 Member Function Documentation

4.8.2.1 int DC_PipeOutStream_t::write_nocopy (DC_Buffer_t & *buf*, void * *pUserArg* = NULL) [inline]

Be VERY careful using this version with local vars on the stack, since it could be needed long after the write call if we are colocated with the consumer.

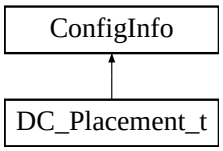
The documentation for this class was generated from the following file:

- **dc_filterservice.h**

4.9 DC_Placement_t Class Reference

`#include <dc_placement.h>`

Inheritance diagram for DC_Placement_t::



Public Methods

- **DC_Placement_t** (void)
- **~DC_Placement_t** (void)
- void **Add** (char *sbFilterName, char *sbHostName, int nCopiesLow=-1, int nCopiesHigh=-1)
- void **AddWritePolicy** (char *sbStreamName, char *sbSourceRe, char *sbSinkRe, char *sbPolicy)

Protected Attributes

- bool **fProcessed**

4.9.1 Detailed Description

allows for in-code creation of placement for a set of filters, instead of reading it from an external file.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 DC_Placement_t::DC_Placement_t (void)

ctor

4.9.2.2 DC_Placement_t::~DC_Placement_t (void)

dtor

4.9.3 Member Function Documentation

4.9.3.1 void DC_Placement_t::Add (char * *sbFilterName*, char * *sbHostName*, int *nCopiesLow* = -1, int *nCopiesHigh* = -1)

sbHostName - "<one_per_node>" == assign filters RR to running appds

The documentation for this class was generated from the following files:

- **dc_placement.h**
- dc_placement.cpp

4.10 DC_Policy_Base_t Class Reference

Base stream policy.

`#include <dc_filterservice.h>`

Public Methods

- virtual int **write_copyset** (int wMax, char *sbStream, **DC_Buffer_t** *pbuf, void *pUserArg)

4.10.1 Detailed Description

Base stream policy.

Applications should subclass this abstract base class to customize a stream's write policy when there are transparent copies.

4.10.2 Member Function Documentation

4.10.2.1 int DC_Policy_Base_t::write_copyset (int *wMax*, char * *sbStream*, DC_Buffer_t * *pbuf*, void * *pUserArg*) [inline, virtual]

Inputs: wMax - total number of transparent copies stream writes to sbStream - name of output stream buf - actual buffer written by the filter pUserArg - user defined argument (from write() call) Return: int - 0..wMax-1 copyset to write to

The documentation for this class was generated from the following file:

- **dc_filterservice.h**

4.11 DC_Work_t Class Reference

Work definition.

`#include <dc_work.h>`

Public Methods

- **DC_Work_t** (void)
- **DC_Work_t** (int wBufSize)
- ~**DC_Work_t** (void)

Public Attributes

- int **wWorkNum**
- **DC_Buffer_t buf**

4.11.1 Detailed Description

Work definition.

The documentation for this class was generated from the following file:

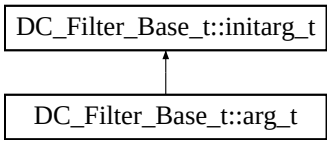
- **dc_work.h**

4.12 DC_Filter_Base_t::initarg_t Class Reference

Argument passed to the init function containing a work description.

`#include <dc_filterservice.h>`

Inheritance diagram for DC_Filter_Base_t::initarg_t::



Public Methods

- ~**initarg_t** (void)
- int **rankWithinCopySet** (void)

  Our rank within the single copyset on the current local host.

- int **maxWithinCopySet** (void)

  The number of transparent copies on the current local host.

- int **rankCopySets** (void)

  Total copyset info for this filter group instance.

- int **maxCopySets** (void)
- int **rankCopies** (void)

  Total filter transparent copy info for this filter group instance.

- int **maxCopies** (void)

Public Attributes

- int **argc**
- char** **argv**
- const char* **sbFilterName**
- const char * **sbFullName**
- const char * **sbCopyName**
- **DC_Work_t*** **pwork**

Protected Attributes

- DC_RemoteFilterCopy* **pFilter**

4.12.1 Detailed Description

Argument passed to the init function containing a work description.

A filter group is a logical description of interconnected filters. At runtime, an instance of the filter group is created, which may optionally contain more than one transparent copies of any filter. These transparent copies execute collectively in parallel to process work within the filter group instance. Within a filter group instance, all the transparent copies of a particular filter on a single host is called a copy set. The rank and max functions allow the filter to determine its rank in this two level heirarchy, which can be useful for read style filters that process disk resident data to partition the data among transparent copies of itself.

The documentation for this class was generated from the following file:

- **dc_filterservice.h**

4.13 DC_ConsoleStreamSpec::WriteRules_t Class Reference

rules for writer subsets.

`#include <dc_instance.h>`

Public Attributes

- DC_Regex **reSource**
- DC_Regex **reSink**
- bool **fNone**
- **DC_StreamWritePolicy_t eStreamWritePolicy**

4.13.1 Detailed Description

rules for writer subsets.

The documentation for this class was generated from the following file:

- **dc_instance.h**

Chapter 5

DataCutter File Documentation

5.1 DataCutter.h File Reference

```
#include "dc_filterservice.h"
#include "dc_instance.h"
```

Defines

- #define **H_DATACUTTER**

5.1.1 Detailed Description

5.1.2 Define Documentation

5.1.2.1 #define H_DATACUTTER

Value:

5.2 dc_buffer.h File Reference

```
#include "lib/dc_memory.h"
```

Compounds

- class **DC_Buffer_t**
  The buffer class.

Defines

- #define **H_DC_BUFFER**

5.2.1 Detailed Description

5.2.2 Define Documentation

5.2.2.1 #define H_DC_BUFFER

Value:

5.3 dc_error.h File Reference

Defines

- #define **H_DC_ERROR**
- #define **DC_ERR_OK** 0
- #define **DC_ERR_Error** -1
- #define **DC_ERR_NotConsoleProcess** -2
- #define **DC_ERR_NotRemoteProcess** -3
- #define **DC_ERR_InstanceNotRunning** -4
- #define **DC_ERR_RemoteInstanceError** -5
- #define **DC_ERR_InvalidLayout** -6
- #define **DC_ERR_InvalidPlacement** -7
- #define **DC_ERR_AppdCommFailed** -8
- #define **DC_ERR_RemoteProcessCommFailed** -9
- #define **DC_ERR_FailedConnect** -10
- #define **DC_ERR_FailedWrite** -11
- #define **DC_ERR_RemoteFailThreadCreate** -12
- #define **DC_ERR_RemoteFailObjCreate** -13
- #define **DC_ERR_RemoteFilterFail** -14
- #define **DC_ERR_InvalidWorkHandle** -15
- #define **DC_ERR_WorkInProgress** -16
- #define **DC_ERR_NoWorkFound** -17
- #define **DC_ERR_StreamEndOfWork** -18
- #define **DC_ERR_StreamEndOfStream** -19
- #define **DC_ERR_InvalidBuffer** -20
- #define **DC_ERR_InvalidStreamName** -21
- #define **DC_ERR_InsSelectAllShutdown** -22
- #define **DC_ERR_WorkFail** -23
- #define **DC_ERR_InstanceFail** -24

Functions

- char* **DC_strerror** (int err)

5.3.1 Detailed Description

5.3.2 Define Documentation

5.3.2.1 #define H_DC_ERROR

Value:

5.4 dc_filterlayout.h File Reference

```
#include "lib/dc_standard.h"
#include "lib/dc_configinfo.h"
```

Compounds

- class **DC_FilterLayout_t**

Defines

- #define **H_DC_FILTERLAYOUT**

5.4.1 Detailed Description

5.4.2 Define Documentation

5.4.2.1 #define H_DC_FILTERLAYOUT

Value:

5.5 dc_filterservice.h File Reference

```
#include <pthread.h>
#include "lib/dc_standard.h"
#include "lib/dc_argv.h"
#include "lib/dc_configinfo.h"
#include "lib/dc_socket.h"
#include "lib/dc_memory.h"
#include "lib/dc_map.h"
#include "lib/dc_timer.h"
#include "lib/dc_metric.h"
#include "lib/dc_mutex.h"
#include "dc_error.h"
#include "dc_buffer.h"
#include "dc_filterlayout.h"
#include "dc_work.h"
#include "dc_placement.h"
```

Compounds

- class **DC_Filter_Base_t::arg_t**
  Argument passed to the process function containing input and output streams.
- class **DC_Filter_Base_t**
  Base filter class.
- class **DC_FilterService_t**
  Main controlling class.
- class **DC_PipeInStream_t**
  Input stream.
- class **DC_PipeOutStream_t**
  Output stream.
- class **DC_Policy_Base_t**
  Base stream policy.
- class **DC_Filter_Base_t::initarg_t**
  Argument passed to the init function containing a work description.

Defines

- #define **H_DC_FILTERSERVICE**

Typedefs

- typedef void (∗ **PerfCallback_f**)(int inst, const char ∗sbFilterName, int rank, double ∗sa, long ∗sal)

Enumerations

- enum **DC_RTN_t** { **DC_RTN_OK**, **DC_RTN_CONT**, **DC_RTN_ABRT** }

Variables

- PerfCallback_f **g_fcnPerfCallback**

5.5.1 Detailed Description

5.5.2 Define Documentation

5.5.2.1 #define H_DC_FILTERSERVICE

Value:

5.6 dc_instance.h File Reference

```
#include "lib/dc_standard.h"
#include "lib/dc_argv.h"
#include "lib/dc_configinfo.h"
#include "lib/dc_socket.h"
#include "lib/dc_memory.h"
#include "lib/dc_map.h"
#include "lib/dc_timer.h"
#include "lib/dc_metric.h"
#include "lib/dc_mutex.h"
#include "lib/dc_cond.h"
#include "lib/dc_regex.h"
#include "packet-tags.h"
#include "dc_work.h"
#include "dc_filterlayout.h"
#include "dc_placement.h"
#include "dc_policy.h"
```

Compounds

- class **DC_ConsoleFilterSpec::CopySet_t**
- class **DC_ConsoleFilterSpec**
- class **DC_ConsoleStreamSpec**
- class **DC_FilterInstance_t**
- class **DC_ConsoleStreamSpec::WriteRules_t**

  rules for writer subsets.

Defines

- #define **H_DC_INSTANCE**

5.6.1 Detailed Description

5.6.2 Define Documentation

5.6.2.1 #define H_DC_INSTANCE

Value:

5.7 dc_placement.h File Reference

```
#include "lib/dc_standard.h"
#include "lib/dc_configinfo.h"
```

Compounds

- class **DC_Placement_t**

Defines

- #define **H_DC_PLACEMENT**

5.7.1 Detailed Description

5.7.2 Define Documentation

5.7.2.1 #define H_DC_PLACEMENT

Value:

5.8 dc_policy.h File Reference

```
#include "lib/dc_standard.h"
```

Defines

- #define **H_DC_POLICY**

Enumerations

- enum **DC_StreamWritePolicy_t** { **DC_StreamWritePolicy_RR** = 0, **DC_StreamWritePolicy_WRR**, **DC_StreamWritePolicy_DD**, **DC_StreamWritePolicy_UD** }

Functions

- const char* **DC_StreamWritePolicy_tostr** (**DC_StreamWritePolicy_t** ePolicy)
- **DC_StreamWritePolicy_t DC_StreamWritePolicy_toenum** (char *sbPolicy)

5.8.1 Detailed Description

5.8.2 Define Documentation

5.8.2.1 #define H_DC_POLICY

Value:

5.8.3 Enumeration Type Documentation

5.8.3.1 enum DC_StreamWritePolicy_t

control: buffer writes to a distributed stream buffer queue defined by: [console] stream_write_policy = < RR | WRR | DD | UD > changed to be defined by: [placement] stream = write src dst <policy> args

5.9 dc_work.h File Reference

```
#include "dc_buffer.h"
```

Compounds

- class **DC_Work_t**
  Work definition.

Defines

- #define **H_DC_WORK**
- #define **WORK_DEFAULT_BUFFER_SIZE** (1024)

Typedefs

- typedef int **DC_WorkHandle_t**

5.9.1 Detailed Description

5.9.2 Define Documentation

5.9.2.1 #define H_DC_WORK

Value:

Index