NRSE Reference Manual

1.0

Generated by Doxygen 1.2.14

Mon Sep 1 10:54:08 2003

Contents

Chapter 1

NRSE Package List

1.1 NRSE Package List

Here are the packages with brief descriptions (if available):

Chapter 2

NRSE Hierarchical Index

2.1 NRSE Class Hierarchy

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

Chapter 3

NRSE Compound Index

3.1 NRSE Compound List

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

Chapter 4

NRSE Package Documentation

4.1 Package client

Classes

- class **Client**

  Console interface to test functions of **BeepClient** (p. 14) *class.*

- class **ClientProfile**

  Beep profile implementation for **NRSE** (p. 30).

- class **ClientProfile::ReplyThread**
- class **Qos**

  Launches the GUI for reservation.

- class **QosAddSLA**

  makes SLS object from the value which is specified by user from GUI and makes xml file.

- class **QosAddUser**

  Fields are User, Password, Public Key, Private Key and Generate to generate public & private key pair.

- class **QosDeleteSLA**
- class **QosLayout**
- class **QosMenu**
- class **QosMenu::EventHandler**
- class **QosModifySLA**
- class **QosModifySLA::EventHandler**
- class **QosQuery**
- class **QosQuery::EventHandler**

4.2 Package nrse

Interfaces

- interface **Router**

Classes

- class **AuthenticationException**
- class **CreateTable**
- class **Database**

  TEST QoS reservation database.

- class **IPerf**

  monitors TCP/UDP packets flow by using iperf.

- class **IPerf::ICanvas**

  plot the data onto the canvas.

- class **MyCanvas**
- class **Notification**

  sends the notification from the server to the client in each stage where we will execute the commands.

- class **NRSE**

  Represents an actual NRSE server.

- class **NRSEProfile**

  Beep profile implementation for **NRSE** (p. 30) *This class describes how to Start, Close the Beep connection depending on configuration.*

- class **NRSEProfile::ReplyThread**
- class **QueryRequest**

  Make a QueryRequest from the <simple_query> xml file or the parameter user and credential.

- class **QueryResults**

  Holds SLS objects that result from database query.

- class **ReservationNotMetException**
- class **ReservationQuota**
- class **SLS**
- class **SLSactivator**

  Ask the database object whether there are reservations which is ready to run in every certain time interval.

- class **TCRouter**

  Implements SLS on a Linux router via TC.

- class **TCRouter::Exec**
- class **TCRouter::StreamGobbler**

- class **WipeDatabase**

  This class is used to delete the all database, QoS_rsv, QoS_user, ScheduleI.

- class **XMLErrorHandler**

  Handles XML parser errors.

4.3 Package test

Classes

- class **AuthTest**

  Tests security and authentication functions.

- class **ClientTest**

  Tests functions of **Client** (p. 15)*, and indirectly* **BeepClient** (p. 14)*.*

- class **TestAll**

  Runs all tests.

- class **TestDatabase**

  Tests function of **Database** (p. 20) *class which interfaces to the database.*

- class **TestNotify**

  Tests notification function.

- class **TestUtil**

  Tests utility functions and SLS class.

4.4 Package util

Classes

- class **BeepClient**

  beep client code.

- class **Config**

  Data structure to store configurable options.

Chapter 5

NRSE Class Documentation

5.1 AuthTest Class Reference

Tests security and authentication functions.

Public Methods

- void **testAuthenticate** () throws Exception
  Sign SLS and authenticate user from database.
- void **testSign** () throws Exception
  Sign SLS and verify signature.
- void **testVerify** () throws Exception
  Create and verify good and bad signatures - bad signature must fail.

5.1.1 Detailed Description

Tests security and authentication functions.

Author:
Richard Smith

Revision:
1.7

Date:
2003/08/27 11:11:32

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

- AuthTest.java

5.2 BeepClient Class Reference

beep client code.

5.2.1 Detailed Description

beep client code.

make a connection depending on properties Here are NRSE_URI and CLIENT_URI profiles

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

- BeepClient.java

5.3 Client Class Reference

Console interface to test functions of **BeepClient** (p. 14) class.

Public Methods

- **Client** (String configFile)

  Initialises settings using a configuration file.

- **Client** (String _host, int _port, **Config** c)

  Initialises settings based on an existing **Config** (p. 19) *object.*

- int **sendSLS** (SLS sls)

  Opens a BEEP channel to the **NRSE** (p. 30) *and transmits the SLS.*

Static Public Methods

- SLS **inputSLS** (InputStream in) throws IOException

  Reads in a list of newline-separated parameters and uses them to construct a SLS object.

5.3.1 Detailed Description

Console interface to test functions of **BeepClient** (p. 14) class.

This was essential before the GUI interface was created. Now the GUI should be preferred by users, and this class just used for testing.

Author:
Keiko Tada , Toshi Aiyoshi , Richard Smith , Andy Liow

Revision:
1.20

Date:
2003/08/26 07:31:37

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Client::Client (String *configFile*) [inline]

Initialises settings using a configuration file.

Will create a **BeepClient** (p. 14) object to do the actual work.

Parameters:
configFile the name of an XML configuration file, must be in classpath

5.3.2.2 Client::Client (String *_host*, int *_port*, Config *c*) [inline]

Initialises settings based on an existing **Config** (p. 19) object.

Hostname and port number to connect to are specified rather than taken from defaults, however. Will create a **BeepClient** (p. 14) object to do the actual work.

Parameters:

_host the hostname of the **NRSE** (p. 30) to connect to

_port the port number of the **NRSE** (p. 30) to connect to

c a configuration object, intended to ensure this Client has the same configuration as the class which created it

5.3.3 Member Function Documentation

5.3.3.1 SLS Client::inputSLS (InputStream *in*) throws IOException [inline, static]

Reads in a list of newline-separated parameters and uses them to construct a SLS object.

Returns:

SLS object.

Parameters:

in InputStream containing parameter (so parameters may be read from file or from console)

5.3.3.2 int Client::sendSLS (SLS *sls*) [inline]

Opens a BEEP channel to the **NRSE** (p. 30) and transmits the SLS.

We first serialize the SLS into XML format, and then use **BeepClient** (p. 14) to do that actual sending.

Returns:

error code

Parameters:

sls SLS to send

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

- Client.java

5.4 ClientProfile Class Reference

Beep profile implementation for **NRSE** (p. 30).

5.4.1 Detailed Description

Beep profile implementation for **NRSE** (p. 30).

Author:
Keiko Tada, Toshi Aiyoshi, Richard Smith, Andy Liow

Version:

Revision:
1.5

,

Date:
2003/08/01 17:32:50

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

- ClientProfile.java

5.5 ClientTest Class Reference

Tests functions of **Client** (p. 15), and indirectly **BeepClient** (p. 14).

Public Methods

- void **testClientSendSLS** () throws Exception

  Reads SLS parameters (from file) to client which should then create and send SLS to **NRSE** (p. 30).

- void **testClientDelete** () throws Exception

  Creates request for deleletion and uses client to it send to **NRSE** (p. 30).

5.5.1 Detailed Description

Tests functions of **Client** (p. 15), and indirectly **BeepClient** (p. 14).

Author:
Richard Smith

Revision:
1.6

Date:
2003/08/27 11:11:32

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

- ClientTest.java

5.6 Config Class Reference

Data structure to store configurable options.

5.6.1 Detailed Description

Data structure to store configurable options.

Author:
Keiko Tada, Toshi Aiyoshi, Richard Smith, Andy Liow

Version:

Revision:
1.3

,

Date:
2003/07/31 16:20:07

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

- Config.java

5.7 Database Class Reference

TEST QoS reservation database.

Public Methods

- **Database** (**Config** c) throws SQLException
- void **setActive** (int id) throws SQLException, Exception

  Set the ACTIVE value to true in the Qos_rsv database.

- void **setStartedNotify** (int id) throws SQLException, Exception

  Set the START_NOTIFY flag after the start notification is issued.

- void **setEndedNotify** (int id) throws SQLException, Exception

  Set the END_NOTIFY flag after the start notification is issued.

- void **deleteSLS** (int id) throws SQLException, Exception

  Deletes one tuple from the QoS_rsv table specified by ID.

- int **addSLS** (SLS sls) throws SQLException, ReservationNotMetException, IOException

  This method is left only for compatability with single domain test environment may be removed.

- int **addSLS** (SLS sls, int ifaceIn, int ifaceOut) throws SQLException, ReservationNotMetException, IOException

  Adds the SLS into QoS reservation tables.

- SLS **getSLS** (int id) throws SQLException, java.text.ParseException

  Gets a SLS specified by ID from the QoS_rsv table.

- int[] **queryByUser** (String user) throws Exception

  Queries all SLSs which were made by the specified user.

- void **createDatabase** ()

  Deletes the previous tables (QoS_rsv, QoS_user, ScheduleI) and creates new ones.

- void **check** (SLS sls) throws SQLException, ReservationNotMetException

  This method is left only for compatability with single domain test environment may be removed.

- void **check** (SLS sls, int iface) throws SQLException, ReservationNotMetException

  Checks the local ScheduleI table whether a requested QoS reservation is acceptable or not.

- int **checkSchedule** (String ts) throws SQLException

  Retreives reservation info from Schedule table and returning bandwidth value corresponding to the time specified.

- boolean **queryByUserFromQoSuser** (String user) throws SQLException

  Checks if there already exists the same use name in the QoS_user table.

- void **addUser** (String user, String pgpKey) throws SQLException, Exception

Adds a user and his/her pgp key to the QoS_user table.

- void **updateUser** (String user, String pgpKey) throws SQLException, Exception

  <for the future use> Changes a user's PGP key in the QoS_user table.

- void **deleteUser** (String user) throws SQLException

  <for the future use> Deletes a user from the QoS_user table.

- String **getPGPKey** (String name) throws SQLException

  Returns the PGP key from the user name.

- void **authenticate** (String xml, SLS sls) throws AuthenticationException, SQLException

Static Public Methods

- void **main** (String[] args) throws SQLException

  does nothing.

Private Methods

- Connection **makeConnection** (String url, String user, String password) throws SQLException

  Registers a proper JDBC driver and establishes a connection between **NRSE** (p. 30) *server and SQL server.*

5.7.1 Detailed Description

TEST QoS reservation database.

The database stores the following information:

- Reserved SLS
- User information
- Network resource allocation schedule

The database consists of three tables to store the above information: QoS_rsv, QoS_user and ScheduleI.

- QoS_rsv: holds the contents of SLSs and notification information.
- QoS_user: holds 'user name' and his/her 'PGP key'.
- ScheduleI: holds free bandwidth for each network interface.

Main methods addSLS check

Version:
1.53

Author:
z15_2 group(Andy, Keiko, Richard, Toshi)

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Database::Database (Config *c*) throws SQLException [inline]

Parameters:
c configuarable parameters.

See also:
util.Config , **makeConnection**(String, String, String) (p. 25)

5.7.3 Member Function Documentation

5.7.3.1 int Database::addSLS (SLS *sls*, int *ifaceIn*, int *ifaceOut*) throws SQLException, ReservationNotMetException, IOException [inline]

Adds the SLS into QoS reservation tables.

Rough steps are:

1. Checks bandwidths of local links with the local ScheduleI table.
2. Requests remote **NRSE** (p. 30) to make the reservation.
3. If the remote reservation was made successfully, adds the SLS to the local tables.
4. If the request is not met either locally or remotely, cancel the SLS.
5. Decrease the bandwidth value from the ScheduleI table.

See also:
check(SLS, int) (p. 23)

Parameters:
sls Service Level Specification which was received by **NRSE** (p. 30).

ifaceIn ID of the network interaface which the flow comes into.

ifaceOut ID of the network interface which the flow goes out.

Returns:
If the SLS has been added into the QoS reservation tables, addSLS assigns an unique ID for it and returns it. This will be changed from id to the whole SLS reserved in the future, because multi-domain non-realtime reservation needs it.

Exceptions:
ReservationNotMetException is throwed when at the lease one bandwidth of the links are short. QoS reservation is not made.

5.7.3.2 int Database::addSLS (SLS *sls*) throws SQLException, ReservationNotMetException, IOException [inline]

This method is left only for compatability with single domain test environment may be removed.

* -

5.7.3.3 void Database::addUser (String *user*, String *pgpKey*) throws SQLException, Exception [inline]

Adds a user and his/her pgp key to the QoS_user table.

Parameters:

user username which will be registered in the Qos_user table.

pgpKey A private key for the user.

Exceptions:

SQLException is throwed when a SQL command can not be accepted.

5.7.3.4 void Database::authenticate (String *xml*, SLS *sls*) throws AuthenticationException, SQLException [inline]

See also:

SLS::authenticate(String, String).

Parameters:

xml xml file of the SLS. Check the signature here.

Exceptions:

AuthenticationException is throwed when authentication is failed.

SQLException is throwed when a SQL command can not be accepted.

5.7.3.5 void Database::check (SLS *sls*, int *iface*) throws SQLException, ReservationNotMetException [inline]

Checks the local ScheduleI table whether a requested QoS reservation is acceptable or not.

Parameters:

sls Requested SLS

iface Network interface ID (the number assigned in the config file) which will be checked.

Exceptions:

ReservationNotMetException is thrown when the network interface does not have enough free bandwidth.

5.7.3.6 void Database::check (SLS *sls*) throws SQLException, ReservationNotMetException [inline]

This method is left only for compatability with single domain test environment may be removed.

* -

5.7.3.7 int Database::checkSchedule (String *ts*) throws SQLException [inline]

Retreives reservation info from Schedule table and returning bandwidth value corresponding to the time specified.

Only for NRSETest.testDatabaseAdd().

5.7.3.8 void Database::createDatabase () [inline]

Deletes the previous tables (QoS_rsv, QoS_user, ScheduleI) and creates new ones.

The ScheduleI is initialised with the initial bandwidths specified in a config file.

5.7.3.9 void Database::deleteSLS (int *id*) throws SQLException, Exception [inline]

Deletes one tuple from the QoS_rsv table specified by ID.

Multi-domain deletion has not been implemented yet.

Parameters:
id SLA's id number which is assigned by **addSLS** (p. 22). ID field is in Qos_rsv table.

5.7.3.10 void Database::deleteUser (String *user*) throws SQLException [inline]

<for the future use> Deletes a user from the QoS_user table.

Parameters:
user username which is registered in the Qos_user table

Exceptions:
SQLException is throwed when a SQL command can not be accepted.

5.7.3.11 String Database::getPGPKey (String *name*) throws SQLException [inline]

Returns the PGP key from the user name.

Parameters:
name username which is registered in the Qos_user table

Exceptions:
SQLException is throwed when a SQL command can not be accepted.

5.7.3.12 SLS Database::getSLS (int *id*) throws SQLException, java.text.ParseException [inline]

Gets a SLS specified by ID from the QoS_rsv table.

Parameters:
id SLS's ID which was assigned by addSLS when the reservation was made.

Returns:
SLS

5.7.3.13 Connection Database::makeConnection (String *url*, String *user*, String *password*) throws SQLException [inline, private]

Registers a proper JDBC driver and establishes a connection between **NRSE** (p. 30) server and SQL server.

Parameters:

url is an address where the SQL server exists.

user is an account name who is administer the SQL database.

password The user's password.

5.7.3.14 int [] Database::queryByUser (String *user*) throws Exception [inline]

Queries all SLSs which were made by the specified user.

Parameters:

user User name

Returns:

int[] SLSs'IDs in integer array.

5.7.3.15 boolean Database::queryByUserFromQoSuser (String *user*) throws SQLException [inline]

Checks if there already exists the same use name in the QoS_user table.

Called by addUser method only. (public to enable testing)

Parameters:

user is queried username.

Exceptions:

SQLException is throwed when a SQL command can not be accepted.

5.7.3.16 void Database::setActive (int *id*) throws SQLException, Exception [inline]

Set the ACTIVE value to true in the Qos_rsv database.

Parameters:

id SLA's id number which is assigned by **addSLS** (p. 22). ID field is in Qos_rsv table.

5.7.3.17 void Database::setEndedNotify (int *id*) throws SQLException, Exception [inline]

Set the END_NOTIFY flag after the start notification is issued.

Parameters:

id SLA's id number which is assigned by **addSLS** (p. 22). ID field is in Qos_rsv table.

5.7.3.18 void Database::setStartedNotify (int *id*) throws SQLException, Exception [inline]

Set the START_NOTIFY flag after the start notification is issued.

Parameters:

id SLA's id number which is assigned by **addSLS** (p. 22). ID field is in Qos_rsv table.

5.7.3.19 void Database::updateUser (String *user*, String *pgpKey*) throws SQLException, Exception [inline]

<for the future use> Changes a user's PGP key in the QoS_user table.

Parameters:

user username which is registered in the Qos_user table

pgpKey A new private key for the user.

Exceptions:

SQLException is throwed when a SQL command can not be accepted.

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

- Database.java

5.8 IPerf Class Reference

monitors TCP/UDP packets flow by using iperf.

Static Public Methods

- void **main** (String args[]) throws Exception
  executes the iperf command, and stores the rate of the packet.

5.8.1 Detailed Description

monitors TCP/UDP packets flow by using iperf.

The result is shown on the screen window.

5.8.2 Member Data Documentation

5.8.2.1 Color [] IPerf::colors [static, private]

Initial value:

```
{Color.RED, Color.BLUE, Color.GREEN, Color.YELLOW,
                            Color.BLACK, Color.CYAN, Color.MAGENTA, Color.ORANGE,
                            Color.PINK, Color.WHITE
   }
```

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

- IPerf.java

5.9 IPerf::ICanvas Class Reference

plot the data onto the canvas.

5.9.1 Detailed Description

plot the data onto the canvas.

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

- IPerf.java

5.10 Notification Class Reference

sends the notification from the server to the client in each stage where we will execute the commands.

5.10.1 Detailed Description

sends the notification from the server to the client in each stage where we will execute the commands.

The trigger for the notifications are,

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

- Notification.java

5.11 NRSE Class Reference

Represents an actual NRSE server.

Public Methods

- **NRSE** (String uri, int port, Profile p, **Config** c) throws Exception

  Parses the beepd element in the configuration file and loads the classes for the specified profiles.

Static Public Methods

- void **main** (String[] args) throws Exception

  Loads configuration file and launched BEEP server.

Static Private Methods

- ProfileConfiguration **parseProfileConfig** (NodeList profileConfig) throws Exception

  Parses the child elements of a profile element into a `ProfileConfiguration` *object.*

5.11.1 Detailed Description

Represents an actual NRSE server.

Running this class creates an instance of NRSE, which then launches a BEEP server using **NRSEProfile** (p. 32). All of the functionality of the NRSE is in **NRSEProfile** (p. 32). By default, configuration options are loaded from a file named 'nrse.properties' in the classpath, but a different configuration file may be specified on the command line.

This class was originally based on the code for the BEEP Daemon launcher, but we found that to be an order of magnitude more complicated than we needed, because it it supported launching multiple BEEP servers with multiple profiles. We only need one server with one profile.

Author:
Keiko Tada , Toshi Aiyoshi , Richard Smith , Andy Liow

Revision:
1.15

Date:
2003/08/26 07:31:37

5.11.2 Constructor & Destructor Documentation

5.11.2.1 NRSE::NRSE (String *uri*, int *port*, Profile *p*, Config *c*) throws Exception [inline]

Parses the beepd element in the configuration file and loads the classes for the specified profiles.

Parameters:
serverConfig &ltbeepd&gt configuration element.

5.11.3 Member Function Documentation

5.11.3.1 void NRSE::main (String *args*[]) throws Exception [inline, static]

Loads configuration file and launched BEEP server.

Parameters:
args first argument is name of configuration file

5.11.3.2 ProfileConfiguration NRSE::parseProfileConfig (NodeList *profileConfig*) throws Exception [inline, static, private]

Parses the child elements of a profile element into a `ProfileConfiguration` object.

Parameters:
profileConfig list of parameter child elements of a profile element.

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

- NRSE.java

5.12 NRSEProfile Class Reference

Beep profile implementation for **NRSE** (p. 30) This class describes how to Start, Close the Beep connection depending on configuration.

5.12.1 Detailed Description

Beep profile implementation for **NRSE** (p. 30) This class describes how to Start, Close the Beep connection depending on configuration.

also receives Beep Message and Reply the beep reply message

Author:
Keiko Tada, Toshi Aiyoshi, Richard Smith, Andy Liow

Version:

Revision:
1.19

,

Date:
2003/08/11 14:27:14

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

- NRSEProfile.java

5.13 Qos Class Reference

Launches the GUI for reservation.

5.13.1 Detailed Description

Launches the GUI for reservation.

Launches the beep client to send SLS, and launches the beep server to receive **Notification** (p. 29). The GUI for reservation field contains User, Password, Credentials, Server Address, Server Port. After user entering the all field and if the button "Add SLS" was pressed, QosAddSLA() will be called.

The GUI for actions are Add User. In the "Add User" action, call QosAddUser(). After that, open the file ".qoskey" and read private key of the user.

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

- Qos.java

5.14 QosAddSLA Class Reference

makes SLS object from the value which is specified by user from GUI and makes xml file.

5.14.1 Detailed Description

makes SLS object from the value which is specified by user from GUI and makes xml file.
Then send that beep file to the server by beep.
The documentation for this class was generated from the following file:

- QosAddSLA.java

5.15 QosAddUser Class Reference

Fields are User, Password, Public Key, Private Key and Generate to generate public & private key pair.

5.15.1 Detailed Description

Fields are User, Password, Public Key, Private Key and Generate to generate public & private key pair.

After pressed "Add user" button, the database class adduser method will be called. after pressed "generate" button, the SLS class generateKey method will be called.

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

- QosAddUser.java

5.16 QueryRequest Class Reference

Make a QueryRequest from the <simple_query> xml file or the parameter user and credential.

Public Methods

- String **serialize** () throws IOException, ParserConfigurationException
  return the xml file with the parameter which is specified in the QueryRequest object.

5.16.1 Detailed Description

Make a QueryRequest from the <simple_query> xml file or the parameter user and credential.

Author:
Keiko Tada, Toshi Aiyoshi, Richard Smith, Andy Liow

Version:

Revision:
1.3

,

Date:
2003/08/10 03:04:26

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

- QueryRequest.java

5.17 QueryResults Class Reference

Holds SLS objects that result from database query.

Public Methods

- **QueryResults** (String x) throws ParserConfigurationException, SAXException, IOException,java.text.ParseException

  Takes an XML document containing multiple SLSs and builds QueryResults.

5.17.1 Detailed Description

Holds SLS objects that result from database query.

Author:
Keiko Tada, Toshi Aiyoshi, Richard Smith, Andy Liow

Version:

Revision:
1.6

,

Date:
2003/07/09 15:41:06

5.17.2 Constructor & Destructor Documentation

5.17.2.1 QueryResults::QueryResults (String *x*) throws ParserConfiguration-Exception, SAXException, IOException,java.text.ParseException [inline]

Takes an XML document containing multiple SLSs and builds QueryResults.

Parameters:
x

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

- QueryResults.java

5.18 SLSactivator Class Reference

Ask the database object whether there are reservations which is ready to run in every certain time interval.

5.18.1 Detailed Description

Ask the database object whether there are reservations which is ready to run in every certain time interval.

(time is specified in the config file) If tehre are, activate SLS by using **TCRouter** (p. 39). Also ask the database object whether there are ready notification for both start and end. If there are, make beep client object and send a message to the beep server which is the **NRSE** (p. 30) client.

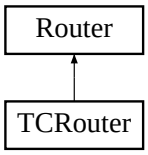
The documentation for this class was generated from the following file:

- SLSactivator.java

5.19 TCRouter Class Reference

Implements SLS on a Linux router via TC.

Inheritance diagram for TCRouter::



Public Methods

- void **activateSLS** (SLS sls) throws IOException

  Sends TC commands to activate reservation
  qdisc -> the set of algorithms you wish to have handle the packets going through a given device or class
  class -> sections of a class-based queueing discipline.

- int **getInterface** (String dst, int direction, String src) throws IOException

  Finds appropriate network interface used for route to ip and returns it as a number which refers to the array of interfaces in **Config** (p. 19).

Static Public Methods

- boolean **ipLookup** (String net, String ip)

  finds the ip address which appropriates to the interface.

- int **getNetAddr** (String ipStr, int prefLen)

  returns the masked ip address depending on the length of the prefix.

5.19.1 Detailed Description

Implements SLS on a Linux router via TC.

Author:
Keiko Tada, Toshi Aiyoshi, Richard Smith, Andy Liow

Version:

Revision:
1.19

,

Date:
2003/08/11 13:37:57

5.19.2 Member Function Documentation

5.19.2.1 void TCRouter::activateSLS (SLS *sls*) throws IOException [inline]

Sends TC commands to activate reservation

qdisc -> the set of algorithms you wish to have handle the packets going through a given device or class

class -> sections of a class-based queueing discipline.

set peak rate

filter -> a way to put packets into queues : set src & dst

Parameters:
sls sls which you want to activate

Exceptions:
IOException I/O

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

- TCRouter.java

5.20 TestAll Class Reference

Runs all tests.

5.20.1 Detailed Description

Runs all tests.

Author:
Richard Smith

Revision:
1.2

Date:
2003/08/27 11:14:08

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

- TestAll.java

5.21 TestDatabase Class Reference

Tests function of **Database** (p. 20) class which interfaces to the database.

Public Methods

- void **testDatabaseDelete** () throws Exception
  Adds SLS to database, deletes it, checks it has gone.
- void **testDatabaseQuery** () throws Exception
  Adds two SLSs to database, then queries for them.
- void **testQuery** () throws Exception
  Processes XML query and executes it.
- void **testDatabaseAdd** () throws Exception
  Adds some SLSs into database, pulls them out, checks they are the same as before they went in.
- void **testDatabaseCheck** () throws Exception
  Adds lots SLSs into database, some of which exceed bandwidth limits.
- void **testDatabaseAddSLSM** () throws Exception
  Ensures database makes correct changes to SLS to meet a non-realtime request.

5.21.1 Detailed Description

Tests function of **Database** (p. 20) class which interfaces to the database.

Author:
Toshi Aiyoshi , Richard Smith

Revision:
1.9

Date:
2003/08/27 11:11:32

5.21.2 Member Function Documentation

5.21.2.1 void TestDatabase::testDatabaseCheck () throws Exception [inline]

Adds lots SLSs into database, some of which exceed bandwidth limits.

Check method should notice this.

5.21.2.2 void TestDatabase::testDatabaseDelete () throws Exception [inline]

Adds SLS to database, deletes it, checks it has gone.

This test might not work if we dont have exclusive access to database

5.21.2.3 void TestDatabase::testDatabaseQuery () throws Exception [inline]

Adds two SLSs to database, then queries for them.

This test doesnt really test the results properly.

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

- TestDatabase.java

5.22 TestNotify Class Reference

Tests notification function.

Public Methods

- int **sendRequest** (String notification, String host, int port)
  Launches a client and sends it a notification.

5.22.1 Detailed Description

Tests notification function.

Author:
Richard Smith

Revision:
1.3

Date:
2003/08/27 11:14:08

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

- TestNotify.java

5.23 TestUtil Class Reference

Tests utility functions and SLS class.

Public Methods

- void **testCloneEquals** () throws Exception
  Tests that SLS.clone() works.
- void **testDeleteSLS** () throws Exception
  Processes deletion request XML and checks against known correct values.
- void **testReadSLS** () throws Exception
  Read XML format SLS from file and compare to known-correct values.
- void **testWriteSLS** () throws Exception
  Write SLS to XML format file and read it back again.

5.23.1 Detailed Description

Tests utility functions and SLS class.

Author:
Richard Smith

Revision:
1.8

Date:
2003/08/27 11:14:09

5.23.2 Member Function Documentation

5.23.2.1 void TestUtil::testWriteSLS () throws Exception [inline]

Write SLS to XML format file and read it back again.

Now fixed to use memory buffer, not file on disk

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

- TestUtil.java

5.24 WipeDatabase Class Reference

This class is used to delete the all database, QoS_rsv, QoS_user, ScheduleI.

5.24.1 Detailed Description

This class is used to delete the all database, QoS_rsv, QoS_user, ScheduleI.

This uses "DROP" command of SQL. And create the new database from the scratch. At the very beginning to start the demo, we should call wipedatabase to clear all previous information.

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

- WipeDatabase.java

5.25 XMLErrorHandler Class Reference

Handles XML parser errors.

5.25.1 Detailed Description

Handles XML parser errors.

Author:
Keiko Tada, Toshi Aiyoshi, Richard Smith, Andy Liow

Version:

Revision:
1.1

,

Date:
2003/06/27 16:41:16

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

- XMLErrorHandler.java