DTS

v1.0

Generated by Doxygen 1.5.9

Tue Jul 14 16:59:01 2009

Contents

Chapter 1

File Index

1.1 File List

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

Chapter 2

File Documentation

2.1 dtsASync.c File Reference

DTS asynchronous command methods.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include "dts.h"
```

Functions

- int dts_nullResponse (void *data)
  Asyncronous null handler.

2.1.1 Detailed Description

DTS asynchronous command methods.

DTSASYNC.C

Methods used in Asynchronous commands.

Author:

Mike Fitzpatrick

Date:

6/10/09

2.1.2 Function Documentation

2.1.2.1 int dts_nullResponse (void * *data*)

Asyncronous null handler.

DTS_NULLRESPONSE – Asyncronous null handler.

Parameters:

data xml caller data

Returns:

status code

Referenced by dtsLog().

2.2 dtsClient.c File Reference

DTS Client-side methods.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include <xmlrpc-c/base.h>
#include <xmlrpc-c/client.h>
#include <xmlrpc-c/server.h>
#include <xmlrpc-c/server_abyss.h>
#include "xrpc.h"
#include "dts.h"
```

2.2.1 Detailed Description

DTS Client-side methods.

DTSCLIENT.C

Client-side methods.

Author:

Mike Fitzpatrick

Date:

6/10/09

2.3 dtsConfig.c File Reference

DTS Config File Interface.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include "dts.h"
```

Defines

- #define **DEBUG** 0
- #define **CON_GLOBAL** 0
- #define **CON_DTS** 1
- #define **CON_QUEUE** 2
- #define **CON_XFER** 3

Functions

- char * dtsGets (char *s, int len, FILE *fd)
- void dts_loadConfig (DTS *dts)

  Load the DTS configuration file.

- int dts_cfgQType (char *s)

  Get the type of queue.

- int dts_cfgQMethod (char *s)

  Get the queue transfer method.

- int dts_cfgQMode (char *s)

  Get the trander mode (push or pull).

- char * dts_cfgQTypeStr (int type)

  Convert mode to string name.

- char * dts_cfgQMethodStr (int method)

  Convert method type to string name.

- char * dts_cfgQModeStr (int mode)

  Convert mode type to string name.

Variables

- DTS * **dts**
- int **dts_monitor**
- int **context** = CON_GLOBAL

2.3.1 Detailed Description

DTS Config File Interface.

DTSCONFIG.C – DTS Config File Interface

Author:

Mike FItzpatrick

Date:

6/15/09

2.3.2 Function Documentation

2.3.2.1 int dts_cfgQMethod (char * *s*)

Get the queue transfer method.

DTS_CFGQMETHOD – Get the queue transfer method.

Parameters:

s string method name

Returns:

method code

Referenced by dts_loadConfig().

2.3.2.2 char* dts_cfgQMethodStr (int *method*)

Convert method type to string name.

DTS_CFGQMETHODSTR – Convert method type to string name.

Parameters:

method method type code

Returns:

nothing

2.3.2.3 int dts_cfgQMode (char * *s*)

Get the trander mode (push or pull).

DTS_CFGQMODE – Get the trander mode (push or pull).

Parameters:

s string mode name

Returns:

nothing

Referenced by dts_loadConfig().

2.3.2.4 char* dts_cfgQModeStr (int *mode*)

Convert mode type to string name.

DTS_CFGMODESTR – Convert mode type to string name

Parameters:

mode mode type code

Returns:

nothing

2.3.2.5 int dts_cfgQType (char * *s*)

Get the type of queue.

DTS_CFGQTYPE – Get the type of queue.

Parameters:

s string name of queue

Returns:

queue type

Referenced by dts_loadConfig().

2.3.2.6 char* dts_cfgQTypeStr (int *type*)

Convert mode to string name.

DTS_CFGQTYPESTR – Convert mode to string name

Parameters:

type queue type code

Returns:

nothing

2.3.2.7 void dts_loadConfig (DTS * *dts*)

Load the DTS configuration file.

DTS_LOADCONFIG – Load the DTS configuration file.

Parameters:

dts DTS struct pointer

Returns:

nothing

References dts_cfgQMethod(), dts_cfgQMode(), dts_cfgQType(), and dtsGets().

2.3.2.8 char* dtsGets (char * *s*, int *len*, FILE * *fp*)

DTSGETS A smart fgets() function

Read the line; unlike fgets(), we read the entire line but dump characters that go past the end of the buffer. We accept CR, LF, or CR LF for the line endings to be "safe".

Parameters:

s string buffer

len length of buffer

fp file descriptor

Returns:

pointer to next string on the stream

Referenced by dts_loadConfig().

2.4 dtsConsole.c File Reference

DTS Console Interface.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include "dts.h"
```

Defines

- #define **CON_PASSWD** "xyzzy"

Functions

- int dts_monAttach (void *data)
- int dts_monDetach (void *data)
- int dts_monConsole (void *data)

Variables

- DTS * **dts**
- int **dts_monitor**

2.4.1 Detailed Description

DTS Console Interface.

DTSCONSOLE.C

DTS Console Interface - These are the methods implemented for he XML-RPC interface. These command are used from a variety of locations during the operation of the DTS (i.e. the queueing, logging or interactive environments.

monAttach - attach logging to the specified dtsmonitor

Author:

Mike FItzpatrick

Date:

6/15/09

2.4.2 Function Documentation

2.4.2.1 int dts_monAttach (void * *data*)

DTS_MONATTACH – Attach the DTS logging output to the dtsmon at the specified host.

Parameters:

data calling parameter data #

Returns:

status (OK|ERR)

References dtsLog().

2.4.2.2 int dts_monConsole (void * *data*)

DTS_MONCONSOLE – Request connection as a controlling console. We provide a (minimal security) password that must be supplied to grant access.

Parameters:

data calling parameter data #

Returns:

status (OK|ERR)

References dtsLog().

2.4.2.3 int dts_monDetach (void * *data*)

DTS_MODETACH – Detach the DTS logging output to the current dtsmon.

Parameters:

data calling parameter data #

Returns:

status (OK|ERR)

References dtsLog().

2.5 dtsFileUtil.c File Reference

DTS file utilities.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <sys/errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <fcntl.h>
#include "dts.h"
```

Functions

- int dts_fileOpen (char *fname, int flags)

  Open the named file and return the file descriptor.

- void dts_fileClose (int fd)

  Close the file descriptor.

- int dts_preAlloc (char *fname, long fsize)

  Pre-allocate the space for a file.

- long dts_fileSize (int fd)

  Return the size (in bytes) of the specified file descriptor.

- long **dts_nameSize** (char *fname)
- int dts_getBuffer (int fd, unsigned char **buf, long chunkSize, int tnum)

  Get a buffer of the specified size from the file.

- int **dts_fileRead** (int fd, void *vptr, int nbytes)
- int **dts_fileWrite** (int fd, void *vptr, int nbytes)

2.5.1 Detailed Description

DTS file utilities.

Author:

Mike Fitzpatrick

Date:

6/10/09

2.5.2 Function Documentation

2.5.2.1 void dts_fileClose (int *fd*)

Close the file descriptor.

DTS_FILECLOSE – Close the file descriptor.

Parameters:

fname file descriptor to close

Returns:

nothing

Referenced by psReceiveFile(), and psSendFile().

2.5.2.2 int dts_fileOpen (char * *fname*, int *flags*)

Open the named file and return the file descriptor.

DTS_FILEOPEN – Open the named file and return the file descriptor.

Parameters:

fname file name to open

flags file open flags

Returns:

file descriptor

Referenced by psReceiveFile(), and psSendFile().

2.5.2.3 long dts_fileSize (int *fd*)

Return the size (in bytes) of the specified file descriptor.

DTS_FILESIZE - Return the size (in bytes) of the specified file descriptor.

Parameters:

fd file descriptor

Returns:

file size

2.5.2.4 int dts_getBuffer (int *fd*, unsigned char ** *buf*, long *chunkSize*, int *tnum*)

Get a buffer of the specified size from the file.

DTS_GETBUFFER – Get a buffer of the specified size from the file. Return the number of characters read.

Parameters:

fd file descriptor
buf buffer to create
chunkSize size of transfer chunk (bytes)
tnum thread number

Returns:

file size

2.5.2.5 int dts_preAlloc (char * *fname*, long *fsize*)

Pre-allocate the space for a file.

DTS_PREALLOC – Pre-allocate the space for a file.

Parameters:

fname file name to create
fsize size of the file

Returns:

status code

Referenced by psReceiveFile().

2.6 dtsLog.c File Reference

DTS logging interface.

```
#include <stdio.h>
#include <fcntl.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include "dts.h"
```

Defines

- #define **SZ_FMTSPEC** 25
- #define **EOS** 0
- #define **TY_INT** 0
- #define **TY_DOUBLE** 1
- #define **TY_CHAR** 2

Functions

- void dtsLog (DTS *dts, char *format,...)
  DTS message logger.
- void dts_encodeString (char *buf, char *format, va_list *argp)

2.6.1 Detailed Description

DTS logging interface.

Author:

Mike Fitzpatrick

Date:

6/10/09

We can log to a local file as well as to a remote monitoring application.

2.6.2 Function Documentation

2.6.2.1 void dts_encodeString (char * *buf*, char * *format*, va_list * *argp*)

DTS_ENCODESTRING – Process the format to the output file, taking arguments from the list pointed to by argp as % format specs are encountered in the input.

Parameters:

buf formatted output buffer

format format string

argp variable-length arguments

Returns:

Referenced by dtsLog().

2.6.2.2 void dtsLog (DTS * *dts*, char * *format*, ...)

DTS message logger.

DTS_LOGMSG – DTS message logger.

Parameters:

dts DTS struct pointer

format message format string

Returns:

nothing

References dts_encodeString(), and dts_nullResponse().

Referenced by dts_monAttach(), dts_monConsole(), dts_monDetach(), and dts_Set().

2.7 dtsMethods.c File Reference

DTS Command Interface.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include <xmlrpc-c/base.h>
#include <xmlrpc-c/client.h>
#include <xmlrpc-c/server.h>
#include <xmlrpc-c/server_abyss.h>
#include "xrpc.h"
#include "dts.h"
```

Functions

- int dts_initDTS (void *data)

  Initialize the DTS.

- int dts_shutdownDTS (void *data)

  Shut down the DTS.

- int dts_Echo (void *data)

  Echo the input text.

- int dts_Ping (void *data)

  Is host and DTS alive.

- int dts_diskSpace (void *data)

  Get disk space available on specified path.

- int dts_startQueue (void *data)

  Start the named DTS queue.

- int dts_stopQueue (void *data)

Stop processing of the named queue.

- int dts_flushQueue (void *data)

  Flush the named DTS queue.

- int dts_restartQueue (void *data)

  Simply stop, then restart, the queue.

- int dts_addToQueue (void *data)

  Add the named object to the DTS queue.

- int dts_removeFromQueue (void *data)

  Remove the named object from the DTS queue.

- int dts_List (void *data)

  List configuration parameters.

- int dts_Set (void *data)

  Set and option value.

- int dts_Get (void *data)

  Get and option value.

- int **dts_initTransfer** (void *data)
- int **dts_doTransfer** (void *data)
- int **dts_endTransfer** (void *data)
- int dts_cancelTransfer (void *data)

  Cancel an active file transfer.

Variables

- DTS * **dts**

2.7.1 Detailed Description

DTS Command Interface.

DTS Command Interface - These are the methods implemented for he XML-RPC interace. These command are used from a variety of locations during the operation of the DTS (i.e. the queueing, logging or interactive environments.

initDTS - reinitialize DTS shutdownDTS - shutdown DTS, cancel all queues

echo - simple echo function ping - simple aliveness test function diskSpace - how much disk space is available?

startQueue - start the named queue stopQueue - stop the named queue flushQueue - flush the named queue restartQueue - restart (stop-then-start) queue addToQueue - add object to queue removeFromQueue - delete object from queue

list <option> - list config parameters set <option>

- set a specified option get <option> - get value of option

initTransfer <method> - initialize an object transfer doTransfer - transfer the actual file endTransfer - end the object transfer cancelTransfer - abort an in-progress transfer

Author:

Mike Fitzpatrick

Date:

6/10/09

2.7.2 Function Documentation

2.7.2.1 int dts_addToQueue (void * *data*)

Add the named object to the DTS queue.

DTS_ADDTOTQUEUE – Add the named object to the DTS queue.

Parameters:

data caller param data

Returns:

status code

2.7.2.2 int dts_cancelTransfer (void * *data*)

Cancel an active file transfer.

DTS_CANCELTRANSFER – Cancel an active file transfer.

Parameters:

data caller param data

Returns:

status code

2.7.2.3 int dts_diskSpace (void * *data*)

Get disk space available on specified path.

DTS_DISKSPACE – Get disk space available on specified path.

Parameters:

data caller param data

Returns:

status code

2.7.2.4 int dts_Echo (void * *data*)

Echo the input text.

DTS_ECHO – Simple echo function.

Parameters:

data caller param data

Returns:

status code

2.7.2.5 int dts_flushQueue (void * *data*)

Flush the named DTS queue.

DTS_FLUSHQUEUE – Flush the named DTS queue.

Parameters:

data caller param data

Returns:

status code

2.7.2.6 int dts_Get (void * *data*)

Get and option value.

Clean up and terminate a transfer operation.

Do the actual transfer of an object.

Initialize a transfer operation.

DTS_GET - Get an option value.

Parameters:

data caller param data

Returns:

status code

DTS_INITTRANSFER – Initialize a transfer operation.

Parameters:

data caller param data

Returns:

status code

DTS_DOTRANSFER – Do the actual transfer of an object.

Parameters:

data caller param data

Returns:

status code

DTS_ENDTRANSFER – Clean up and terminate a transfer operation.

Parameters:

data caller param data

Returns:

status code

2.7.2.7 int dts_initDTS (void * *data*)

Initialize the DTS.

DTS_INITDTS – (Re-)Initialize the DTS.

Parameters:

data caller param data

Returns:

status code

2.7.2.8 int dts_List (void * *data*)

List configuration parameters.

DTS_LIST – List config parameters.

Parameters:

data caller param data

Returns:

status code

2.7.2.9 int dts_Ping (void * *data*)

Is host and DTS alive.

DTS_PING – Simple aliveness test function.

Parameters:

data caller param data

Returns:

status code

2.7.2.10 int dts_removeFromQueue (void * *data*)

Remove the named object from the DTS queue.

DTS_REMOVEFROMQUEUE – Remove the named object from the DTS queue.

Parameters:

data caller param data

Returns:

status code

2.7.2.11 int dts_restartQueue (void * *data*)

Simply stop, then restart, the queue.

DTS_RESTARTQUEUE – Simply stop, then restart, the queue. We pass through the calling params and don't need to interpret them here.

Parameters:

data caller param data

Returns:

status code

References dts_startQueue(), and dts_stopQueue().

2.7.2.12 int dts_Set (void * *data*)

Set and option value.

DTS_SET - Set an option value.

Parameters:

data caller param data

Returns:

status code

References dtsLog().

2.7.2.13 int dts_shutdownDTS (void * *data*)

Shut down the DTS.

DTS_SHUTDOWNDTS – Shut down the DTS daemon.

Parameters:

data caller param data

Returns:

status code

2.7.2.14 int dts_startQueue (void * *data*)

Start the named DTS queue.

DTS_STARTQUEUE – Start the named DTS queue

Parameters:

data caller param data

Returns:

status code

Referenced by dts_restartQueue().

2.7.2.15 int dts_stopQueue (void * *data*)

Stop processing of the named queue.

DTS_STOPQUEUE – Stop processing of the named queue.

Parameters:

data caller param data

Returns:

status code

Referenced by dts_restartQueue().

2.8 dtsPSock.c File Reference

DTS parallel socket transfer routines.

```
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <fcntl.h>
#include <sys/errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include "dts.h"
#include "dtsPSock.h"
```

Enumerations

- enum checksumPolicies { **CS_NONE**, **CS_CHUNK**, **CS_STRIPE** }

  Different methods for checksumming during transfer.

Functions

- int psSpawnWorker (void *worker, int nthreads, char *fname, long fsize, int mode, int port, char *host, int verbose)

  Spawn a worker thread for the transfer.

- int **psSpawnThreads** (void *worker, int nthreads, char *fname, long fsize, int mode, int port, char *host, int verbose, pthread_t *tids)

- int **psCollectThreads** (int nthreads, pthread_t *tids)
- void psSendFile (void *data)

  Send a file to a remote DTS.

- void psReceiveFile (void *data)

  Read a file from a remote DTS.

- int psSendStripe (int sock, unsigned char *dbuf, long offset, int tnum, long maxbytes)

  Do actual transfer of data stripe to the socket.

- unsigned char * psReceiveStripe (int sock, long offset, int tnum)

  Read data stripe from the socket connection.

- void psComputeStripe (long fsize, int nthreads, int tnum, long *chsize, long *start, long *end)

  Compute the various parameters of a 'stripe'.

Variables

- pthread_mutex_t svc_mutex = PTHREAD_MUTEX_INITIALIZER
- int **psock_checksum_policy** = CS_NONE
- int **err_return** = -1
- struct timeval **io_tv**

2.8.1 Detailed Description

DTS parallel socket transfer routines.

Author:

Mike Fitzpatrick

Date:

6/10/09

2.8.2 Function Documentation

2.8.2.1 psComputeStripe (long *fsize*, int *nthreads*, int *tnum*, long * *chsize*, long * *start*, long * *end*)

Compute the various parameters of a 'stripe'.

PSCOMPUTESTRIPE – Compute the parameters of a data stripe given the file size and number of worker threads.

Parameters:

fsize file size
nthreads numbers of threads being processed
tnum this thread number
chsize chunk size
start starting byte number of stripe (output)
end ending byte number of stripe (output)

Returns:

nothing

Referenced by psSpawnWorker().

2.8.2.2 psReceiveFile (void * *data*)

Read a file from a remote DTS.

psReceiveFile – Receive a file from a remote DTS.

This function can be called to read a portion of a file from a remote host. Arguments are passed in through a generic 'data' pointer to the psArg struct defined for this 'stripe' of the data.

In this procedure, we act as a client in the connection, i.e. the transfer won't begin until we connect to a remote server sending the data.

Parameters:

data caller thread data

Returns:

nothing

References dts_fileClose(), dts_fileOpen(), dts_openClientSocket(), dts_openServerSocket(), dts_preAlloc(), psReceiveStripe(), and svc_mutex.

Referenced by dts_xferReceiveFile().

2.8.2.3 uchar * psReceiveStripe (int *sock*, long *offset*, int *tnum*)

Read data stripe from the socket connection.

psReceiveStripe – Do the actual transfer of the data stripe to the client connection. A 'stripe' of data is actually transferred in much smaller 'chunks' which can be tuned to be optimal for the given connection. The checksum policy allows us to perform a checksum of the data either for the entire stripe, or for each individual chunk. The former is generally more efficient as it involves fewer round-trips to the client (i.e. send the checksum and wait for verification before sending next chunk). File-level checksum policy is enforced by our parent.

Parameters:

sock socket descriptor

offset file offset for this stripe

tnum thread (i.e. stripe) number

Returns:

a pointer to the data read

References addcheck32().

Referenced by psReceiveFile().

2.8.2.4 psSendFile (void * *data*)

Send a file to a remote DTS.

psSendFile – Send a file to a remote DTS.

This function can be called to send a portion of a file to a remote host. Arguments are passed in through a generic 'data' pointer to the psArg struct defined for this 'stripe' of the data.

In this procedure, we act as a server, i.e. we open the specified tcp/ip socket and wait for a client connection before beginning any transfer.

Parameters:

data caller thread data

Returns:

nothing

References dts_fileClose(), dts_fileOpen(), dts_openClientSocket(), dts_openServerSocket(), psSendStripe(), and svc_mutex.

Referenced by dts_xferPush(), and dts_xferSendFile().

2.8.2.5 int psSendStripe (int *sock*, unsigned char * *dbuf*, long *offset*, int *tnum*, long *maxbytes*)

Do actual transfer of data stripe to the socket.

psSendStripe – Do the actual transfer of the data stripe to the client connection. A 'stripe' of data is actually transferred in much smaller 'chunks' which can be tuned to be optimal for the given connection. The checksum policy allows us to perform a checksum of the data either for the entire stripe, or for each individual chunk. The former is generally more efficient as it involves fewer round-trips to the client (i.e. send the checksum and wait for verification before sending next chunk). File-level checksum policy is enforced by our parent.

Parameters:

sock socket descriptor

dbuf data buffer

offset file offset for this stripe

tnum thread number

maxbytes max bytes to transfer

Returns:

number of chunks sent

References addcheck32().

Referenced by psSendFile().

2.8.2.6 int psSpawnWorker (void * *worker*, int *nthreads*, char * *fname*, long *fsize*, int *mode*, int *port*, char * *host*, int *verbose*)

Spawn a worker thread for the transfer.

PSSPAWNWORKER – Spawn a worker thread for the transfer. All we do here is start a thread to run the function passed in. This may be used to either read or write the data.

Parameters:

worker worker function

nthreads number of threads to create

fname file name

fsize file size

mode transfer mode (push or pull)

port client base port number

host client host name

verbose verbose output flag

Returns:

status code

References measure_start(), measure_stop(), and psComputeStripe().

2.8.3 Variable Documentation

2.8.3.1 pthread_mutex_t svc_mutex = PTHREAD_MUTEX_INITIALIZER

Mutex lock for thread startup to protect file I/O.

Referenced by psReceiveFile(), and psSendFile().

2.9 dtsPull.c File Reference

DTS Pull-Model Transfer Methods.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include <pthread.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <xmlrpc-c/base.h>
#include <xmlrpc-c/client.h>
#include <xmlrpc-c/server.h>
#include <xmlrpc-c/server_abyss.h>
#include "xrpc.h"
#include "dts.h"
#include "dtsPSock.h"
```

Functions

- int **dts_xferPull** (void *data)
- int dts_xferSendFile (void *data)

  Push a file from the src to the dest machine.

Variables

- DTS * **dts**

2.9.1 Detailed Description

DTS Pull-Model Transfer Methods.

DTS Pull-Model Transfer Methods

The methods defined here are both the client and server-side code needed to implement the "push" model of data transfer. In this mode, an RPC command is called to push data from the source to the destination machine. The source machine acts as a server, opening the transfer sockets locally before sending a request to the remote machine to receive the file. Once the connection is established, transfer begins. The return status of this method then determines whether the transfer was successful.

A sequence diagram of the process would look something like:

Push Model:

QMgr Src Dest

(1) —-[Pull Msg] ———-> (2) Socket setup (3) —-[FileRecieve Msg]—-> (4) <=====[Data Transfer]=====> (5) <—-[Status Return]——-

The 'QMgr' is the Queue Manager which initiates the request on the local machine.

RPC Methods Implemented:

dts_xferPull initiate Pull transfer of file (dest method) dts_xferSendFile begin sending file (src method)

Author:

Mike Fitzpatrick

Date:

6/10/09

2.9.2 Function Documentation

2.9.2.1 int dts_xferSendFile (void * *data*)

Push a file from the src to the dest machine.

DTS_XFERSENDFILE – Send a file to the dest machine.

RPC Params: xferId S transfer id fileName S file name fileSize I file size numThreads I num of transfer threads to open srcPort I starting port number srcIP S IP address of caller (string)

RPC Return: 0 transfer succeeded 1 transfer failed ———————————————————

Receive a file on the dest machine.

Parameters:

data caller param data

Returns:

status code

References psSendFile().

2.10 dtsPush.c File Reference

DTS Push-Model Transfer Methods.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include <pthread.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <xmlrpc-c/base.h>
#include <xmlrpc-c/client.h>
#include <xmlrpc-c/server.h>
#include <xmlrpc-c/server_abyss.h>
#include "xrpc.h"
#include "dts.h"
#include "dtsPSock.h"
```

Functions

- int dts_xferPush (void *data)
  Push a file from the src to the dest machine.
- int dts_xferReceiveFile (void *data)
  Push a file from the src to the dest machine.

Variables

- DTS * **dts**

2.10.1 Detailed Description

DTS Push-Model Transfer Methods.

DTS Push-Model Transfer Methods

The methods defined here are both the client and server-side code needed to implement the "push" model of data transfer. In this mode, an RPC command is called to push data from the source to the destination machine. The source machine acts as a server, opening the transfer sockets locally before sending a request to the remote machine to receive the file. Once the connection is established, transfer begins. The return status of this method then determines whether the transfer was successful.

A sequence diagram of the process would look something like:

Push Model:

QMgr Src Dest

(1) —-[Push Msg] ———-> (2) Socket setup (3) —-[FileRecieve Msg]—-> (4) <=====[Data Transfer]=====> (5) <—-[Status Return]——-

The 'QMgr' is the Queue Manager which initiates the request on the local machine.

RPC Methods Implemented:

dts_xferPush initiate Push transfer of file (src method) dts_xferReceiveFile begin receiving file (dest method)

Author:

Mike Fitzpatrick

Date:

6/10/09

2.10.2 Function Documentation

2.10.2.1 int dts_xferPush (void * *data*)

Push a file from the src to the dest machine.

DTS_XFERPUSH – Push a file from the src to the dest machine.

RPC Params: xferId S transfer id method S transfer method fileName S file name fileSize I file size numThreads I num of transfer threads to open srcPort I starting port number destHost S FQDN of destination machine destCmdURL S destination xmlrpc URI

RPC Returns: tsec I transfer time seconds tusec I transfer time micro-seconds status I

transfer succeeded (0) or failed (1) ————————————————————————

Parameters:

data caller param data

Returns:

status code

References psSendFile(), transferMB(), and transferMb().

2.10.2.2 int dts_xferReceiveFile (void * *data*)

Push a file from the src to the dest machine.

DTS_XFERRECEIVEFILE – Receive a file on the dest machine.

RPC Params: xferId S transfer id fileName S file name fileSize I file size numThreads I num of transfer threads to open srcPort I starting port number srcIP S IP address of caller (string)

RPC Return: 0 transfer succeeded 1 transfer failed ————————————————————————

Receive a file on the dest machine.

Parameters:

data caller param data

Returns:

status code

References psReceiveFile().

2.11 dtsQueue.c File Reference

DTS queue interface.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include <stdarg.h>
#include "dts.h"
```

Defines

- #define **DEBUG** 1

Variables

- DTS ∗ **dts**
- int **dts_monitor**

2.11.1 Detailed Description

DTS queue interface.

DTSQUEUE.C

Author:

Mike Fitzpatrick

Date:

6/10/09

2.12 dtsSockUtil.c File Reference

DTS socket utilities.

```
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <netdb.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <fcntl.h>
#include <sys/errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include "dts.h"
#include "dtsPSock.h"
```

Functions

- int dts_openServerSocket (int port)
  Open a socket to be used on the 'server' side.
- int dts_openClientSocket (char *host, int port)
  Open a socket to be used on the 'client' side.
- int **dts_sockRead** (int fd, void *vptr, int nbytes)
- int **dts_sockWrite** (int fd, void *vptr, int nbytes)

2.12.1 Detailed Description

DTS socket utilities.

Author:

Mike Fitzpatrick

Date:

6/10/09

2.12.2 Function Documentation

2.12.2.1 int dts_openClientSocket (char * *host*, int *port*)

Open a socket to be used on the 'client' side.

dts_openClientSocket – Open a socket to be used on the 'client' side.

Parameters:

host host name

port port number to open

Returns:

socket descriptor

Referenced by psReceiveFile(), and psSendFile().

2.12.2.2 int dts_openServerSocket (int *port*)

Open a socket to be used on the 'server' side.

dts_openServerSocket – Open a socket to be used on the 'server' side.

Parameters:

port port number to open

Returns:

socket descriptor

Referenced by psReceiveFile(), and psSendFile().

2.13 dtsUtil.c File Reference

DTS Utility methods.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <ctype.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <signal.h>
#include <fcntl.h>
#include "dts.h"
```

Functions

- void dtsDaemonize (DTS *dts)
- void dts_sigHandler (int sig)
- void checksum (unsigned char *data, int length, ushort *sum16, uint *sum32)
- unsigned int addcheck32 (unsigned char *array, int length)
- char * dts_getLocalIP ()
- void measure_start (void)

  Start timer.

- void measure_stop (long transferred)

  Stop timer and print summary of transfer stats.

- double transferMb (long fileSize, int sec, int usec)

  Return throughput in megabits/s.

- double transferMB (long fileSize, int sec, int usec)

Return throughput in megabits/s.

- char * dtsGets (char *s, int len, FILE *fp)
- void **dtsError** (char *msg)
- char * **dts_printDTS** (DTS *dts, FILE *fd)
- char * **dts_printDTSClient** (DTS *dts, FILE *fd)
- char * **dts_printAllQueues** (DTS *dts, FILE *fd)
- char * **dts_printQueue** (dtsQueue *dtsq, FILE *fd)
- void **doprnt** (char *buf, char *format,...)

Variables

- struct timeval measure_start_tv
- struct timeval **io_tv** = {0, 0}

2.13.1 Detailed Description

DTS Utility methods.

DTSUTIL.C – Utility routines for the DTS.

Author:

Mike FItzpatrick

Date:

6/15/09

2.13.2 Function Documentation

2.13.2.1 unsigned int addcheck32 (unsigned char * *array*, int *length*)

ADDCHECK32 – Internet checksum, 32 bit unsigned integer version.

Parameters:

array data buffer

length length of data buffer (bytes)

Returns:

32-bit checksum value

Referenced by psReceiveStripe(), and psSendStripe().

2.13.2.2 void checksum (unsigned char * *data*, int *length*, ushort * *sum16*, uint * *sum32*)

CHECKSUM – Increment the checksum of a character array. The calling routine must zero the checksum initially. Shorts are assumed to be 16 bits, ints 32 bits.

Parameters:

data data buffer

length length of data buffer

sum16 16-bit checksum (output)

sum32 32-bit checksum (output)

2.13.2.3 char* dts_getLocalIP ()

DTS_GETLOCALIP – Get the local IP address as a string.

Returns:

character string of IP, e.g. "127.0.0.1"

2.13.2.4 void dts_sigHandler (int *sig*)

DTS_SIGHANDLER – Signal handler for the DTS daemon.

Parameters:

sig signal number

Referenced by dtsDaemonize().

2.13.2.5 void dtsDaemonize (DTS * *dts*)

DTSDAEMONIZE – Daemonize the DTS.

Parameters:

dts DTS struct pointer

Returns:

References dts_sigHandler().

2.13.2.6 char* dtsGets (char * *s*, int *len*, FILE * *fp*)

DTSGETS A smart fgets() function

Read the line; unlike fgets(), we read the entire line but dump characters that go past the end of the buffer. We accept CR, LF, or CR LF for the line endings to be "safe".

Parameters:

s string buffer

len length of buffer

fp file descriptor

Returns:

pointer to next string on the stream

Referenced by dts_loadConfig().

2.13.2.7 void measure_stop (long *transferred*)

Stop timer and print summary of transfer stats.

Parameters:

transferred number of bytes transferred

References measure_start_tv.

Referenced by psSpawnWorker().

2.13.2.8 double transferMB (long *fileSize*, int *sec*, int *usec*)

Return throughput in megabits/s.

Parameters:

fileSize number of bytes transferred

sec transfer time seconds

usec transfer time micro-seconds

Referenced by dts_xferPush().

2.13.2.9 double transferMb (long *fileSize*, int *sec*, int *usec*)

Return throughput in megabits/s.

Parameters:

fileSize number of bytes transferred

sec transfer time seconds

usec transfer time micro-seconds

Referenced by dts_xferPush().

2.13.3 Variable Documentation

2.13.3.1 struct timeval measure_start_tv

Transfer and measurement utilities.

Referenced by measure_start(), and measure_stop().

Index