Reference Manual

Generated by Doxygen 1.7.4

Wed Oct 9 2013 09:16:20

Contents

Chapter 1

LDMS

LDMS means Lightweight Distributed Metric Service. It is an Application Programming Interface for publishing and gathering system metrics in a clustered environment. A metric set provider publishes metric sets to remote peers. The remote peers update their local copies of these metrics whenever they choose, i.e. the metric set published does not push the data, it is pulled by the client. Only the metric set publisher can change the contents of the metric set. Although the client can change its local copy, the data will be overwritten the next time the set is updated.

LDMS supports multiple network transports. The desired transport is elected when the transport handle is created. If the RDMA transport is available on your platform, then updating the contents of the metric set is done by the hardware via RDMA_READ and the metric set publisher's CPU is unaware and uninvolved with the transfer; i.e. with RDMA the metric set updates have zero CPU overhead.

A **Metric Set** is a named collection of **Metric** values. A Metric is a named and typed value. The contents of a metric set are updated as a single entity, i.e. it is not possible to fetch just one value from the server.

Metric sets have a generation number that allows the client to determine if any data in the metric set has changed since the last time it was updated. This is useful because the data is pulled by the client, not pushed by the producer.

1.1 Connection Management

The LDMS connection management model is similar to the tranditional sockets API. Peers are addressed with a struct sockaddr, servers `listen` for incoming connection requests and clients `connect` to servers.

The connection management API consists of the following functions:

- **ldms_create_xprt()** Create a transport instance.
- **ldms_listen()** Create a listening endpoint and respond to queries from peers.
- **ldms_connect()** Request a connection with a remote peer.

- **ldms_close()** Close a connection with a remote peer.

1.2 Creating Metric Sets

A Metric Set is defined and maintained by a compute node server. Metric Sets are created and updated in local memory. The principle functions for creating and destroying local metric sets are the following:

- **ldms_create_set()** Create a new metric set of the specified size.
- **ldms_mmap_set()** Use the memory specified as the contents of a metric set.
- **ldms_destroy_set()** Destroy a metric set.
- **ldms_add_metric()** Add a metric to a metric set.

1.3 Querying Metric Sets

A Metric Set consumer uses these function to query the server for the publisehd Metric Sets:

- **ldms_dir()** Return a list of published metric set names."
- **ldms_lookup()** Lookup and gather the detail of a particular metric set.
- **ldms_update()** Update the contents of a previously looked-up metric set.

1.4 Setting and Getting Metric Values.

A Metric Set producer sets the values of the metrics after creating the metric set and periodically thereafter as required. The client updates the metric set and gets the values of the metrics as required. The functions for doing this are as follows:

- **ldms_get_metric()** Get a metric handle.
- **ldms_set_metric()** Set the value of a metric.
- **ldms_get_XXX()** Get the value of a metric where the XXX specifies the data type

1.5 Push Notifications

Nominally, LDMS is a pull oriented transport, i.e. the Metric Set consumer calls a function to pull the data from the provider and update it's local copy of the data. There is, however, a way for the producer to initiate communication using Notifications.

- **ldms_register_notify_cb()** Register a callback function to be called when Notification Events are received for a particular Metric Set.
- **ldms_cancel_notify()** Stop receiving notification events for a metric set.
- **ldms_notify()** Send a notification event to all consumers registered to receive events on a metric set.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Directory Hierarchy

3.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 Fowler-Noll-Vo hash functions.

Implementation of Fowler-Noll-Vo hash functions. The algorithms are developed by Glenn Fowler, London Curt Noll and Phong Vo. For more information about FNV hash functions, please visit http://www.isthe.com/chongo/tech/comp/fnv/index.html.

Defines

- #define **FNV_32_PRIME** 0x01000193
- #define **FNV_64_PRIME** 0x100000001b3ULL

6.1.1 Detailed Description

Implementation of Fowler-Noll-Vo hash functions. The algorithms are developed by Glenn Fowler, London Curt Noll and Phong Vo. For more information about FNV hash functions, please visit http://www.isthe.com/chongo/tech/comp/fnv/index.html.

6.2 LDMS Connection Management

Typedefs

- typedef void(∗ ldms_log_fn_t)(const char ∗fmt,...)

  Create a transport handle.

Functions

- ldms_t **ldms_create_xprt** (const char ∗name, ldms_log_fn_t log_fn)

- void ldms_release_xprt (ldms_t x)

  Release a reference on a transport handle.
- const char * ldms_get_xprt_name (ldms_t x)

  Return the name of a transport.
- int ldms_connect (ldms_t x, struct sockaddr *sa, socklen_t sa_len)

  Request a connection to an LDMS host.
- int ldms_listen (ldms_t x, struct sockaddr *sa, socklen_t sa_len)

  Listen for connection requests from LDMS peers.
- int ldms_close (ldms_t x)

  Close a connection to an LDMS host.

6.2.1 Detailed Description

These functions initiate, terminate and manage connections with LDMS peers.

6.2.2 Typedef Documentation

6.2.2.1 typedef void(* ldms_log_fn_t)(const char *fmt,...)

Create a transport handle.

Metric sets are exported on the network through a transport. A transport handle is required to communicate on the network.

Parameters

name	The name of the transport type to create.
log_fn	An optional function to call when logging transport messages

Returns

A transport handle on success.
0 If the transport could not be created.

6.2.3 Function Documentation

6.2.3.1 int ldms_close (ldms_t *x*)

Close a connection to an LDMS host.

Parameters

x	The transport handle

Returns

0 if the connection was closed.
!0 if the transport handle is not valid or not connected.

6.2.3.2 int ldms_connect (ldms_t *x*, struct sockaddr * *sa*, socklen_t *sa_len*)

Request a connection to an LDMS host.

Parameters

x	The transport handle
sa	Socket address specifying the host address and port.
sa_len	The length of the socket address.

Returns

0 if the connection was established.
An error indicating why the connection failed.

6.2.3.3 const char* ldms_get_xprt_name (ldms_t *x*)

Return the name of a transport.

Return a character string representing the transport.

Parameters

x	The transport handle

Returns

A character string representing the local interface address on which the transport is communicating.
0 if the transport handle is invalid.

6.2.3.4 int ldms_listen (ldms_t *x*, struct sockaddr * *sa*, socklen_t *sa_len*)

Listen for connection requests from LDMS peers.

Parameters

x	The transport handle
sa	Socket address specifying the host address and port.
sa_len	The length of the socket address.

Returns

0 if a listening endpoint was successfully created.
An error indicating why the listen failed.

6.2.3.5 void ldms_release_xprt (ldms_t *x*)

Release a reference on a transport handle.

Drop a reference on the transport handle. When all references have been dropped, free all the resources associated with the transport.

Parameters

x	The transport handle to free.

6.3 LDMS Query Functions

Classes

- struct ldms_dir_s

  The format of the directory data returned by `ldms_dir` *request.*

Defines

- #define LDMS_DIR_F_NOTIFY 1

  Query the sets published by a host.
- #define **LDMS_XPRT_LIBPATH_DEFAULT** "/usr/local/lib/"
- #define **LDMS_DEFAULT_PORT** 50000
- #define **LDMS_LOOKUP_PATH_MAX** 511

Typedefs

- typedef struct ldms_dir_s * ldms_dir_t

  The format of the directory data returned by `ldms_dir` *request.*
- typedef void(* ldms_dir_cb_t)(ldms_t t, int status, ldms_dir_t dir, void *cb_arg)

  The `ldms_dir` *callback function.*

Enumerations

- enum ldms_dir_type { **LDMS_DIR_LIST**, LDMS_DIR_DEL, LDMS_DIR_ADD }

  ldms_dir callback function

Functions

- void ldms_dir_release (ldms_t t, ldms_dir_t dir)

  Release the resources consumed by a directory.
- void ldms_dir_cancel (ldms_t t)

  Cancel LDMS directory updates.
- int **ldms_dir** (ldms_t x, ldms_dir_cb_t cb, void *cb_arg, uint32_t flags)
- int ldms_lookup (ldms_t t, const char *name, ldms_lookup_cb_t cb, void *cb_arg)

  Query the contents of a metric set.

6.3.1 Detailed Description

These functions query LDMS peers for published metric sets.

6.3.2 Define Documentation

6.3.2.1 #define LDMS_DIR_F_NOTIFY 1

Query the sets published by a host.

This function queries the peer for the set of published metric sets. The caller specifies the callback function to invoke when the directory has been returned by the peer. If the caller specifies the `LDMS_DIR_F_NOTIFY` flag, the callback function will be invoked whenever the peer updates its set of published metrics.

See the ldms_dir_cancel function to cancel directory updates.

Parameters

x	The transport handle
cb	The callback function to invoke when the directory is returned by the peer.
cb_arg	A user context that will be provided as a parameter to the `cb` function.
flags	If set to LDMS_DIR_F_NOTIFY, the specified callback function will be invoked whenever the directory changes on the peer.

Returns

0 if the query was submitted successfully

6.3.3 Typedef Documentation

6.3.3.1 typedef void(* ldms_dir_cb_t)(ldms_t t, int status, ldms_dir_t dir, void *cb_arg)

The `ldms_dir` callback function.

This function is called when the directory request has been returned by the queried host.

Parameters

t	The transport handle
status	The status of the request
dir	Pointer to the returned directory data
cb_arg	The callback argument specified to the ldms_dir function.

6.3.4 Enumeration Type Documentation

6.3.4.1 enum ldms_dir_type

ldms_dir callback function

This function is called in response to a call to ldms_dir and thereafter whenever there is an update to the set directory. That is, this function may be called repeatedly until the transport is closed.

The application should call `ldms_dir_release` when it is finished processing the directory to free the associated resources.

There are three different types of directory updates: LDMS_DIR_LIST, LDMS_DIR_ADD, and LDMS_DIR_REPLY. An LDMS_DIR_LIST contains a list of all sets published by the server.LDMS_DIR_ADD contains a list of newly added sets since the callback was last called, and LDMS_DIR_REM contains a list of those sets removed. The user can expect a LDMS_DIR_LIST directory when the callback is first called, followed by any number of LDMS_DIR_ADD and LDMS_DIR_REM directory types. See the ldms_dir_s structure for more information.

Parameters

x	The transport handle
cb_arg	The callback argument specified in the call to `ldms_dir`
status	If zero, the query was successful, ENOMEM indicates that there was insufficient memory available to build the directory.
dir	Pointer to an ldms_dir_s structure on success, NULL otherwise. The directory update type

Enumerator:

LDMS_DIR_DEL A complete list of available metric sets

LDMS_DIR_ADD The listed metric sets have been deleted

The listed metric sets have been added

6.3.5 Function Documentation

6.3.5.1 void ldms_dir_cancel (ldms_t *t*)

Cancel LDMS directory updates.

This function cancels updates to the LDMS directory initiated by a call to ldms_dir.

Parameters

t	The transport handle

6.3.5.2 void ldms_dir_release (ldms_t *t*, ldms_dir_t *dir*)

Release the resources consumed by a directory.

This function is called by the application to release the resources used by an ldms_dir_t.

Parameters

t	The transport handle
dir	Pointer to an ldms_dir_s structure to be released.

6.3.5.3 int ldms_lookup (ldms_t *t*, const char * *name*, ldms_lookup_cb_t *cb*, void * *cb_arg*)

Query the contents of a metric set.

This function queries the peer for the detail of the metric set `path`. If the query is successful, the function puts the set handle in the pointer provided by the `s` parameter. This function takes a reference on the metric set. The memory for the metric set will not be released until all references have been removed. The ldms_release() function should be called when this metric set is no longer needed.

See the `ldms_dir()` function for detail on how to query a host for the list of published metric sets.

Parameters

t	The transport handle
name	The set name.
cb	The callback function to invoke when the lookup has completed.
cb_arg	A user context that will be provided as a parameter to the `cb` function.

Returns

0 if the query was submitted successfully.

6.4 LDMS Metric Set Management

Typedefs

- typedef void(* ldms_update_cb_t)(ldms_t t, ldms_set_t s, int status, void *arg)

  Prototype for the function called when update completes.

Functions

- void ldms_set_release (ldms_set_t s)

  Release a reference on the metric set.
- int ldms_update (ldms_set_t s, ldms_update_cb_t update_cb, void *arg)

  Update the metric set contents.
- int ldms_create_set (const char *set_name, size_t meta_sz, size_t data_sz, ldms_set_t *s)

  Create a Metric set.
- int ldms_mmap_set (void *meta_addr, void *data_addr, ldms_set_t *s)

  Map a metric set for remote access.

- void ldms_destroy_set (ldms_set_t s)

  Destroy a Metric set.
- const char * ldms_get_set_name (ldms_set_t s)

  Get the set name.
- uint32_t ldms_get_set_card (ldms_set_t s)

  Get the number of metrics in the metric set.
- ldms_set_t ldms_get_set (const char *set_name)

  Get a set by name.
- uint64_t ldms_get_meta_gn (ldms_set_t s)

  Get the metric schema generation number.
- uint64_t ldms_get_data_gn (ldms_set_t s)

  Get the metric data generation number.
- uint32_t ldms_get_max_size (ldms_set_t s)

  Return the maxmimum size of the metric set.
- uint32_t ldms_get_size (ldms_set_t s)

  Return the current size of the metric set.
- uint32_t ldms_get_cardinality (ldms_set_t s)

  Return the number of metrics in the set.

6.4.1 Detailed Description

These functions are used to create and destroy local metric sets and to update the contents of remote metric sets.

6.4.2 Typedef Documentation

6.4.2.1 typedef void(* ldms_update_cb_t)(ldms_t t, ldms_set_t s, int status, void *arg)

Prototype for the function called when update completes.

You cannot call back into the transport from this function or it will deadlock.

Parameters

t	The transport endpoint.
s	The metric set handle updated.
rc	0 if the update was successful, or an error value.
arg	The callback argument specified in the call to `ldms_update`.

6.4.3 Function Documentation

6.4.3.1 int ldms_create_set (const char * *set_name*, size_t *meta_sz*, size_t *data_sz*, ldms_set_t * *s*)

Create a Metric set.

Create a metric set on the local host. The metric set is added to the data base of metric sets exported by this host. The `set_name` parameter specifies the logical name of the metric set. This name is returned to the client when queried with the `ldms_dir` function.

Parameters

set_name	The name of the metric set.
meta_sz	The maximum meta data size.
data_sz	The maximum data size.
s	Pointer to ldms_set_t handle that will be set to the new handle.

Returns

0 on success

6.4.3.2 void ldms_destroy_set (ldms_set_t *s*)

Destroy a Metric set.

Called to destroy a local metric set created with ldms_create_set. This also invalidates any handles that remote users may have to this metric set.

Parameters

s	The ldms_set_t handle to destroy.

6.4.3.3 uint32_t ldms_get_cardinality (ldms_set_t *s*)

Return the number of metrics in the set.

Returns the number of metrics in the set.

Parameters

s	The ldms_set_t handle.

Returns

The number of metrics in the set.

6.4.3.4 uint64_t ldms_get_data_gn (ldms_set_t *s*)

Get the metric data generation number.

A metric set has a `generation` number that chnages when metric values are modified.

Parameters

s	The ldms_set_t handle.

Returns

The 64bit data generation number.

6.4.3.5 uint32_t ldms_get_max_size (ldms_set_t *s*)

Return the maxmimum size of the metric set.

The maximum size of a metric set is specified when it is creates. The library may make this larger, but will never make it smaller than specified. The size of the metric set determines how many metrics the set can hold.

Parameters

s	The ldms_set_t handle.

Returns

The maximum size of the metric set.

6.4.3.6 uint64_t ldms_get_meta_gn (ldms_set_t *s*)

Get the metric schema generation number.

A metric set has a `generation` number that chnages when metrics are added or removed from the metric set.

Parameters

s	The ldms_set_t handle.

Returns

The 64bit meta data generation number.

6.4.3.7 ldms_set_t ldms_get_set (const char * *set_name*)

Get a set by name.

Find a local metric set by name.

Parameters

set_name	The set name.

Returns

The ldms_set_t handle or 0 if not found.

6.4.3.8 uint32_t ldms_get_set_card (ldms_set_t *s*)

Get the number of metrics in the metric set.

Parameters

s	The ldms_set_t handle.

Returns

The number of metrics in the metric set. -1 otherwise.

6.4.3.9 const char* ldms_get_set_name (ldms_set_t *_set*)

Get the set name.

Return the name of the metric set.

Parameters

s	The ldms_set_t handle.

Returns

The metric set name as a character string.

6.4.3.10 uint32_t ldms_get_size (ldms_set_t *s*)

Return the current size of the metric set.

The current size of a metric set depends on how many metrics have been created in the set.

Parameters

s	The ldms_set_t handle.

Returns

The current size of the metric set.

6.4.3.11 int ldms_mmap_set (void * *meta_addr,* void * *data_addr,* ldms_set_t * *s*)

Map a metric set for remote access.

This service is used to map a local metric set for access on the network. The `addr` parameter specifies the address of the memory containing the metric set. This service is typically used to export metric sets that are created in the kernel.

Parameters

meta_addr	Address of the metric set meta data
data_addr	Address of the metric set data
s	Pointer to memory to receive handle.

Returns

0 Success

6.4.3.12 void ldms_set_release (ldms_set_t *s*)

Release a reference on the metric set.

Releases the reference obtained by ldms_lookup(). The specified set handle `s` should not be used after calling this function.

Parameters

s	The metric set handle.

6.4.3.13 int ldms_update (ldms_set_t *s*, ldms_update_cb_t *update_cb*, void * *arg*)

Update the metric set contents.

Updates the local copy of the metric set.

Parameters

s	The metric set handle to update.
update_cb	The function to call when the update has completed and the metric data has been updated.
cb_arg	A user defined context value to provide to the update_cb function.

Returns

0 on success or a non-zero value to indicate an error.

6.5 LDMS Metric Manaegment

Typedefs

- typedef void(* ldms_visit_cb_t)(struct ldms_value_desc *vd, union ldms_value *v, void *arg)

LDMS Visit Callback Function.

Functions

- int ldms_start_transaction (ldms_set_t s)

  Start an LDMS transaction.
- int ldms_begin_transaction (ldms_set_t s)

  Begin an LDMS transaction.
- int ldms_end_transaction (ldms_set_t s)

  End an LDMS transaction.
- struct ldms_timestamp const * ldms_get_timestamp (ldms_set_t s)

  Get the time the set was last modified.
- int ldms_is_set_consistent (ldms_set_t s)

  Returns TRUE if the metric set is consistent.
- ldms_metric_t ldms_add_metric (ldms_set_t s, const char *name, enum ldms_value_type t)

  Add a metric to the set.
- int ldms_get_metric_size (const char *name, enum ldms_value_type t, size_t *meta_sz, size_t *data_sz)

  Return the storage needed for metric.
- ldms_metric_t ldms_get_metric (ldms_set_t s, const char *name)

  Get the metric handle for a metric.
- ldms_metric_t ldms_make_metric (ldms_set_t s, struct ldms_value_desc *vd)

  Create a metric handle from a value descriptor.
- void ldms_metric_release (ldms_metric_t m)

  Release a reference on the metric.
- const char * ldms_get_metric_name (ldms_metric_t m)

  Returns the name of a metric.
- enum ldms_value_type ldms_get_metric_type (ldms_metric_t m)

  Returns the type of a metric.
- const char * ldms_type_to_str (enum ldms_value_type t)

  Get a metric set type as a string.
- enum ldms_value_type ldms_str_to_type (const char *name)

  Convert a string to an ldms_value_type.
- void ldms_visit_metrics (ldms_set_t s, ldms_visit_cb_t cb, void *arg)

  Iterate over the Metric Values in a Metric set.
- struct ldms_value_desc * ldms_first (struct ldms_iterator *i, ldms_set_t set)

  Get the first metric in the metric set.
- struct ldms_value_desc * ldms_next (struct ldms_iterator *i)

  Get the next metric in the metric set.
- void **ldms_print_set_metrics** (ldms_set_t _set)

6.5.1 Detailed Description

These functions are used to create and destroy local metrics and to get and set the value of a metric.

6.5.2 Typedef Documentation

6.5.2.1 typedef void(* ldms_visit_cb_t)(struct ldms_value_desc *vd, union ldms_value *v, void *arg)

LDMS Visit Callback Function.

A function of the type ldms_vist_cb_t is passed as an argument to ldms_visit_metrics(). The ldms_vist_metrics() function calls the specified `cb` function once for each metric in the set.

The parameters to the callback function are a pointer to an ldms_value_desc structure, a pointer ot the value itself, and a pointer to a callback parameter argument specified in teh ldms_visit_metrics() call.

Parameters

vd	Pointer to an ldms_value_desc structure containing the name and type of the metric.
v	Pointer to an ldms_value union.
arg	Callback argument provided to the ldms_visit_metrics() function.

6.5.3 Function Documentation

6.5.3.1 ldms_metric_t ldms_add_metric (ldms_set_t *s*, const char * *name*, enum ldms_value_type *t*)

Add a metric to the set.

Adds a metric to the metric set. The `name` of the metric must be unique.

Parameters

s	The ldms_set_t handle.
name	The name of the metric.
t	The type of the metric.

Returns

A handle to the newly created metric if successful.
0 if an error occured.

6.5.3.2 int ldms_begin_transaction (ldms_set_t *s*)

Begin an LDMS transaction.

A metric set provider may provide information to the peer that the metric set is consistent by updating metrics inside a transaction. In this way, the peer can determine if the metric set updated was in the middle of a transaction and therefore contains potentially inconsistent data.

Parameters

s	The ldms_set_t handle.

Returns

0 If the transaction was started.
!0 If the specified metric set is invalid.

6.5.3.3 int ldms_end_transaction (ldms_set_t *s*)

End an LDMS transaction.

Marks the metric set as consistent and time-stamps the data.

Parameters

s	The ldms_set_t handle.

Returns

0 If the transaction was started.
!0 If the specified metric set is invalid.

6.5.3.4 struct ldms_value_desc* ldms_first (struct ldms_iterator * *i*, ldms_set_t *set*) [read]

Get the first metric in the metric set.

These functions are used to iterate over the metrics in a metric set. The struct ldms_iterator is used to keep track of the current position in the set and must be provided in the call the ldms_next().

Parameters

i	Pointer to ldms_iterator structure.
set	The metric set handle.

Returns

An ldms_value_desc for the first metric in the metric set.
Null if the set is empty or invalid.

6.5.3.5 ldms_metric_t ldms_get_metric (ldms_set_t *s*, const char * *name*)

Get the metric handle for a metric.

Returns the metric handle for the metric with the specified name. This handle can then be used with the ldms_get_XXX functions to return the value of the metric.

Parameters

s	The metric set handle
name	The name of the metric.

Returns

The metric set handle or 0 if there is none was found.

6.5.3.6 const char* ldms_get_metric_name (ldms_metric_t *m*)

Returns the name of a metric.

Returns the name of the metric specified by the handle.

Parameters

m	The metric handle

Returns

A character string containing the name of the metric.

6.5.3.7 int ldms_get_metric_size (const char * *name*, enum ldms_value_type *t*, size_t * *meta_sz*, size_t * *data_sz*)

Return the storage needed for metric.

Given a metric name and a metric type, return the number of bytes the metric would consume in a metric set. This function is provided as a means to estimate the required size of the containing metric set given the names and types of the metrics it will contain.

Parameters

name	The name of the metric.
t	The LDMS metric type.
meta_sz	Pointer to the variable to receive the meta data size
data_sz	Pointer to the variable to receive the data size

Returns

0 on success or EINVAL if `t` is an unrecognized type.

6.5.3.8 enum ldms_value_type ldms_get_metric_type (ldms_metric_t *m*)

Returns the type of a metric.

Returns the type of the metric specified by the handle.

Parameters

m	The metric handle

Returns

The ldms_value_type for the metric.

6.5.3.9 struct ldms_timestamp const* ldms_get_timestamp (ldms_set_t *s*) [read]

Get the time the set was last modified.

Returns an ldms_timestamp structure that specifies when ldms_end_transaction was last called by the metric set provider. If the metric set provider does not update it's metric sets inside transactions, then this value is invalid. This value is undefined if the metric set is not consistent, see `ldms_is_set_consistent`.

Parameters

s	The metric set handle

Returns

ts A pointer to a timestamp structure.

6.5.3.10 int ldms_is_set_consistent (ldms_set_t *s*)

Returns TRUE if the metric set is consistent.

A metric set is consistent if it is not in the middle of being updated. This is indicated by the metric set provider if they are using transaction boundaries on metric set updates: see `ldms_begin_transaction` and `ldms_end_transaction`. Using transactions to update metric sets is computatationaly inexpensive, but optional.

6.5.3.11 ldms_metric_t ldms_make_metric (ldms_set_t *s*, struct ldms_value_desc * *vd*)

Create a metric handle from a value descriptor.

Returns the metric handle for the metric with the specified name. This handle can then be used with the ldms_get_XXX functions to return the value of the metric.

Parameters

s	The metric set handle
vd	The value descriptor.

Returns

A new metric handle or NULL if memory was not available.

6.5.3.12 void ldms_metric_release (ldms_metric_t *m*)

Release a reference on the metric.

Parameters

m	The metric set handle

6.5.3.13 struct ldms_value_desc* ldms_next (struct ldms_iterator * *i*) [read]

Get the next metric in the metric set.

Returns the ldms_value_desc for the next metric in the metric set.

Parameters

i	Pointer to ldms_iterator structure.

Returns

An ldms_value_desc for the next metric in the metric set.
Null if the set is empty or invalid.

6.5.3.14 int ldms_start_transaction (ldms_set_t *s*)

Start an LDMS transaction.

Metric sets are updated one metric at a time. A metric set provider may provide information to the peer that the metric set is consistent by updating metrics inside a transaction. In this way, the peer can determine if the metric set updated was in the middle of a transaction and therefore contains potentially inconsistent data.

Parameters

s	The ldms_set_t handle.

Returns

0 If the transaction was started.
!0 If the specified metric set is invalid.

6.5.3.15 enum ldms_value_type ldms_str_to_type (const char * *name*)

Convert a string to an ldms_value_type.

Parameters

name	Character string representing the type name.

Returns

The associated ldms_value_type.

6.5.3.16 const char* ldms_type_to_str (enum ldms_value_type *t*)

Get a metric set type as a string.

Returns a string representing the data type.

Parameters

t	The metric value type.

Returns

A character string representing the metric value type.

6.5.3.17 void ldms_visit_metrics (ldms_set_t *s*, ldms_visit_cb_t *cb*, void * *arg*)

Iterate over the Metric Values in a Metric set.

The function of the type ldms_vist_cb_t is passed as an argument to ldms_visit_metrics(). The ldms_vist_metrics() function calls the specified `cb` function once for each metric in the set.

Parameters

s	The metric set handle.
cb	The callback function to call.
arg	The `arg` parameter to provide to the callback function.

6.6 LDMS Notifications

Classes

- struct ldms_notify_event_s

  Notification event type.

Typedefs

- typedef struct ldms_notify_event_s * ldms_notify_event_t

  Notification event type.

- typedef void(∗ ldms_notify_cb_t)(ldms_t x, ldms_set_t s, ldms_notify_event_t e, void ∗arg)

  Nofication callback handler function.

Functions

- int ldms_register_notify_cb (ldms_t x, ldms_set_t s, int flags, ldms_notify_cb_t cb_fn, void ∗cb_arg)

  Register callback handler to receive update notifications.
- void ldms_event_release (ldms_t x, ldms_notify_event_t e)

  Release the resources consumed by an event.
- int ldms_cancel_notify (ldms_t x, ldms_set_t s)

  Cancel notifications.
- void ldms_notify (ldms_set_t s, ldms_notify_event_t e)

  Notify registered clients.

6.6.1 Detailed Description

These functions are used register for and deliver Notification Events.

6.6.2 Typedef Documentation

6.6.2.1 typedef void(∗ ldms_notify_cb_t)(ldms_t x, ldms_set_t s, ldms_notify_event_t e, void ∗arg)

Nofication callback handler function.

This is the function prototype for registered notification handlers.

Parameters

x	The transport endpoint.
s	The metric set handle
e	The notification event
arg	The user-supplied argument provided when the callback was registered.

6.6.2.2 typedef struct ldms_notify_event_s ∗ ldms_notify_event_t

Notification event type.

The ldms_notify_event_t should be initialized using one of the following functions:

∗ ldms_init_notify_modified ∗ ldms_init_notify_user_data

6.6.3 Function Documentation

6.6.3.1 int ldms_cancel_notify (ldms_t *x*, ldms_set_t *s*)

Cancel notifications.

Cancel notifications for a metric set

Parameters

x	The transport endpoint.
s	The metric set handle

Returns

0 Success
!0 An error was encountered. See errno for details.

6.6.3.2 void ldms_event_release (ldms_t *x*, ldms_notify_event_t *e*)

Release the resources consumed by an event.

This function is called by the application to release the resources used by an ldms_notify_event_t.

Parameters

t	The transport handle
dir	Pointer to an ldms_notify_event_t structure to be released.

6.6.3.3 void ldms_notify (ldms_set_t *s*, ldms_notify_event_t *e*)

Notify registered clients.

Notify clients that have registered callback handlers that the metric set has been modified. Note that this will unconditionally notify the clients whether or not a change has been made to the metric set. The expected usage is:

ldms_set_metric(m0, ...); ldms_set_metric(m1, ...); ldms_set_metric(m2, ...); ... ldms_notify(s);

Parameters

s	The metric set handle.
e	The event

6.6.3.4 int ldms_register_notify_cb (ldms_t *x*, ldms_set_t *s*, int *flags*, ldms_notify_cb_t *cb_fn*, void * *cb_arg*)

Register callback handler to receive update notifications.

If the metric set producer supports update notifications, the registered callback handler is invoked when the producer calls the `ldms_notify` service.

Parameters

x	The transport endpoint.
s	The metric set handle
flags	The events of interest. 0 means all events.
cb_fn	Pointer to the function to call to receive notifications
cb_arg	User-supplied argument to the cb_fn function.

Returns

0 on success
!0 on failure. Refer to `errno` for error details

Chapter 7

Directory Documentation

7.1 src/core/ Directory Reference

Files

- file **ldms.c**
- file **ldms.h**
- file **ldms_private.h**
- file **ldms_xprt.c**
- file **ldms_xprt.h**
- file **ldmsd.h**
- file **ogc_rbt.c**
- file **ogc_rbt.h**

7.2 src/ldmsd/ Directory Reference

Files

- file **ldms_ls.c**
- file **ldmsctl.c**
- file **ldmsd.c**
- file **ldmsd_store.c**

7.3 src/sampler/lustre/ Directory Reference

Files

- file fnv_hash.h
- file **lustre2_client.c**

- file **lustre2_mds.c**
- file **lustre2_oss.c**
- file lustre_sampler.c

  Lustre sampler common routine implementation.
- file lustre_sampler.h

  Lustre sampler header file.
- file str_map.c

  String-Object mapping utility.
- file str_map.h

  String-Object mapping utility.

7.4 src/sampler/ Directory Reference

Directories

- directory lustre

Files

- file gem_link_perf_util.c

  Utilities for the gem_link_perf sampler also used in ncsa_unified.
- file gem_link_perf_util.h

  Utilities for gem_link_perf/nic_perf_sampler also used in ncsa_unified.
- file geminfo.c

  /proc/geminfo data provider
- file **gemini.h**
- file meminfo.c

  /proc/meminfo data provider
- file ncsa_unified.c

  unified custom data provider for ncsa interested metrics. Combo of metrics from other samplers.
- file perfevent.c

  perfevent data provider
- file **procdiskstats.c**
- file procinterrupts.c

  /proc/interrupts data provider
- file procnetdev.c

  /proc/net/dev data provider
- file procnfs.c

  /proc/net/rpc/nfs data provider
- file procsensors.c

  *reads from proc the data that populates lm sensors (in*_input, fan*_input, temp*_input)*

- file procstatutil.c

  /proc/stat/util data provider
- file sampler_atasmart.c

  Collect S.M.A.R.T. attribute values.
- file sedc.c

  sedc data provider.
- file sysclassib.c

  reads from 1) all files in: /sys/class/infiniband/mlx4_{0/1}/ports/{1/2}/counters which have well-known names 2) /sys/class/infiniband/mlx4_{0/1}/ports/{1,2}/rate
- file vmstat.c

  /proc/vmstat data provider

7.5 src/ Directory Reference

Directories

- directory core
- directory ldmsd
- directory sampler
- directory store
- directory xprt

7.6 src/store/ Directory Reference

Files

- file **store_csv.c**
- file **store_flatfile.c**
- file **store_mysql.c**
- file **store_mysqlbulk.c**
- file **store_sos.c**

7.7 src/xprt/ Directory Reference

Files

- file **ldms_rdma.c**
- file **ldms_rdma_xprt.h**
- file **ldms_sock.c**
- file **ldms_sock_xprt.h**
- file **ldms_ugni.c**
- file **ldms_ugni.h**

Chapter 8

Class Documentation

8.1 atatsmart_set_size Struct Reference

Public Attributes

- size_t **tot_meta_sz**
- size_t **tot_data_sz**
- int **metric_count**

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

- src/sampler/sampler_atasmart.c

8.2 csv_store_handle Struct Reference

Public Attributes

- struct ldmsd_store * **store**
- char * **path**
- FILE * **file**
- int **printheader**
- char * **store_key**
- pthread_mutex_t **lock**
- void * **ucontext**

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

- src/store/store_csv.c

8.3 flatfile_metric_store Struct Reference

Store for individual metric.

Public Attributes

- FILE * file
- pthread_mutex_t lock
- char * path

8.3.1 Detailed Description

Store for individual metric.

8.3.2 Member Data Documentation

8.3.2.1 FILE* flatfile_metric_store::file

File handle

8.3.2.2 pthread_mutex_t flatfile_metric_store::lock

lock at metric store level

8.3.2.3 char* flatfile_metric_store::path

path of the flatfile store

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

- src/store/store_flatfile.c

8.4 flatfile_store_instance Struct Reference

Public Attributes

- struct ldmsd_store * **store**
- char * path
- char * **container**
- void * **ucontext**
- idx_t **ms_idx**

8.4.1 Member Data Documentation

8.4.1.1 char* flatfile_store_instance::path

(root_path)/(comp_type)

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

- src/store/store_flatfile.c

8.5 fset Struct Reference

Public Attributes

- ldms_set_t **sd**
- ldms_metric_t **metrichandles** [MAXMETRICSPERSET]

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

- src/sampler/sedc.c

8.6 gemini_coord_t Struct Reference

Public Attributes

- int **x**
- int **y**
- int **z**

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

- src/sampler/gemini.h

8.7 gemini_link_t Struct Reference

Public Attributes

- int **dir**
- double **bw**
- uint64_t **counters** [GEMINI_NUM_TILE_COUNTERS]

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

- src/sampler/gemini.h

8.8 gemini_nic_t Struct Reference

Public Attributes

- uint64_t **counters** [GEMINI_NUM_NIC_COUNTERS]

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

- src/sampler/gemini.h

8.9 gemini_state_t Struct Reference

Public Attributes

- int **my_nid**
- gemini_coord_t **my_coord**
- gemini_nic_t **nic**
- gemini_coord_t **neighbor** [GEMINI_NUM_LOGICAL_LINKS]
- gemini_tile_t **tile** [GEMINI_NUM_TILES]
- gemini_link_t **link** [GEMINI_NUM_LOGICAL_LINKS]
- int **num_tile_samples**
- int **tile_sample_index**

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

- src/sampler/gemini.h

8.10 gemini_tile_t Struct Reference

Public Attributes

- int **type**
- int **dir**
- uint64_t **counters** [GEMINI_NUM_TILE_COUNTERS]

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

- src/sampler/gemini.h

8.11 gni_dom_info_t Struct Reference

```
#include <ldms_ugni.h>
```

Public Attributes

- uint8_t **ptag**
- uint32_t **cookie**
- uint32_t **pe_addr**
- uint32_t **inst_id**

8.11.1 Detailed Description

uGNI Transport private data

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

- src/xprt/ldms_ugni.h

8.12 gni_dom_t Struct Reference

Public Attributes

- gni_dom_info_t **info**
- gni_cdm_handle_t **cdm**
- gni_nic_handle_t **nic**
- gni_cq_handle_t **src_cq**

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

- src/xprt/ldms_ugni.h

8.13 hostset Struct Reference

Public Types

- enum { **LDMSD_SET_CONFIGURED**, **LDMSD_SET_LOOKUP**, **LDMSD_SET_-BUSY**, **LDMSD_SET_READY** }

Public Member Functions

- **LIST_ENTRY** (hostset) entry

Public Attributes

- struct hostspec * **host**
- char * **name**
- struct ldmsd_store_policy_ref_list lsp_list
- enum hostset:: { ... } **state**
- pthread_mutex_t **state_lock**
- ldms_set_t **set**
- uint64_t **gn**
- int **refcount**
- pthread_mutex_t **refcount_lock**
- struct ldms_mvec * mvec
- uint64_t curr_busy_count
- uint64_t total_busy_count

8.13.1 Member Data Documentation

8.13.1.1 uint64_t hostset::curr_busy_count

The count of current busy access

8.13.1.2 struct ldmsd_store_policy_ref_list hostset::lsp_list

Store policy list that is related to this hostset.

8.13.1.3 struct ldms_mvec* hostset::mvec

Metric vector

8.13.1.4 uint64_t hostset::total_busy_count

The count of total busy access

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

- src/core/ldmsd.h

8.14 hostset_ref Struct Reference

Public Member Functions

- **LIST_ENTRY** (hostset_ref) entry

Public Attributes

- char * hostname
- struct hostset * **hset**

8.14.1 Member Data Documentation

8.14.1.1 char* hostset_ref::hostname

The hostname is here as a part of the configuration.

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

- src/core/ldmsd.h

8.15 hostspec Struct Reference

Public Types

- enum { **HOST_DISCONNECTED** = 0, **HOST_CONNECTING**, **HOST_CONNECTED** }
- enum { **ACTIVE**, **PASSIVE**, **BRIDGING** }

Public Member Functions

- **LIST_HEAD** (set_list, hostset) set_list
- **LIST_ENTRY** (hostspec) link

Public Attributes

- struct sockaddr_in **sin**
- char * **hostname**
- char * **xprt_name**
- int **connect_interval**
- int **sample_interval**
- enum hostspec:: { ... } **conn_state**
- pthread_mutex_t **conn_state_lock**
- enum hostspec:: { ... } **type**
- int **thread_id**
- struct event * **event**
- struct timeval **timeout**
- ldms_t **x**
- pthread_mutex_t **set_list_lock**

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

- src/core/ldmsd.h

8.16 kw Struct Reference

Public Attributes

- char * **token**
- int(* **action**)(char *kw, char *err_str)

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

- src/ldmsd/ldmsctl.c
- src/sampler/procnetdev.c
- src/sampler/sysclassib.c

8.17 ldms_atasmart Struct Reference

Public Attributes

- SkDisk * **d**

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

- src/sampler/sampler_atasmart.c

8.18 ldms_cancel_notify_cmd_param Struct Reference

Public Attributes

- uint64_t **set_id**

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

- src/core/ldms_xprt.h

8.19 ldms_context Struct Reference

Public Attributes

- sem_t **sem**
- sem_t * **sem_p**
- int **rc**

- union {
  struct {
  int **set_count**
  char * **set_list**
  size_t **set_list_len**
  uint32_t **flags**
  ldms_dir_cb_t **cb**
  void * **cb_arg**
  } **dir**
  struct {
  char * **path**
  ldms_lookup_cb_t **cb**
  void * **cb_arg**
  struct ldms_set * **set**
  } **lookup**
  struct {
  ldms_set_t **s**
  ldms_update_cb_t **cb**
  void * **arg**
  } **update**
  struct {
  ldms_set_t **s**
  ldms_notify_cb_t **cb**
  void * **arg**
  } **req_notify**
  };

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

- src/core/ldms_xprt.h

8.20 ldms_data_hdr Struct Reference

Public Attributes

- struct ldms_transaction **trans**
- uint32_t **pad**
- uint64_t **gn**
- uint64_t **size**
- uint64_t **meta_gn**
- uint32_t **head_off**
- uint32_t **tail_off**

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

- src/core/ldms.h

8.21 ldms_dir_cmd_param Struct Reference

Public Attributes

- uint32_t **flags**

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

- src/core/ldms_xprt.h

8.22 ldms_dir_reply Struct Reference

Public Attributes

- uint32_t **type**
- uint32_t **more**
- uint32_t **set_count**
- uint32_t **set_list_len**
- char **set_list** [0]

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

- src/core/ldms_xprt.h

8.23 ldms_dir_s Struct Reference

The format of the directory data returned by `ldms_dir` request.

`#include <ldms.h>`

Public Attributes

- enum ldms_dir_type type
- int more
- int set_count
- char * set_names [0]

8.23.1 Detailed Description

The format of the directory data returned by `ldms_dir` request.

8.23.2 Member Data Documentation

8.23.2.1 int ldms_dir_s::more

!0 if this is the first of multiple updates

8.23.2.2 int ldms_dir_s::set_count

count of sets in the set_name array

8.23.2.3 char* ldms_dir_s::set_names[0]

each string is null terminated.

8.23.2.4 enum ldms_dir_type ldms_dir_s::type

the type of update

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

- src/core/ldms.h

8.24 ldms_hello_cmd_param Struct Reference

Public Attributes

- uint32_t **msg_len**
- char **msg** [0]

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

- src/core/ldms_xprt.h

8.25 ldms_iterator Struct Reference

Metric value iterator.

```
#include <ldms.h>
```

Public Attributes

- struct ldms_set * **set**
- struct ldms_value_desc * **curr_desc**
- union ldms_value * **curr_value**
- uint32_t **curr_off**

8.25.1 Detailed Description

Metric value iterator.

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

- src/core/ldms.h

8.26 ldms_lookup_cmd_param Struct Reference

Public Attributes

- uint32_t **path_len**
- char **path** [LDMS_LOOKUP_PATH_MAX+1]

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

- src/core/ldms_xprt.h

8.27 ldms_lookup_reply Struct Reference

Public Attributes

- uint64_t **set_id**
- uint32_t meta_len
- uint32_t **data_len**
- uint32_t **xprt_data_len**
- char **xprt_data** [0]

8.27.1 Member Data Documentation

8.27.1.1 uint32_t ldms_lookup_reply::meta_len

server handle for set

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

- src/core/ldms_xprt.h

8.28 ldms_metric Struct Reference

Public Attributes

- struct ldms_set * **set**

- struct ldms_value_desc * **desc**
- union ldms_value * **value**

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

- src/core/ldms.h

8.29 ldms_mvec Struct Reference

Public Attributes

- int **count**
- ldms_metric_t **v** []

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

- src/core/ldms.h

8.30 ldms_notify_event_s Struct Reference

Notification event type.

`#include <ldms.h>`

Public Types

- enum **ldms_notify_event_type** { **LDMS_SET_MODIFIED** = 1, **LDMS_USER_-DATA** = 2 }

Public Attributes

- enum ldms_notify_event_s::ldms_notify_event_type **type**
- size_t len
- union {
  unsigned char **u_data** [0]
  };

8.30.1 Detailed Description

Notification event type.

The ldms_notify_event_t should be initialized using one of the following functions:

* ldms_init_notify_modified * ldms_init_notify_user_data

8.30.2 Member Data Documentation

8.30.2.1 size_t ldms_notify_event_s::len

Specifies the type of event

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

- src/core/ldms.h

8.31 ldms_rbuf_desc Struct Reference

Public Member Functions

- **LIST_ENTRY** (ldms_rbuf_desc) set_link
- **LIST_ENTRY** (ldms_rbuf_desc) xprt_link

Public Attributes

- struct ldms_xprt * **xprt**
- struct ldms_set * **set**
- uint64_t **remote_set_id**
- uint64_t **local_notify_xid**
- uint64_t **remote_notify_xid**
- uint32_t **notify_flags**
- uint32_t **flags**
- uint64_t **xid**
- uint32_t **xprt_data_len**
- void * **xprt_data**
- void * **lcl_data**

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

- src/core/ldms_xprt.h

8.32 ldms_rdma_xprt Struct Reference

Public Member Functions

- **TAILQ_HEAD** (xprt_credit_list, rdma_context) io_q
- **LIST_ENTRY** (ldms_rdma_xprt) client_link

Public Attributes

- struct ldms_xprt * **xprt**
- int **server**
- enum rdma_conn_status **conn_status**
- struct ibv_comp_channel * **cq_channel**
- struct ibv_cq * **rq_cq**
- struct ibv_cq * **sq_cq**
- struct ibv_pd * **pd**
- struct ibv_qp * **qp**
- sem_t **sem**
- pthread_t **server_thread**
- struct rdma_event_channel * **cm_channel**
- struct rdma_cm_id * **cm_id**
- int **rem_rq_credits**
- int **lcl_rq_credits**
- int **sq_credits**
- pthread_mutex_t **credit_lock**

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

- src/xprt/ldms_rdma_xprt.h

8.33 ldms_reply Struct Reference

Public Attributes

- struct ldms_reply_hdr **hdr**
- union {
  struct ldms_lookup_reply **lookup**
  struct ldms_dir_reply **dir**
  struct ldms_req_notify_reply **req_notify**
  };

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

- src/core/ldms_xprt.h

8.34 ldms_reply_hdr Struct Reference

Public Attributes

- uint64_t **xid**

- uint32_t **cmd**
- uint32_t **len**
- uint32_t **rc**

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

- src/core/ldms_xprt.h

8.35 ldms_req_notify_cmd_param Struct Reference

Public Attributes

- uint64_t **set_id**
- uint32_t flags

8.35.1 Member Data Documentation

8.35.1.1 uint32_t ldms_req_notify_cmd_param::flags

The set we want notifications for

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

- src/core/ldms_xprt.h

8.36 ldms_req_notify_reply Struct Reference

Public Attributes

- struct ldms_notify_event_s **event**

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

- src/core/ldms_xprt.h

8.37 ldms_request Struct Reference

Public Attributes

- struct ldms_request_hdr **hdr**

- union {
  struct ldms_dir_cmd_param **dir**
  struct ldms_lookup_cmd_param **lookup**
  struct ldms_req_notify_cmd_param **req_notify**
  struct ldms_cancel_notify_cmd_param **cancel_notify**
  };

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

- src/core/ldms_xprt.h

8.38 ldms_request_hdr Struct Reference

Public Attributes

- uint64_t **xid**
- uint32_t cmd
- uint32_t len

8.38.1 Member Data Documentation

8.38.1.1 uint32_t ldms_request_hdr::cmd

Transaction id returned in reply

8.38.1.2 uint32_t ldms_request_hdr::len

The operation being requested

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

- src/core/ldms_xprt.h

8.39 ldms_set Struct Reference

Public Member Functions

- **LIST_HEAD** (rbd_list, ldms_rbuf_desc) rbd_list

Public Attributes

- unsigned long **flags**

- struct ldms_set_hdr * **meta**
- struct ldms_data_hdr * **data**
- struct ogc_rbn **rb_node**

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

- src/core/ldms.h

8.40 ldms_set_desc Struct Reference

Public Attributes

- struct ldms_rbuf_desc * **rbd**
- struct ldms_set * **set**

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

- src/core/ldms.h

8.41 ldms_set_hdr Struct Reference

Public Attributes

- uint64_t **meta_gn**
- uint32_t **version**
- uint32_t **flags**
- uint32_t **card**
- uint32_t **meta_size**
- uint32_t **data_size**
- uint64_t **values**
- uint32_t **head_off**
- uint32_t **tail_off**
- char **name** [LDMS_SET_NAME_MAX]

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

- src/core/ldms.h

8.42 ldms_sock_xprt Struct Reference

```
#include <ldms_sock_xprt.h>
```

Public Member Functions

- **LIST_ENTRY** (ldms_sock_xprt) client_link

Public Attributes

- struct ldms_xprt * **xprt**
- enum sock_conn_status **conn_status**
- int **sock**
- struct bufferevent * **buf_event**
- struct evconnlistener * **listen_ev**

8.42.1 Detailed Description

SOCK Transport private data

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

- src/xprt/ldms_sock_xprt.h

8.43 ldms_timestamp Struct Reference

Public Attributes

- uint32_t **sec**
- uint32_t **usec**

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

- src/core/ldms.h

8.44 ldms_transaction Struct Reference

Public Attributes

- struct ldms_timestamp **ts**
- uint32_t **flags**

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

- src/core/ldms.h

8.45 ldms_ugni_xprt Struct Reference

Public Member Functions

- **LIST_ENTRY** (ldms_ugni_xprt) client_link

Public Attributes

- struct ldms_xprt * **xprt**
- enum ugni_conn_status **conn_status**
- int **sock**
- gni_dom_t **dom**
- gni_ep_handle_t **ugni_ep**
- uint32_t **rem_pe_addr**
- uint32_t **rem_inst_id**
- struct bufferevent * **buf_event**
- struct evconnlistener * **listen_ev**

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

- src/xprt/ldms_ugni.h

8.46 ldms_value Union Reference

Metric value union.

`#include <ldms.h>`

Public Attributes

- uint8_t **v_u8**
- int8_t **v_s8**
- uint16_t **v_u16**
- int16_t **v_s16**
- uint32_t **v_u32**
- int32_t **v_s32**
- uint64_t **v_u64**
- int64_t **v_s64**
- float **v_f**
- double **v_d**
- long double **v_ld**

8.46.1 Detailed Description

Metric value union.

A generic union that encapsulates all of the LDMS value types.

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

- src/core/ldms.h

8.47 ldms_value_desc Struct Reference

Metric value descriptor.

`#include <ldms.h>`

Public Attributes

- uint64_t **user_data**
- uint32_t next_offset
- uint32_t data_offset
- uint32_t type
- uint8_t name_len
- char name [0]

8.47.1 Detailed Description

Metric value descriptor.

This structure describes a metric value in the metric set. Metrics are self describing.

8.47.2 Member Data Documentation

8.47.2.1 uint32_t ldms_value_desc::data_offset

Offset of next descriptor

8.47.2.2 char ldms_value_desc::name[0]

The length of the metric name in bytes

8.47.2.3 uint8_t ldms_value_desc::name_len

The type of the value, enum ldms_value_type

8.47.2.4 uint32_t ldms_value_desc::next_offset

User defined meta-data

8.47.2.5 uint32_t ldms_value_desc::type

Offset of the value in ldms_data_hdr

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

- src/core/ldms.h

8.48 ldms_xprt Struct Reference

Public Member Functions

- **LIST_HEAD** (xprt_rbd_list, ldms_rbuf_desc) rbd_list
- **LIST_ENTRY** (ldms_xprt) xprt_link

Public Attributes

- char **name** [LDMS_MAX_TRANSPORT_NAME_LEN]
- int **ref_count**
- struct sockaddr_storage **local_ss**
- struct sockaddr_storage **remote_ss**
- socklen_t **ss_len**
- pthread_mutex_t **lock**
- int **connected**
- int **closed**
- int **max_msg**
- uint64_t **local_dir_xid**
- uint64_t **remote_dir_xid**
- int(∗ connect)(struct ldms_xprt ∗, struct sockaddr ∗sa, socklen_t sa_len)
- int(∗ listen)(struct ldms_xprt ∗, struct sockaddr ∗sa, socklen_t sa_len)
- void(∗ close)(struct ldms_xprt ∗)
- void(∗ destroy)(struct ldms_xprt ∗)
- int(∗ send)(struct ldms_xprt ∗, void ∗, size_t)
- int(∗ read_data_start)(struct ldms_xprt ∗, ldms_set_t, size_t, void ∗)
- int(∗ read_meta_start)(struct ldms_xprt ∗, ldms_set_t, size_t, void ∗)
- int(∗ read_complete_cb)(struct ldms_xprt ∗, void ∗)
- int(∗ recv_cb)(struct ldms_xprt ∗, void ∗)
- ldms_dir_cb_t ∗ dir_cb
- void ∗ **dir_cb_arg**
- struct ldms_rbuf_desc ∗(∗ alloc)(struct ldms_xprt ∗, struct ldms_set ∗s, void ∗xprt_data, size_t xprt_data_len)

- void(∗ free)(struct ldms_xprt ∗, struct ldms_rbuf_desc ∗)
- ldms_log_fn_t log
- void ∗ private

8.48.1 Member Data Documentation

8.48.1.1 struct ldms_rbuf_desc∗(∗ ldms_xprt::alloc)(struct ldms_xprt ∗, struct ldms_set ∗s, void ∗xprt_data, size_t xprt_data_len) [read]

Allocate a remote buffer

8.48.1.2 void(∗ ldms_xprt::close)(struct ldms_xprt ∗)

Close the connection

8.48.1.3 int(∗ ldms_xprt::connect)(struct ldms_xprt ∗, struct sockaddr ∗sa, socklen_t sa_len)

Request a connection with a server

8.48.1.4 void(∗ ldms_xprt::destroy)(struct ldms_xprt ∗)

Destroy the transport instance

8.48.1.5 ldms_dir_cb_t∗ ldms_xprt::dir_cb

User callback invoked when ldms_dir completes

8.48.1.6 void(∗ ldms_xprt::free)(struct ldms_xprt ∗, struct ldms_rbuf_desc ∗)

Free a remote buffer

8.48.1.7 int(∗ ldms_xprt::listen)(struct ldms_xprt ∗, struct sockaddr ∗sa, socklen_t sa_len)

Listen for incoming connection requests

8.48.1.8 ldms_log_fn_t ldms_xprt::log

Transport message logging callback

8.48.1.9 void∗ ldms_xprt::private

Pointer to the transport's private data

8.48.1.10 int(∗ ldms_xprt::read_complete_cb)(struct ldms_xprt ∗, void ∗)

User callback routine invoked when the read completes.

8.48.1.11 int(∗ ldms_xprt::read_data_start)(struct ldms_xprt ∗, ldms_set_t, size_t, void ∗)

Read remote data buffer

8.48.1.12 int(∗ ldms_xprt::read_meta_start)(struct ldms_xprt ∗, ldms_set_t, size_t, void ∗)

Read remote metadata buffer

8.48.1.13 int(∗ ldms_xprt::recv_cb)(struct ldms_xprt ∗, void ∗)

User callback routine called when data arrives on the transport.

8.48.1.14 int(∗ ldms_xprt::send)(struct ldms_xprt ∗, void ∗, size_t)

Send a request/reply

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

- src/core/ldms_xprt.h

8.49 ldmsd_plugin Struct Reference

Public Types

- enum **ldmsd_plugin_type** { **LDMSD_PLUGIN_SAMPLER**, **LDMSD_PLUGIN_STORE** }

Public Attributes

- char **name** [LDMSD_MAX_PLUGIN_NAME_LEN]
- enum ldmsd_plugin::ldmsd_plugin_type **type**
- enum ldmsd_plugin_type(∗ **get_type**)()
- int(∗ **config**)(struct attr_value_list ∗kwl, struct attr_value_list ∗avl)
- void(∗ **term**)(void)
- const char ∗(∗ **usage**)(void)

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

- src/core/ldmsd.h

8.50 ldmsd_sampler Struct Reference

Public Attributes

- struct ldmsd_plugin **base**
- ldms_set_t(* **get_set**)()
- int(* **sample**)(void)

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

- src/core/ldmsd.h

8.51 ldmsd_stat Struct Reference

Public Attributes

- size_t **curr_busy_count**
- size_t **total_busy_count**

8.51.1 Detailed Description

Some statistics for ldmsd.

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

- src/ldmsd/ldmsd.c

8.52 ldmsd_store Struct Reference

ldms_store

`#include <ldmsd.h>`

Public Attributes

- struct ldmsd_plugin **base**
- void * **ucontext**
- ldmsd_store_handle_t(* **get**)(const char *container)
- ldmsd_store_handle_t(* **new**)(struct ldmsd_store *s, const char *comp_type, const char *container, struct ldmsd_store_metric_index_list *metric_list, void *ucontext)
- void(* **destroy**)(ldmsd_store_handle_t sh)
- int(* **flush**)(ldmsd_store_handle_t sh)
- void(* **close**)(ldmsd_store_handle_t sh)
- void *(* **get_context**)(ldmsd_store_handle_t sh)
- int(* **store**)(ldmsd_store_handle_t sh, ldms_set_t set, ldms_mvec_t mvec)

8.52.1 Detailed Description

ldms_store

A `ldms_store` encapsulates a storage strategy. For example, MySQL, or SOS. Each strategy provides a library that exports the `ldms_store` structure. This structure contains strategy routines for initializing, configuring and storing metric set data to the persistent storage used by the strategy. When a metric set is sampled, if metrics in the set are associated with a storage strategy, the sample is saved automatically by `ldmsd`.

An `ldms_store` manages Metric Series. A Metric Series is a named, grouped, and time ordered series of metric samples. A Metric Series is indexed by Component ID, and Time.

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

- src/core/ldmsd.h

8.53 ldmsd_store_metric_index Struct Reference

Public Member Functions

- **LIST_ENTRY** (ldmsd_store_metric_index) entry

Public Attributes

- char * name
- int index

8.53.1 Member Data Documentation

8.53.1.1 int ldmsd_store_metric_index::index

The index

8.53.1.2 char* ldmsd_store_metric_index::name

For configuration

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

- src/core/ldmsd.h

8.54 ldmsd_store_policy Struct Reference

Public Types

- enum { **STORE_POLICY_CONFIGURING** = 0, **STORE_POLICY_READY**, **STORE_-POLICY_WRONG_CONFIG** }

Public Member Functions

- **LIST_ENTRY** (ldmsd_store_policy) link

Public Attributes

- struct hostset_ref_list **hset_ref_list**
- char * container
- char * setname
- int metric_count
- struct ldmsd_store_metric_index_list metric_list
- char * **comp_type**
- struct store_instance * si
- enum ldmsd_store_policy:: { ... } **state**
- pthread_mutex_t **idx_create_lock**

8.54.1 Member Data Documentation

8.54.1.1 char* ldmsd_store_policy::container

This is store policy ID.

8.54.1.2 int ldmsd_store_policy::metric_count

The number of metrics.

8.54.1.3 struct ldmsd_store_metric_index_list ldmsd_store_policy::metric_list

List of the indices.

8.54.1.4 char* ldmsd_store_policy::setname

It is here for configuration.

8.54.1.5 struct store_instance* ldmsd_store_policy::si

Store instance.

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

- src/core/ldmsd.h

8.55 ldmsd_store_policy_ref Struct Reference

Public Member Functions

- **LIST_ENTRY** (ldmsd_store_policy_ref) entry

Public Attributes

- struct ldmsd_store_policy * **lsp**

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

- src/core/ldmsd.h

8.56 ldmsd_store_tuple_s Struct Reference

Public Attributes

- struct timeval **tv**
- uint32_t **comp_id**
- ldms_metric_t **value**

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

- src/core/ldmsd.h

8.57 ls_set Struct Reference

Public Attributes

- char * **name**

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

- src/ldmsd/ldms_ls.c

8.58 lustre_svc_stats Struct Reference

Public Member Functions

- **LIST_ENTRY** (lustre_svc_stats) link

Public Attributes

- char * **path**
- char * **name**
- FILE * **f**
- struct str_map * **key_id_map**
- void * **metrics** [0]

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

- src/sampler/lustre/lustre_sampler.h

8.59 make_dir_arg Struct Reference

Public Attributes

- int **reply_size**
- struct ldms_reply * **reply**
- struct ldms_xprt * **x**
- int **reply_count**
- int **set_count**
- char * **set_list**
- ssize_t **set_list_len**

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

- src/core/ldms_xprt.c

8.60 mysql_metric_store Struct Reference

Public Attributes

- MYSQL * **conn**
- char * **tablename**
- char * **cleansedmetricname**
- pthread_mutex_t **lock**

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

- src/store/store_mysql.c

8.61 mysql_store_instance Struct Reference

Public Attributes

- struct ldmsd_store * **store**
- char * **container**
- char * **comp_type**
- void * **ucontext**
- idx_t **ms_idx**

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

- src/store/store_mysql.c

8.62 mysqlbulk_metric_store Struct Reference

Public Attributes

- MYSQL * **conn**
- char * **tablename**
- char * **cleansedmetricname**
- char * **insertvalues** [NUM_BULK_INSERT]
- int **insertcount**
- pthread_mutex_t **lock**

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

- src/store/store_mysqlbulk.c

8.63 mysqlbulk_store_instance Struct Reference

Public Attributes

- struct ldmsd_store * **store**
- char * **container**
- char * **comp_type**
- void * **ucontext**
- idx_t **ms_idx**

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

- src/store/store_mysqlbulk.c

8.64 obj_list Struct Reference

Public Member Functions

- **LIST_ENTRY** (obj_list) link

Public Attributes

- char * **key**
- uint64_t **obj**

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

- src/sampler/lustre/str_map.h

8.65 ogc_rbn Struct Reference

Public Attributes

- struct ogc_rbn * **left**
- struct ogc_rbn * **right**
- struct ogc_rbn * **parent**
- int **color**
- void * **key**

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

- src/core/ogc_rbt.h

8.66 ogc_rbt Struct Reference

Public Attributes

- struct ogc_rbn * **root**
- ogc_rbn_comparator_t **comparator**

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

- src/core/ogc_rbt.h

8.67 pe_sample Struct Reference

Public Attributes

- uint64_t **value**
- uint64_t **time_running**

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

- src/sampler/perfevent.c

8.68 pevent Struct Reference

Public Attributes

- struct perf_event_attr **attr**
- char * **name**
- int **pid**
- int **cpu**
- int **fd**
- uint64_t **val**
- uint64_t **tstamp**
- ldms_metric_t **metric_value**
- ldms_metric_t **metric_mean**
- ldms_metric_t **metric_variance**
- ldms_metric_t **metric_stddev**
- double **mean**
- double **variance**
- double **card**
- int **warmup**
- struct pe_sample **sample**
- uint64_t **last_time_running**
- uint64_t **last_value**

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

- src/sampler/perfevent.c

8.69 plugin Struct Reference

Public Attributes

- void * **handle**
- char * **name**

- char * **libpath**
- unsigned long **sample_interval_us**
- int **thread_id**
- int **ref_count**
- union {
  struct ldmsd_plugin * **plugin**
  struct ldmsd_sampler * **sampler**
  struct ldmsd_store * **store**
  };
- struct timeval **timeout**
- struct event * **event**
- pthread_mutex_t **lock**

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

- src/ldmsd/ldmsd.c

8.70 rdma_buf_local_data Struct Reference

Public Attributes

- void * **meta**
- size_t **meta_size**
- struct ibv_mr * **meta_mr**
- void * **data**
- size_t **data_size**
- struct ibv_mr * **data_mr**

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

- src/xprt/ldms_rdma_xprt.h

8.71 rdma_buf_remote_data Struct Reference

Public Attributes

- uint64_t **meta_buf**
- uint32_t **meta_rkey**
- uint32_t **meta_size**
- uint64_t **data_buf**
- uint32_t **data_rkey**
- uint32_t **data_size**

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

- src/xprt/ldms_rdma_xprt.h

8.72 rdma_buffer Struct Reference

Public Member Functions

- **LIST_ENTRY** (rdma_buffer) link

Public Attributes

- char * **data**
- size_t **data_len**
- struct ibv_mr * **mr**

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

- src/xprt/ldms_rdma_xprt.h

8.73 rdma_context Struct Reference

Public Member Functions

- **TAILQ_ENTRY** (rdma_context) pending_link

Public Attributes

- void * **usr_context**
- struct ldms_rdma_xprt * **x**
- struct ibv_send_wr **wr**
- struct ibv_sge **sge**
- enum ibv_wc_opcode **op**
- struct rdma_buffer * **rb**

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

- src/xprt/ldms_rdma_xprt.h

8.74 rdma_credit_update_req Struct Reference

Public Attributes

- struct rdma_request_hdr **rdma_hdr**
- struct ldms_request_hdr **hdr**

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

- src/xprt/ldms_rdma_xprt.h

8.75 rdma_request_hdr Struct Reference

Public Attributes

- uint32_t **credits**

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

- src/xprt/ldms_rdma_xprt.h

8.76 set_list_arg Struct Reference

Public Attributes

- char * **set_list**
- ssize_t **set_list_len**
- int **count**

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

- src/core/ldms.c

8.77 sock_buf_local_data Struct Reference

Public Attributes

- void * **meta**
- size_t **meta_size**
- void * **data**
- size_t **data_size**

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

- src/xprt/ldms_sock_xprt.h

8.78 sock_buf_remote_data Struct Reference

Public Attributes

- uint64_t **rbuf**
- uint64_t **lbuf**
- uint32_t **size**

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

- src/xprt/ldms_sock_xprt.h

8.79 sock_buf_xprt_data Struct Reference

Public Attributes

- struct sock_buf_remote_data **meta**
- struct sock_buf_remote_data **data**

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

- src/xprt/ldms_sock_xprt.h

8.80 sock_read_req Struct Reference

Public Attributes

- struct ldms_request_hdr **hdr**
- struct sock_buf_remote_data **buf_info**

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

- src/xprt/ldms_sock_xprt.h

8.81 sock_read_rsp Struct Reference

Public Attributes

- struct ldms_request_hdr **hdr**
- struct sock_buf_remote_data **buf_info**
- uint32_t **status**

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

- src/xprt/ldms_sock_xprt.h

8.82 sos_metric_store Struct Reference

Store for individual metric.

Public Attributes

- sos_t sos
- pthread_mutex_t lock
- char * path

8.82.1 Detailed Description

Store for individual metric.

8.82.2 Member Data Documentation

8.82.2.1 pthread_mutex_t sos_metric_store::lock

lock at metric store level

8.82.2.2 char* sos_metric_store::path

path of the sos store

8.82.2.3 sos_t sos_metric_store::sos

sos handle

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

- src/store/store_sos.c

8.83 sos_store_instance Struct Reference

Public Attributes

- struct ldmsd_store * **store**
- char * path
- char * **container**
- void * **ucontext**
- idx_t **ms_idx**

8.83.1 Member Data Documentation

8.83.1.1 char* sos_store_instance::path

(root_path)/(comp_type)

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

- src/store/store_sos.c

8.84 store_instance Struct Reference

Public Types

- enum { **STORE_STATE_INIT** = 0, **STORE_STATE_OPEN**, **STORE_STATE_CLOSED**, **STORE_STATE_ERROR** }

Public Member Functions

- **TAILQ_ENTRY** (store_instance) lru_entry
- **LIST_ENTRY** (store_instance) flush_entry

Public Attributes

- struct ldmsd_store ∗ store_engine
- ldmsd_store_handle_t store_handle
- struct flush_thread ∗ ft
- enum store_instance:: { ... } **state**
- size_t **dirty_count**
- pthread_mutex_t **lock**
- io_work_fn **work_fn**
- int **work_pending**

8.84.1 Member Data Documentation

8.84.1.1 struct flush_thread∗ store_instance::ft

The pointer to the assigned flush_thread

8.84.1.2 struct ldmsd_store∗ store_instance::store_engine

The store plugin.

8.84.1.3 ldmsd_store_handle_t store_instance::store_handle

The store handle from store->new or store->get

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

- src/core/ldmsd.h

8.85 str_list Struct Reference

Public Member Functions

- **LIST_ENTRY** (str_list) link

Public Attributes

- char * **str**

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

- src/sampler/lustre/lustre_sampler.h

8.86 str_map Struct Reference

str_map definition.

`#include <str_map.h>`

Public Attributes

- size_t hash_size
- struct obj_list_head lh_table [0]

8.86.1 Detailed Description

str_map definition.

str_map is a map that maps strings (keys) to objects (void* or uint64_t).

To create a str_map, call str_map_create(size_t sz). To insert an object into the map, call ::str_map_insert(map, key, obj); To remove an object, call ::str_map_remove(map, key); To get an object, call ::str_map_get(map, key);

8.86.2 Member Data Documentation

8.86.2.1 size_t str_map::hash_size

hash size

8.86.2.2 struct obj_list_head str_map::lh_table[0]

hash table

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

- src/sampler/lustre/str_map.h

8.87 ugni_buf_local_data Struct Reference

Public Attributes

- void * **meta**
- size_t **meta_size**
- gni_mem_handle_t **meta_mh**
- void * **data**
- size_t **data_size**
- gni_mem_handle_t **data_mh**

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

- src/xprt/ldms_ugni.h

8.88 ugni_buf_remote_data Struct Reference

Public Attributes

- uint64_t **meta_buf**
- gni_mem_handle_t **meta_mh**
- uint32_t **meta_size**
- uint64_t **data_buf**
- gni_mem_handle_t **data_mh**
- uint32_t **data_size**

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

- src/xprt/ldms_ugni.h

8.89 ugni_desc Struct Reference

Public Member Functions

- **LIST_ENTRY** (ugni_desc) link

Public Attributes

- gni_post_descriptor_t **post**
- struct ldms_ugni_xprt * **gxp**
- void * **context**

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

- src/xprt/ldms_ugni.h

8.90 ugni_hello_req Struct Reference

Public Attributes

- struct ldms_request_hdr **hdr**
- uint32_t **pe_addr**
- uint32_t **inst_id**

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

- src/xprt/ldms_ugni.h

8.91 ugni_hello_rpl Struct Reference

Public Attributes

- struct ldms_request_hdr **hdr**
- uint32_t **pe_addr**
- uint32_t **inst_id**

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

- src/xprt/ldms_ugni.h

8.92 ugni_mh Struct Reference

Public Member Functions

- **LIST_ENTRY** (ugni_mh) link

Public Attributes

- unsigned long **start**
- unsigned long **end**
- gni_mem_handle_t **mh**
- int **ref_count**

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

- src/xprt/ldms_ugni.h

Chapter 9

File Documentation

9.1 src/sampler/gem_link_perf_util.c File Reference

Utilities for the gem_link_perf sampler also used in ncsa_unified.

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <ctype.h>
#include <time.h>
#include <pthread.h>
#include <limits.h>
#include <rca_lib.h>
#include <rs_id.h>
#include <rs_meshcoord.h>
#include "gem_link_perf_util.h"
#include "ldmsd.h"
#include "ldms.h"
```

Functions

- int **get_my_nid** (void)
- void **get_my_coord** (gemini_coord_t *coord)
- void **set_coord_invalid** (gemini_coord_t *coord)
- int **coord_invalid** (gemini_coord_t *coord)
- int **coord_valid** (gemini_coord_t *coord)
- int **coords_equal** (gemini_coord_t *a, gemini_coord_t *b)
- int tid_to_tcoord (int tid, int *row, int *col)
- int tcoord_to_tid (int row, int col, int *tid)
- int **get_my_pattern** (ldmsd_msg_log_f *msglog_outer, int *pattern, int *zind)
- int **tile_to_linkdir** (ldmsd_msg_log_f *msglog_outer, int my_pattern, int my_z_pattern, char *link_file, gemini_tile_t *tile)
- int str_to_linkdir (char *str)
- int str_to_linktype (char *str)
- double tile_to_bw (ldmsd_msg_log_f *msglog_outer, int tile_type)
- int gem_link_perf_parse_interconnect_file (ldmsd_msg_log_f *msglog_outer, char *filename, gemini_coord_t *neighbor, gemini_tile_t *tile, gemini_coord_t *my_coord, double(*max_link_bw)[], int(*tiles_per_dir)[])
- gpcd_context_t * gem_link_perf_create_context (ldmsd_msg_log_f *msglog_outer)
- gpcd_context_t * nic_perf_create_context (ldmsd_msg_log_f *msglog)

9.1.1 Detailed Description

Utilities for the gem_link_perf sampler also used in ncsa_unified. Link aggregation methodlogy based on Kevin Pedretti's (Sandia National Laboratories) gemini performance counter interface and link aggregation library. It has been augmented with pattern analysis of the interconnect file.

9.1.2 Function Documentation

9.1.2.1 gpcd_context_t* gem_link_perf_create_context (ldmsd_msg_log_f * *msglog_outer*)

Build linked list of tile performance counters we wish to get values for

9.1.2.2 int gem_link_perf_parse_interconnect_file (ldmsd_msg_log_f * *msglog_outer*, char * *filename*, gemini_coord_t * *neighbor*, gemini_tile_t * *tile*, gemini_coord_t * *my_coord*, double(*) *max_link_bw[]*, int(*) *tiles_per_dir[]*)

Parses rtr tool interconnect dump file to determine each gemini's logical links and which gemini tiles are associated with each logical link. This is a collective call. Only rank 0 actually reads interconnect.txt.

Returns

0 on success.
-1 on failure.

9.1.2.3 gpcd_context_t* nic_perf_create_context (ldmsd_msg_log_f * *msglog*)

Build linked list of performance counters we wish to get values for

9.1.2.4 int str_to_linkdir (char * *str*)

Converts the input link direction string (e.g., "X+") into a link direction integer value, representing the direction of the link.

9.1.2.5 int str_to_linktype (char * *str*)

Converts the input link type string (e.g., "mezzanine") into a link type integer value, representing the type of the link.

9.1.2.6 int tcoord_to_tid (int *row,* int *col,* int * *tid*)

Converts a Gemini tile (row, col) coordinate to tile ID.

Returns

0 on success, -1 on failure.

9.1.2.7 int tid_to_tcoord (int *tid,* int * *row,* int * *col*)

Converts a Gemini tile ID to tile (`row`, `col`) coordinate.

Returns

0 on success.
-1 on failure.

9.1.2.8 double tile_to_bw (ldmsd_msg_log_f * *msglog_outer,* int *tile_type*)

Return link bandwidth based on type. This is specified in ldms_gemini.h

9.2 src/sampler/gem_link_perf_util.h File Reference

Utilities for gem_link_perf/nic_perf_sampler also used in ncsa_unified.

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
```

```
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <ctype.h>
#include <rca_lib.h>
#include <rs_id.h>
#include <rs_meshcoord.h>
#include <gpcd_lib.h>
#include "gemini.h"
#include "ldms.h"
#include "ldmsd.h"
```

Defines

- #define **_GNU_SOURCE**
- #define **NETTOPODIM** 3
- #define **NUM_NIC_PERF_RAW** 12
- #define **STR_WRAP**(NAME) #NAME
- #define **PREFIX_ENUM_R**(NAME) R_ ## NAME
- #define **NIC_PERF_RAW_LIST**(WRAP)

Enumerations

- enum **nic_perf_raw_t**

Functions

- int **get_my_nid** (void)
- void **get_my_coord** (gemini_coord_t *coord)
- void **set_coord_invalid** (gemini_coord_t *coord)
- int **coord_invalid** (gemini_coord_t *coord)
- int **coord_valid** (gemini_coord_t *coord)
- int **coords_equal** (gemini_coord_t *a, gemini_coord_t *b)
- int tid_to_tcoord (int tid, int *row, int *col)
- int tcoord_to_tid (int row, int col, int *tid)
- int **str_to_tid** (char *str)
- int str_to_linkdir (char *str)
- int str_to_linktype (char *str)

- double tile_to_bw (ldmsd_msg_log_f *msglog_outer, int tile_type)
- int **get_my_pattern** (ldmsd_msg_log_f *msglog_outer, int *pattern, int *zind)
- int gem_link_perf_parse_interconnect_file (ldmsd_msg_log_f *msglog_outer, char *filename, gemini_coord_t *neighbor, gemini_tile_t *tile, gemini_coord_t *mycoord, double(*max_link_bw)[], int(*tiles_per_dir)[])
- gpcd_context_t * gem_link_perf_create_context (ldmsd_msg_log_f *)
- gpcd_context_t * nic_perf_create_context (ldmsd_msg_log_f *)

9.2.1 Detailed Description

Utilities for gem_link_perf/nic_perf_sampler also used in ncsa_unified.

9.2.2 Define Documentation

9.2.2.1 #define NIC_PERF_RAW_LIST(*WRAP*)

Value:

```
WRAP(GM_ORB_PERF_VC1_STALLED),          \
                WRAP(GM_ORB_PERF_VC0_STALLED),  \
                WRAP(GM_ORB_PERF_VC1_PKTS),     \
                WRAP(GM_ORB_PERF_VC0_PKTS),     \
                WRAP(GM_ORB_PERF_VC1_FLITS),    \
                WRAP(GM_ORB_PERF_VC0_FLITS),    \
                WRAP(GM_NPT_PERF_NPT_FLIT_CNTR),        \
                WRAP(GM_NPT_PERF_NPT_PKT_CNTR),         \
                WRAP(GM_NPT_PERF_NPT_BLOCKED_CNTR),     \
                WRAP(GM_NPT_PERF_NPT_STALLED_CNTR),     \
                WRAP(GM_RAT_PERF_HEADER_FLITS_VC0),     \
                WRAP(GM_RAT_PERF_DATA_FLITS_VC0)
```

9.2.3 Function Documentation

9.2.3.1 gpcd_context_t* gem_link_perf_create_context (ldmsd_msg_log_f * *msglog_outer*)

Build linked list of tile performance counters we wish to get values for

9.2.3.2 int gem_link_perf_parse_interconnect_file (ldmsd_msg_log_f * *msglog_outer*, char * *filename*, gemini_coord_t * *neighbor*, gemini_tile_t * *tile*, gemini_coord_t * *my_coord*, double(*) *max_link_bw[]*, int(*) *tiles_per_dir[]*)

Parses rtr tool interconnect dump file to determine each gemini's logical links and which gemini tiles are associated with each logical link. This is a collective call. Only rank 0 actually reads interconnect.txt.

Returns

0 on success.
-1 on failure.

9.2.3.3 gpcd_context_t* nic_perf_create_context (ldmsd_msg_log_f * *msglog*)

Build linked list of performance counters we wish to get values for

9.2.3.4 int str_to_linkdir (char * *str*)

Converts the input link direction string (e.g., "X+") into a link direction integer value, representing the direction of the link.

9.2.3.5 int str_to_linktype (char * *str*)

Converts the input link type string (e.g., "mezzanine") into a link type integer value, representing the type of the link.

9.2.3.6 int tcoord_to_tid (int *row,* int *col,* int * *tid*)

Converts a Gemini tile (row, col) coordinate to tile ID.

Returns

0 on success, -1 on failure.

9.2.3.7 int tid_to_tcoord (int *tid,* int * *row,* int * *col*)

Converts a Gemini tile ID to tile (`row`, `col`) coordinate.

Returns

0 on success.
-1 on failure.

9.2.3.8 double tile_to_bw (ldmsd_msg_log_f * *msglog_outer,* int *tile_type*)

Return link bandwidth based on type. This is specified in ldms_gemini.h

9.3 src/sampler/geminfo.c File Reference

/proc/geminfo data provider

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
```

```
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <ctype.h>
#include "ldms.h"
#include "ldmsd.h"
```

Defines

- #define **PROC_FILE** "/proc/kgnilnd/stats"

Functions

- char * **replace_space** (char *s)
- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- ldms_set_t **set**
- FILE * **mf**
- ldms_metric_t * **metric_table**
- ldmsd_msg_log_f **msglog**
- uint64_t **comp_id**

9.3.1 Detailed Description

/proc/geminfo data provider

9.4 src/sampler/lustre/fnv_hash.h File Reference

Defines

- #define **FNV_32_PRIME** 0x01000193
- #define **FNV_64_PRIME** 0x100000001b3ULL

9.4.1 Detailed Description

9.5 src/sampler/lustre/lustre_sampler.c File Reference

Lustre sampler common routine implementation.

```
#include <stdlib.h>
#include <dirent.h>
#include <wordexp.h>
#include "lustre_sampler.h"
```

Defines

- #define **__LBUF_SIZ** 256

Functions

- void lustre_sampler_set_msglog (ldmsd_msg_log_f f)
- struct lustre_svc_stats * lustre_svc_stats_alloc (const char *path, int mlen)
- void lustre_svc_stats_free (struct lustre_svc_stats *lss)
- void lustre_svc_stats_list_free (struct lustre_svc_stats_head *h)
- int __add_metric_routine (ldms_set_t set, uint64_t udata, const char *metric_name, struct str_map *id_map, const char *key, struct lustre_svc_stats *lss)
- int lss_open_file (struct lustre_svc_stats *lss)
- int lss_close_file (struct lustre_svc_stats *lss)
- int stats_construct_routine (ldms_set_t set, uint64_t comp_id, const char *stats_path, const char *metric_name_base, struct lustre_svc_stats_head *stats_head, char **keys, int nkeys, struct str_map *key_id_map)
- int lss_sample (struct lustre_svc_stats *lss)

  Sample the metrics in `lss`.
- void **free_str_list** (struct str_list_head *h)
- struct str_list_head * construct_str_list (const char *strlist)
- struct str_list_head * construct_dir_list (const char *path)

9.5.1 Detailed Description

Lustre sampler common routine implementation.

9.5.2 Function Documentation

9.5.2.1 int __add_metric_routine (ldms_set_t *set*, uint64_t *udata*, const char * *metric_name*, struct str_map * *id_map*, const char * *key*, struct lustre_svc_stats * *lss*)

Returns

0 on success.
Error code on error.

9.5.2.2 struct str_list_head* construct_dir_list (const char * *path*) [read]

Construct string list of directories inside the given `path`.

Parameters

path	The path (directory) to query from.

Returns

The list head.

9.5.2.3 struct str_list_head* construct_str_list (const char * *strlist*) [read]

Construct str_list out of comma-separated `strlist`.

Parameters

strlist	The comma-separated string.

Returns

Pointer to ::str_list_head.

9.5.2.4 int lss_close_file (struct lustre_svc_stats * *lss*)

Close the file opened by `lss`.

9.5.2.5 int lss_open_file (struct lustre_svc_stats * *lss*)

Open the file (which can be a pattern) in lss.

Returns

0 on success.
EEXIST if `lss->f` has already been opened.
EINVAL if `lss->path` matches more than one file.
Error code on other error.

9.5.2.6 int lss_sample (struct lustre_svc_stats * *lss*)

Sample the metrics in `lss`.

Parameters

lss	The lustre_svc_stats structure.

9.5.2.7 void lustre_sampler_set_msglog (ldmsd_msg_log_f *f*)

Set message log function to `f`.

Parameters

f	The pointer to logging function.

9.5.2.8 struct lustre_svc_stats* lustre_svc_stats_alloc (const char * *path*, int *mlen*) [read]

lustre_svc_stats allocation.

Parameters

path	The path of the stats file that ties to the structure.
mlen	The number of metrics for lustre_svc_stats structure.

9.5.2.9 void lustre_svc_stats_free (struct lustre_svc_stats * *lss*)

Free the `lss`.

Parameters

lss	The lustre_svc_stats to free.

9.5.2.10 void lustre_svc_stats_list_free (struct lustre_svc_stats_head * *h*)

Free a list of lustre_svc_stats.

Parameters

h	The head of the list.

9.5.2.11 int stats_construct_routine (ldms_set_t *set*, uint64_t *comp_id*, const char * *stats_path*, const char * *metric_name_base*, struct lustre_svc_stats_head * *stats_head*, char ** *keys*, int *nkeys*, struct str_map * *key_id_map*)

Routine for a stats file.

Returns

0 on success.
Error code on error.

9.6 src/sampler/lustre/lustre_sampler.h File Reference

Lustre sampler header file.

```
#include <stdio.h>
#include <sys/queue.h>
#include "str_map.h"
#include "ldms.h"
#include "ldmsd.h"
```

Classes

- struct str_list
- struct lustre_svc_stats

Defines

- #define **__ALEN**(x) (sizeof(x)/sizeof(*x))
- #define **STATS_KEY_LEN** (__ALEN(stats_key))

Functions

- **LIST_HEAD** (str_list_head, str_list)
- void **free_str_list** (struct str_list_head *h)
- **LIST_HEAD** (lustre_svc_stats_head, lustre_svc_stats)
- struct lustre_svc_stats * lustre_svc_stats_alloc (const char *path, int mlen)
- void lustre_svc_stats_free (struct lustre_svc_stats *lss)
- void lustre_svc_stats_list_free (struct lustre_svc_stats_head *h)
- int stats_construct_routine (ldms_set_t set, uint64_t comp_id, const char *stats_path, const char *metric_name_base, struct lustre_svc_stats_head *stats_head, char **keys, int nkeys, struct str_map *key_id_map)
- void lustre_sampler_set_msglog (ldmsd_msg_log_f f)
- int lss_sample (struct lustre_svc_stats *lss)

Sample the metrics in `lss`.

- int lss_open_file (struct lustre_svc_stats *lss)
- int lss_close_file (struct lustre_svc_stats *lss)
- struct str_list_head * construct_str_list (const char *strlist)
- struct str_list_head * construct_dir_list (const char *path)

Variables

- char * **stats_key** []

9.6.1 Detailed Description

Lustre sampler header file. This header file contains contents shared among multiple Lustre samplers, such as possible keys in a regular 'stats' file.

9.6.2 Function Documentation

9.6.2.1 struct str_list_head* construct_dir_list (const char * *path*) [read]

Construct string list of directories inside the given `path`.

Parameters

path	The path (directory) to query from.

Returns

The list head.

9.6.2.2 struct str_list_head* construct_str_list (const char * *strlist*) [read]

Construct str_list out of comma-separated `strlist`.

Parameters

strlist	The comma-separated string.

Returns

Pointer to ::str_list_head.

9.6.2.3 int lss_close_file (struct lustre_svc_stats * *lss*)

Close the file opened by `lss`.

9.6.2.4 int lss_open_file (struct lustre_svc_stats * *lss*)

Open the file (which can be a pattern) in lss.

Returns

0 on success.
EEXIST if `lss->f` has already been opened.
EINVAL if `lss->path` matches more than one file.
Error code on other error.

9.6.2.5 int lss_sample (struct lustre_svc_stats * *lss*)

Sample the metrics in `lss`.

Parameters

lss	The lustre_svc_stats structure.

9.6.2.6 void lustre_sampler_set_msglog (ldmsd_msg_log_f *f*)

Set message log function to `f`.

Parameters

f	The pointer to logging function.

9.6.2.7 struct lustre_svc_stats* lustre_svc_stats_alloc (const char * *path*, int *mlen*) [read]

lustre_svc_stats allocation.

Parameters

path	The path of the stats file that ties to the structure.
mlen	The number of metrics for lustre_svc_stats structure.

9.6.2.8 void lustre_svc_stats_free (struct lustre_svc_stats * *lss*)

Free the `lss`.

Parameters

lss	The lustre_svc_stats to free.

9.6.2.9 void lustre_svc_stats_list_free (struct lustre_svc_stats_head * *h*)

Free a list of lustre_svc_stats.

Parameters

h	The head of the list.

9.6.2.10 int stats_construct_routine (ldms_set_t *set*, uint64_t *comp_id*, const char * *stats_path*, const char * *metric_name_base*, struct lustre_svc_stats_head * *stats_head*, char ** *keys*, int *nkeys*, struct str_map * *key_id_map*)

Routine for a stats file.

Returns

0 on success.
Error code on error.

9.7 src/sampler/lustre/str_map.c File Reference

String-Object mapping utility.

```
#include "str_map.h"
```

Functions

- struct str_map * str_map_create (size_t sz)

  Create a String-Object map.
- void str_map_free (struct str_map *m)
- uint64_t str_map_get (struct str_map *map, const char *key)
- int str_map_insert (struct str_map *map, const char *key, uint64_t obj)
- int str_map_remove (struct str_map *map, const char *key)

  Remove an object (specified by key) from the map.
- int str_map_id_init (struct str_map *map, char **keys, int nkeys, uint64_t start_id)

9.7.1 Detailed Description

String-Object mapping utility.

Author

Narate Taerat <narate@ogc.us> This shall be moved to lib later, to share with other projects.

9.7.2 Function Documentation

9.7.2.1 struct str_map* str_map_create (size_t *sz*) [read]

Create a String-Object map.

Parameters

sz	Hash table size (the unit is number of elements not bytes).

Returns

NULL on error.
Pointer to str_map on success.

9.7.2.2 void str_map_free (struct str_map * *m*)

Free the map.

9.7.2.3 uint64_t str_map_get (struct str_map * *map*, const char * *key*)

Parameters

map	The map.
key	The key.

Returns

NULL if there is no such object.
Pointer to the object.

9.7.2.4 int str_map_id_init (struct str_map * *map*, char ** *keys*, int *nkeys*, uint64_t *start_id*)

Parameters

map	The empty map to be initialized.
keys	The keys
nkeys	The number of keys
start_id	The starting number of the IDs.

9.7.2.5 int str_map_insert (struct str_map * *map*, const char * *key*, uint64_t *obj*)

Returns

0 on success.
Error code on error.

9.7.2.6 int str_map_remove (struct str_map * *map*, const char * *key*)

Remove an object (specified by key) from the map.

Returns

0 on success.
Error code on error.

9.8 src/sampler/lustre/str_map.h File Reference

String-Object mapping utility.

```
#include <sys/mman.h>
#include <sys/queue.h>
#include <inttypes.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include "fnv_hash.h"
```

Classes

- struct obj_list
- struct str_map

  str_map definition.

Functions

- **LIST_HEAD** (obj_list_head, obj_list)
- struct str_map * str_map_create (size_t sz)

  Create a String-Object map.
- void str_map_free (struct str_map *m)
- uint64_t str_map_get (struct str_map *map, const char *key)
- int str_map_insert (struct str_map *map, const char *key, uint64_t obj)
- int str_map_remove (struct str_map *map, const char *key)

  Remove an object (specified by key) from the map.
- int str_map_id_init (struct str_map *map, char **keys, int nkeys, uint64_t start_id)

9.8.1 Detailed Description

String-Object mapping utility.

Author

Narate Taerat <narate@ogc.us> This shall be moved to lib later, to share with other projects.

9.8.2 Function Documentation

9.8.2.1 struct str_map* str_map_create (size_t *sz*) [read]

Create a String-Object map.

Parameters

sz	Hash table size (the unit is number of elements not bytes).

Returns

NULL on error.
Pointer to str_map on success.

9.8.2.2 void str_map_free (struct str_map * *m*)

Free the map.

9.8.2.3 uint64_t str_map_get (struct str_map * *map*, const char * *key*)

Parameters

map	The map.
key	The key.

Returns

NULL if there is no such object.
Pointer to the object.

9.8.2.4 int str_map_id_init (struct str_map * *map*, char ** *keys*, int *nkeys*, uint64_t *start_id*)

Parameters

map	The empty map to be initialized.
keys	The keys
nkeys	The number of keys
start_id	The starting number of the IDs.

9.8.2.5 int str_map_insert (struct str_map * *map*, const char * *key*, uint64_t *obj*)

Returns

0 on success.
Error code on error.

9.8.2.6 int str_map_remove (struct str_map * *map*, const char * *key*)

Remove an object (specified by key) from the map.

Returns

0 on success.
Error code on error.

9.9 src/sampler/meminfo.c File Reference

/proc/meminfo data provider

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include <pthread.h>
#include "ldms.h"
#include "ldmsd.h"
```

Defines

- #define **_GNU_SOURCE**
- #define **PROC_FILE** "/proc/meminfo"

Functions

- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- ldms_set_t **set**
- FILE * **mf**
- ldms_metric_t * **metric_table**
- ldmsd_msg_log_f **msglog**
- uint64_t **comp_id**

9.9.1 Detailed Description

/proc/meminfo data provider

9.10 src/sampler/ncsa_unified.c File Reference

unified custom data provider for ncsa interested metrics. Combo of metrics from other samplers.

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <pthread.h>
#include <sys/types.h>
#include <ctype.h>
#include <time.h>
#include <wordexp.h>
#include "lustre/lustre_sampler.h"
#include "gem_link_perf_util.h"
```

Defines

- #define _GNU_SOURCE
- #define **FALSE** 0
- #define **TRUE** 1
- #define **VMSTAT_FILE** "/proc/vmstat"
- #define **LOADAVG_FILE** "/proc/loadavg"
- #define **CURRENT_FREEMEM_FILE** "/proc/current_freemem"

- #define **KGNILND_FILE** "/proc/kgnilnd/stats"
- #define **LUSTRE_METRICS_LEN** (sizeof(LUSTRE_METRICS)/sizeof(LUSTRE_-METRICS[0]))
- #define **LLITE_PREFIX** "/proc/fs/lustre/llite"
- #define **PREFIX_ENUM_M**(NAME) M_ ## NAME
- #define **NIC_PERF_METRIC_LIST**(WRAP)
- #define **NUM_NIC_PERF_METRICS** 9

Enumerations

- enum **nsca_sources_t** {
  NS_VMSTAT, **NS_LOADAVG**, **NS_CURRENT_FREEMEM**, **NS_KGNILND**,
  NS_LUSTRE, **NS_GEM_LINK_PERF**, **NS_NIC_PERF**, **NS_NUM** }
- enum **nic_perf_metric_t**

Functions

- char ∗ **replace_space** (char ∗s)
- int **get_metric_size_nic_perf** (size_t ∗m_sz, size_t ∗d_sz, int ∗mc)
- int **get_metric_size_gem_link_perf** (size_t ∗m_sz, size_t ∗d_sz, int ∗mc)
- int **get_metric_size_lustre** (size_t ∗m_sz, size_t ∗d_sz, int ∗mc)
- int **gem_link_perf_setup** ()
- int **handle_llite** (const char ∗llite)
- int **sample_metrics_lustre** (int curr_metric_no, int ∗mc)
- uint64_t **__nic_perf_metric_calc** (int metric, uint64_t time_delta)
- int **sample_metrics_nic_perf** (int curr_metric_no, int ∗mc)
- uint64_t **__gem_link_metric_calc** (int i, int j, uint64_t sample_link_ctrs[ ][GEMINI_-NUM_TILE_COUNTERS], uint64_t time_delta)
- int **sample_metrics_gem_link_perf** (int curr_metric_no, int ∗mc)
- struct ldmsd_plugin ∗ **get_plugin** (ldmsd_msg_log_f pf)

Variables

- FILE ∗ **cf_f**
- int **num_current_freemem_metrics** = 1
- ldms_metric_t ∗ **metric_table_current_freemem**
- FILE ∗ **v_f**
- int **num_vmstat_metrics** = 2
- ldms_metric_t ∗ **metric_table_vmstat**
- FILE ∗ **l_f**
- int **num_loadavg_metrics** = 4
- ldms_metric_t ∗ **metric_table_loadavg**
- FILE ∗ **k_f**
- int **num_kgnilnd_metrics** = 8
- ldms_metric_t ∗ **metric_table_kgnilnd**

- struct str_map * lustre_idx_map = NULL
- struct lustre_svc_stats_head **lustre_svc_head** = {0}
- struct str_list_head **llite_str_list** = {0}
- uint64_t **ns_nic_diff** [NUM_NIC_PERF_RAW]
- gpcd_context_t * **ns_nic_curr_ctx**
- gpcd_context_t * **ns_nic_prev_ctx**
- gpcd_context_t * **ns_nic_int_ctx**
- gpcd_mmr_list_t * **ns_nic_listp**
- gpcd_mmr_list_t * **ns_nic_plistp**
- ldms_metric_t **ns_nic_metric_table** [NUM_NIC_PERF_METRICS]
- struct timespec ns_nic_time1 **ns_nic_time2**
- struct timespec * **ns_nic_curr_time**
- struct timespec * **ns_nic_prev_time**
- struct timespec * **ns_nic_int_time**
- int **ns_nic_valid** = 0
- ldms_metric_t **ns_gemlink_meshcoord_metric_handle** [NETTOPODIM]
- gpcd_context_t * **ns_gemlink_curr_ctx**
- gpcd_context_t * **ns_gemlink_prev_ctx**
- gpcd_context_t * **ns_gemlink_int_ctx**
- gpcd_mmr_list_t * **ns_gemlink_listp**
- gpcd_mmr_list_t * **ns_gemlink_plistp**
- ldms_metric_t * **ns_gemlink_metric_table**
- int **num_gem_link_perf_metrics** = 0
- int **ns_gemlink_valid** = 0
- gemini_state_t * **ns_gemlink_state**
- gemini_coord_t **ns_gemlink_my_coord**
- char * **ns_gemlink_rtrfile** = NULL
- int **ns_gemlink_rc_to_tid** [GEMINI_NUM_TILE_ROWS][GEMINI_NUM_TILE_-COLUMNS]
- struct timespec ns_gemlink_time1 **ns_gemlink_time2**
- struct timespec * **ns_gemlink_curr_time**
- struct timespec * **ns_gemlink_prev_time**
- struct timespec * **ns_gemlink_int_time**
- uint64_t **ns_gemlink_diff**
- ldms_set_t **set**
- ldmsd_msg_log_f **msglog**
- uint64_t **comp_id**

9.10.1 Detailed Description

unified custom data provider for ncsa interested metrics. Combo of metrics from other samplers.

9.10.2 Define Documentation

9.10.2.1 #define _GNU_SOURCE

Sub-sampler Notes: GEM_LINK_PERF: Utilizes the aggregation methology of Kevin Pedretti, SNL.

9.10.2.2 #define NIC_PERF_METRIC_LIST(*WRAP*)

Value:

```
WRAP(GM_ORB_PERF_STALLED), \
                WRAP(GM_NPT_PERF_NPT_BLOCKED_CNTR), \
                WRAP(GM_NPT_PERF_NPT_STALLED_CNTR), \
                WRAP(GM_RAT_PERF_HEADER_BYTES_VC0), \
                WRAP(GM_RAT_PERF_DATA_BYTES_VC0), \
                WRAP(GM_ORB_BW), \
                WRAP(GM_NPT_BW), \
                WRAP(GM_ORB_PACKETSIZE_AVE), \
                WRAP(GM_NPT_PACKETSIZE_AVE)
```

9.10.3 Variable Documentation

9.10.3.1 struct str_map* lustre_idx_map = NULL

str<->idx in LUSTRE_METRICS.

9.11 src/sampler/perfevent.c File Reference

perfevent data provider

```
#include <inttypes.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/errno.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <linux/perf_event.h>
#include <math.h>
#include "ldms.h"
#include "ldmsd.h"
```

Classes

- struct pe_sample
- struct pevent

Defines

- #define **ARRAY_SIZE**(a) (sizeof(a) / sizeof(∗a))

Functions

- **LIST_HEAD** (pevent_list, pevent)
- struct ldmsd_plugin ∗ **get_plugin** (ldmsd_msg_log_f pf)

Variables

- ldms_set_t **set**
- ldmsd_msg_log_f **msglog**
- struct kw **add_token_tbl** [ ]
- struct kw **kw_tbl** [ ]

9.11.1 Detailed Description

perfevent data provider Reads perf counters.

9.11.2 Variable Documentation

9.11.2.1 struct kw add_token_tbl[ ]

Initial value:

```
{
        { "cpu", add_event_cpu },
        { "id", add_event_id },
        { "name", add_event_name },
        { "pid", add_event_pid },
        { "type", add_event_type },
}
```

9.11.2.2 struct kw kw_tbl[ ]

Initial value:

```
 {
        { "add", add_event },
        { "del", del_event },
        { "init", init },
        { "ls", list },
}
```

9.12 src/sampler/procinterrupts.c File Reference

/proc/interrupts data provider

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include "ldms.h"
#include "ldmsd.h"
```

Defines

- #define **PROC_FILE** "/proc/interrupts"

Functions

- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- ldms_set_t **set**
- FILE * **mf**
- ldms_metric_t * **metric_table**
- ldmsd_msg_log_f **msglog**
- int **nprocs**
- uint64_t **comp_id**

9.12.1 Detailed Description

/proc/interrupts data provider

9.13 src/sampler/procnetdev.c File Reference

/proc/net/dev data provider

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include "ldms.h"
#include "ldmsd.h"
```

Classes

- struct kw

Defines

- #define **ARRAY_SIZE**(a) (sizeof(a) / sizeof(*a))
- #define **PROC_FILE** "/proc/net/dev"
- #define **NVARS** 16
- #define **MAXIFACE** 5

Functions

- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- int **niface** = 0
- ldms_set_t **set**
- FILE * **mf**

- ldms_metric_t * **metric_table**
- ldmsd_msg_log_f **msglog**
- uint64_t **comp_id**
- struct kw **kw_tbl** []

9.13.1 Detailed Description

/proc/net/dev data provider

9.13.2 Variable Documentation

9.13.2.1 struct kw kw_tbl[ ]

Initial value:

```
 {
        { "add", add_iface },
        { "init", init },
}
```

9.14 src/sampler/procnfs.c File Reference

/proc/net/rpc/nfs data provider

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include "ldms.h"
#include "ldmsd.h"
```

Defines

- #define PROC_FILE "/proc/net/rpc/nfs"
- #define **MAXOPTS** 2
- #define **LINE_FMT**

Functions

- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- ldms_set_t **set**
- FILE * **mf**
- ldms_metric_t * **metric_table**
- ldmsd_msg_log_f **msglog**
- uint64_t **comp_id**

9.14.1 Detailed Description

/proc/net/rpc/nfs data provider

9.14.2 Define Documentation

9.14.2.1 #define LINE_FMT

Value:

```
"%s %s %s %" PRIu64 " %" PRIu64 " %" PRIu64 " %" \
        PRIu64 " %" PRIu64 " %" PRIu64 " %" PRIu64 " %" PRIu64 " %" \
PRIu64 " %" PRIu64 " %" PRIu64 " %" PRIu64 " %" PRIu64 " %" \
PRIu64 " %" PRIu64 " %" PRIu64 " %" PRIu64 " %" PRIu64 " %" \
        PRIu64 " %" PRIu64 " %" PRIu64 " %s\n"
```

9.14.2.2 #define PROC_FILE "/proc/net/rpc/nfs"

File: /proc/net/rpc/nfs

Gets the following selected data items:

Second line: rpc 2 numeric fields field1: Total number of RPC calls to NFS, field2: Number of times a call had to be retransmitted due to a timeout while waiting for a reply from server

Fourth line: proc3 23 numeric fields: field3: getattr field4: setattr field5: lookup field6: access field7: readlink field8: read field9: write field10: create field11: mkdir field12: symlink field13: mknod field14: remove field15: rmdir field16: rename field17: link field18: readdir field19: readdirplus field20: fsstat field21: fsinfo field22: pathconf field23: commit

9.15 src/sampler/procsensors.c File Reference

reads from proc the data that populates lm sensors (in∗_input, fan∗_input, temp∗_input)

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include <pthread.h>
#include "ldms.h"
#include "ldmsd.h"
```

Functions

- struct ldmsd_plugin ∗ **get_plugin** (ldmsd_msg_log_f pf)

Variables

- ldms_set_t **set**
- FILE ∗ **mf**
- ldms_metric_t ∗ **metric_table**
- int **metric_count**
- uint64_t ∗ **metric_values**
- uint64_t ∗ **metric_times**
- int **num_metric_times**
- ldmsd_msg_log_f **msglog**
- uint64_t **comp_id**

9.15.1 Detailed Description

reads from proc the data that populates lm sensors (in∗_input, fan∗_input, temp∗_input) NOTE: data files have to be opened and closed on each file in sys for the data vaules to change. The actual functionality of the data gathering by the system takes time on systems. Sample stores the data locally and then writes it out so that the collection will not occur during partial set. There is therefore some slop in the actaul time for the data point.

filename will be the variable name. mysql inserter will have to convert names and downselect which ones to record.

FIXME: decideif multipliers should go here....

9.16 src/sampler/procstatutil.c File Reference

/proc/stat/util data provider

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include "ldms.h"
#include "ldmsd.h"
```

Functions

- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- ldms_set_t **set**
- FILE * **mf**
- ldms_metric_t * **metric_table**
- ldmsd_msg_log_f **msglog**
- int **numcpu_plusone**
- uint64_t **comp_id**

9.16.1 Detailed Description

/proc/stat/util data provider

9.17 src/sampler/sampler_atasmart.c File Reference

Collect S.M.A.R.T. attribute values.

```
#include <assert.h>
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include <pthread.h>
#include <malloc.h>
#include <atasmart.h>
#include "ldms.h"
#include "ldmsd.h"
```

Classes

- struct ldms_atasmart
- struct atatsmart_set_size

Defines

- #define **_GNU_SOURCE**
- #define **NFIELD** 8

Functions

- int **atasmart_get_disk_info** (SkDisk *d, const SkSmartAttributeParsedData *a, void *userdata)
- int **atasmart_add_metric** (SkDisk *d, const SkSmartAttributeParsedData *a, void *userdata)
- int **atasmart_set_metric** (SkDisk *d, SkSmartAttributeParsedData *a, void *userdata)

- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- struct ldms_atasmart * **smarts**

9.17.1 Detailed Description

Collect S.M.A.R.T. attribute values. The sampler uses `libatasmart` to collect the S.M.A.R.T metrics.

For each attribute, all values are, except the raw value, collected by the sampler. The metric name format is ID_name, where 'ID' is the attribute ID and 'name' is the attribute name. The invalid values are collected as -1.

9.18 src/sampler/sedc.c File Reference

sedc data provider.

```
#include <glib.h>
#include <inttypes.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include "ldms.h"
#include "ldmsd.h"
```

Classes

- struct fset

Defines

- #define **MAXMETRICSPERSET** 100

Functions

- int **createMetricSet** (char *hostname, int compid, char *shortname)

- int **processSEDCData** (char *line)
- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- GHashTable * **compidmap**
- GHashTable * **setmap**
- char * **dirnamex** = NULL
- char * **filebasename** = NULL
- char **currdate** [20] = ""
- char * **setshortname** = NULL
- char * **filetype** = NULL
- char * **logfile** = NULL
- int **lastpos** = 0
- char **sedcfname** [LDMSD_MAX_CONFIG_STR_LEN] = ""
- FILE * **sedcf** = NULL
- FILE * **mf** = NULL
- ldms_metric_t * **metric_table**
- ldmsd_msg_log_f **msglog**
- int **minindex** = 2

9.18.1 Detailed Description

sedc data provider. Reads the sedc data from a file (to be gotten via rsyslog) and writes to ldms metric sets. Notes:

- Currently reads the headers from a separate file
- Currently reads the compids from a separate file (when these items are inserted via the mysql insert, we will want a nice way to do remote assoc)
- Metric sets are currently added with all metric names, whether or not there is data for them.
- Metric sets are only added when they need to be (that is when a new component appears in the file) -- this is going to be a problem for the mysql inserter
- Still have debugging statements, fixed size arrays.

9.19 src/sampler/sysclassib.c File Reference

reads from 1) all files in: /sys/class/infiniband/mlx4_{0/1}/ports/{1/2}/counters which have well-known names 2) /sys/class/infiniband/mlx4_{0/1}/ports/{1,2}/rate

```
#include <inttypes.h>
#include <unistd.h>
```

```
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <pthread.h>
#include "ldms.h"
#include "ldmsd.h"
```

Classes

- struct kw

Defines

- #define **_GNU_SOURCE**
- #define **MAXIFACE** 4
- #define **V1** 2
- #define **V2** 6
- #define **V3** 34
- #define **ARRAY_SIZE**(a) (sizeof(a) / sizeof(*a))

Functions

- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- ldms_set_t **set**
- FILE * **mf**
- ldms_metric_t * **metric_table**
- ldmsd_msg_log_f **msglog**
- ldms_metric_t **compid_metric_handle**
- ldms_metric_t **counter_metric_handle**
- union ldms_value **comp_id**
- int **newerkernel**
- struct kw **kw_tbl** []

9.19.1 Detailed Description

reads from 1) all files in: /sys/class/infiniband/mlx4_{0/1}/ports/{1/2}/counters which have well-known names 2) /sys/class/infiniband/mlx4_{0/1}/ports/{1,2}/rate in config, you can specify ib0 --> mlx4_0 and port1 ib1 --> mlx4_0 and port2 ib2 --> mlx4_1 and port1 ib3 --> mlx4_1 and port2

for older kernels, the filehandles have to be opened and closed each time; for newer kernels (>= 2.6.35) they do not.

FIXME: verify that if you unload & load sampler that filehandles close and reopen properly

9.19.2 Variable Documentation

9.19.2.1 struct kw kw_tbl[ ]

Initial value:

```
 {
  { "add", add_iface },
  { "init", init },
}
```

9.20 src/sampler/vmstat.c File Reference

/proc/vmstat data provider

```
#include <inttypes.h>
#include <unistd.h>
#include <sys/errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include "ldms.h"
#include "ldmsd.h"
```

Defines

- #define **_GNU_SOURCE**
- #define **PROC_FILE** "/proc/vmstat"

Functions

- struct ldmsd_plugin * **get_plugin** (ldmsd_msg_log_f pf)

Variables

- ldms_set_t **set**
- FILE * **mf**
- ldms_metric_t * **metric_table**
- ldmsd_msg_log_f **msglog**
- uint64_t **comp_id**

9.20.1 Detailed Description

/proc/vmstat data provider

Index