RatheDG Cachalot System 1.2.0
User Manual

License Agreement

This License Agreement ("Agreement") is a legal agreement between you, the user of RatheDG Cachalot System ("System"), and Rathe Development Group ("RatheDG"). The user of System is physical or juridical person who has a copy of System or any of its parts on hard drive (or any other media) or uses a running instance of System or any of its parts.

If you do not agree with Agreement you must stop using System and remove all copies of System from your environment. Any use that is not allowed by Agreement shall result in immediate and automatic termination of Agreement and may result in criminal and/or civil prosecution.

You may not decompile, disassemble or otherwise reverse engineer code of System. You may not modify System (including documentation files or any other files supplied with System, except configuration files and batch scripts). You may not rent or resell for profit the System. You may not redistribute System in any other form beside original unchanged distributive package.

SYSTEM IS DISTRIBUTED "AS IS". NO WARRANTY OF ANY KIND IS EXPRESSED OR IMPLIED. YOU USE IT AT YOUR OWN RISK. RATHEDG WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THE SYSTEM OR BECAUSE OF IMPOSSIBILITY TO USE SYSTEM.

Current License Agreement is subject to change without any notifications.

All rights not expressly granted are reserved by RatheDG.

1. General Description

1.1. Purpose of the System

Often it becomes necessary for the organizations to gather statistical data about the usage of the network or/and the computers connected to another net and monitor changes in real-time. Typical examples of these organizations are so-called Internet Cafe and Internet service providers. Statistical data includes the information about user sessions, payments, network traffic, etc.

Cachalot System provides an ability to gather statistical data and have a convenient access to this data and provides en ability of monitoring the events over the network. The system is not limited to a particular network, it uses pure TCP/IP and HTTP protocols; it can be used in both Intranet and Internet.

The system requires manual installation and system administrator support.

Java FAQ – http://www.ibiblio.org/javafaq/javafaq.html.

1.2. Definitions

This chapter gives the description of entities and notions involved into the system.

Host – a computer connected to network. Host can start up and shut down.

Internal host – a host registered in system.

External host – a host not registered in system.

User – a person using an internal host. The user can log on and log off a host. At each moment of time there can be only one user logged on a particular internal host.

Event – some action occurred on particular internal host, optionally associated with a user. The following events are used: startup and shutdown (host starting up and host shutting down), logon and logoff (user logging on and user logging off).

Group – a set of users. User can belong to a single group. Each group has a role assigned by administrator. There are three roles being used in the system:

- Administrator – can manage users, groups and hosts, Administrator is also an Observer and a User
- Observer – can monitor events and see statistics, Observer is also a User
- User – can do nothing in the meaning of the system except to be associated with events.

Traffic is the data being transmitted over the network between internal and external hosts. Typically it is incoming/outgoing data from/to Internet/local network transmitted through a router (any other configuration including router may be used).

Two subsequent logon and logoff events of the user on the same host define a **session**. Startup, shutdown and logon are treated as logoff event if there was a preceding logon event. All traffic transmitted from/to this host during the session is associated with the session.

Often it becomes necessary to measure a cost of the user's session depending on the duration of the session and its traffic. It is supposed that cost C is

$$C=C_h*P_h*D + C_i*P_i*T_i + C_o*P_o*T_o$$

where C_h, C_i and C_o are coefficients defined for a user or a group; P_h, P_i and P_o are prices for one hour of a session, megabyte of incoming traffic and megabyte of outgoing traffic; D, T_i and T_o are the duration of a session, total incoming traffic of a session, and total outgoing traffic of a session. Thus the amount C can be calculated for each session and the term 'calculated amount for a session' means C calculated for the particular session.

Operation – some abstract service that can be measured with an integer number of units (instances). Operation has a price assigned, price for a single unit. Sample operations: printing a page, burning a CD, making a cup of coffee, etc. ('Operations' functionality is realized as a plug-in and is optional).

Internal hosts, users, groups and operations require manual registration in the system by administrator.

1.3. List of Features

List of system features is presented below. Please, pay your attention to future plans if you hesitate whether it meets your needs. Features are described in short and simple manner, take a look at our online demo for more details: http://www.rathedg.com/products/cachalot/demo/.

- User sessions registration.
- Gathering the information about the traffic transmitted over the router.
- Administrator interface for managing users, groups, hosts, operations and system settings.
- Observer interface for real-time active sessions monitoring.
- Observer interface for session payments registration.
- Observer interface for operations registration.
- Observer interface for statistics generation. To have an idea about what a report is, see the summary list of sessions report columns (report based on information gathered about sessions):
    - user name
    - group the user belongs to
    - host the user is (was) logged on
    - logon/logoff time
    - duration of the session
    - amount of incoming/outgoing/summary traffic for the whole session/per external host/per external port
    - amount calculated
    - amount collected.

  List of traffic report columns:
    - internal host
    - internal port
    - remote host
    - remote port
    - incoming/outgoing/summary traffic

  List of operations report columns:
    - operation
    - cashier (Observer who registered operation)
    - date
    - number of units
    - payment amount

  Report interface is flexible and provides an ability to get various kinds of statistics. There is also ability present to switch the columns on/off and to specify particular hosts, users, etc for which to generate a report.
- Administrator interface for removal of registered data from the database.

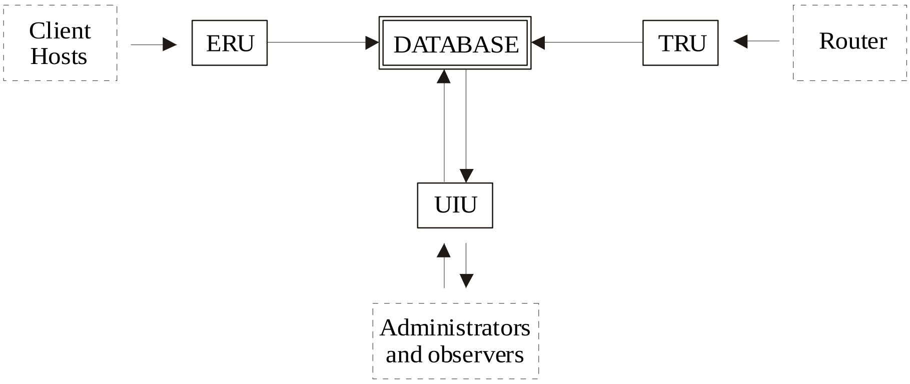
2. Architecture of System

2.1. The System Structure

The system consists of three units: Event Registration Unit (ERU), Traffic Registration Unit (TRU) and User Interface Unit (UIU). The information about the purpose of each unit follows:

- ERU – registers events (startup, logon, logoff, shutdown) in the database
- TRU – registers traffic in the database
- UIU – user interface for management, monitoring and generation of statistical reports.

ERU and UIU units must work on a single computer. TRU unit may work on another computer as well.



There is also a possibility to install optional plug-ins to UIU unit. Custom plug-ins may be developed to meet your specific user interface needs at any moment.

2.2. Event Registration Unit

ERU has the client-server architecture. Its server-side part (ERU/S) runs on any computer that is accessible via TCP/IP from all client computers. Server-side part must have an ability to access the database. Client-side part (ERU/C) runs on each client computer and must have an ability to connect ERU/S via TCP/IP. All the communications between ERU/S and ERU/C are performed via pure TCP/IP via single TCP/IP port.

When the event is about to happen, ERU/C connects to ERU/S, notifies it that event is about to happen and provides all the information concerning the event. ERU/S registers event in the database.

2.3. Traffic Registration Unit

TRU captures the information about the traffic (IP packets) and puts it to the database. TRU captures the information about the traffic between internal and external hosts only; i.e. it skips information about internal-to-internal and external-to-external traffic.

TRU must have an access to the database.

2.4. User Interface Unit

UIU provides the user interface for Administrators and Observers: generation of statistics, monitoring and management of system. This interface is web-based; i.e. the user accesses it using the web browser navigating to the web server running the UIU. UIU must have an access to the database.

User interface has multi-language support. Currently it has English and Russian translations and may be translated to any language at any time (it may require additional help of customer interested in particular translation).

3. Requirements

This chapter does not cover hardware requirements. The system does not require any special hardware base. This chapter covers the requirements to the software and environment that system relies on.

A very important part of the system is database. The system is not limited to be used with a particular database and any database may be used if it meets noticed requirements. It must have SQL interface (SQL/92 entry level is required), and the suitable JDBC or ODBC driver for this interface must be available.

Environmental requirements for each unit:
- ERU
  - ERU/C – the ability to connect to ERU/S via single TCP/IP port
  - ERU/S – the ability to listen the single TCP/IP port and access the database
- TRU –privileges necessary for capturing traffic, database access
- UIU – JSP container and database access.

4. Implementations

4.1. Event Registration Unit

Since ERU has client-server architecture client and server may have independent implementations.

4.1.S. Implementation of ERU/S

ERU/S is implemented in Java and requires JRE 1.3.0 or compatible. To learn about the hardware requirements see the corresponding documentation of JRE for your platform.

4.1.C. Implementations of ERU/C

Command-line clients can be executed by scripts (e.g. host start-up script and user's logon script). In case of any kind of error command-line clients write message to the standard output and exit with result code '1', otherwise with result code '0'.

4.1.C.1. Java Command-line Client

Java command-line ERU/C requires JRE 1.3.0 or compatible.

Usage: *Client host port password <timeout ms> <number of attempts> action [username]* where *action* is one of *startup*, *shutdown*, *logon* and *logoff*. Username is for logon action only. *Host* and *port* must locate the running ERU/S. *Password* is the password which client must provide for authentication; this password is a part ERU/S configuration.

4.1.C.2. Native Command-line Client

Native command-line ERU/C usage syntax is equal to syntax of Java command-line client.

Native command-line client distributed with current version of Cachalot System is compiled for x86 MS Windows 32bit platform; it may be compiled for any other platform (possibly for additional payment) at any time.

4.1.C.3. Windows Integrated Client

Windows integrated ERU/C client is implemented as Windows DLL participating in place of Windows system DLL. It can be used in MS Windows NT4.0 or compatible (at the moment Windows 2000, Windows XP).

The advantage of integration is that there's no need in search of a place where to perform a call of command-line client. DLL captures and sends events automatically; thus no additional scripting or development is necessary. Windows integrated ERU/C can be configured to log events and results of their sending in a local log file.

4.2. Traffic Registration Unit

TRU is implemented as java application. TRU is built on JPcap library, which is java interface to well-known WinPcap/LibPcap library. In fact, traffic registration is performed by underlying library. TRU handles information about traffic packets provided by library, summarizes (if configured) information and fills it to the database. Algorithm of traffic information summarizing is below. This algorithm provides significant decreasing of data amount in the database and increases system performance.

Traffic between internal hosts is not accounted unless port number of one of hosts (1) belongs to the list of ports, which has to be accounted for this host (let's mark this list D). This list can be specified in UIU for each host. In such case that internal host (1), whose port has to be accounted, is interpreted as external host (and appears as external host in statistics). Typical examples of such special ports are HTTP server on port 8080, HTTP proxy, tunnels.

If packets having same parameters (protocol, source and destination hosts, ports) are registered during the single traffic accumulation interval (see TRU configuration), then they accounted as a single packet, which has length equal to total length of real packets.

For better summarization each packet (TCP and UDP) undergoes resetting of port numbers to 65535 before the summarization. General idea of this resetting – do not keep non-informational port numbers; these are ports of the client host, which has opened the connection. Usually these are ports greater than 1024. Exact rule of resetting is represented with a table:

external host internal host	port<=1024	port>1024	port belongs Dext
port<=1024	✕	external	-
port>1024	internal	-	internal
port belongs Dint	-	external	✕

Port number of internal host is reset in two of seven possible cases (connection made from internal host to external), port number of external host is reset in other two cases (connection made from external host to internal), in rest cases port numbers are not reset as they can not be certainly considered non-informational. Dint – ports of internal host, greater than 1024, which are listed as accountable; Dext – ports of (pseudo) external host (only for case of traffic between two internal hosts), which are listed as accountable.

Resetting of ports number can be completely turned off for individual hosts through UIU.

It is necessary to note that if strength of the computer where TRU is running is not enough for registration and processing of all packets, then information about part of packets will be lost. In this case number of missed packets will be output to the log.

4.3. User Interface Unit

UIU is implemented as web application using JSP technology. UIU requires JSP container (JSP 1.1 and Java Servlet 2.3 compatible).

Most of user interface is clear and easy to use and doesn’t require special manual. UI guide supplied with documentation contains notes for usage of few little complicated parts of interface.

ATTENTION: TRU and ERU units must be restarted after settings of system have been changed or some entities (hosts, users, etc) have been created, changed or deleted.

4.4. Platforms Summary

ERU/S and UIU are platform independent; they require JRE 1.3.0 or compatible.

ERU/C requires one of the following:
- JRE 1.3.0 or compatible (platform independent)
- x86 MS Windows 32bit (native command-line client may be compiled for other platform upon additional request).

TRU is platform independent; it requires JRE 1.3.0 or compatible. Since TRU uses JPcap, it also requires one of following:
- x86/Alpha MS Windows 32bit with WinPcap+JPcap installed
- Any Unix with LibPcap+JPcap installed.

5. Installation Guide

5.1. Package Structure and Contents

Cachalot System is supplied as an archive of the following structure:
- bin – batch scripts for running/stopping system units
  - internal: environment.cmd, environment.sh
  - installation: cachalot.cmd, cachalot.sh
  - ERU: eru_client.cmd, eru_client.sh, eru_jclient.cmd, eru_jclient.sh, eru_start.cmd, eru_start.sh, eru_stop.cmd, eru_stop.sh
  - TRU: tru_listdevices.cmd, tru_listdevices.sh, tru_start.cmd, tru_start.sh, tru_stop.cmd, tru_stop.sh
  - UIU: uiu_start.cmd, uiu_start.sh, uiu_stop.cmd, uiu_stop.sh.
- conf – configuration files:
  - erus.conf
  - rdgcswic.conf
  - tru.conf
  - uiu.conf.
- doc – documentation:
  - license.en.html, license.ru.html
  - manual.en.pdf, manual.ru.pdf

  - install.linux.en.html, install.linux.ru.html, install.winnt.en.html, install.winnt.ru.html
  - faq.en.html, faq.ru.html
  - uinotes.en.html, uinotes.ru.html
  - changes.ru.html, changes.en.html
  - upgrade.en.html, upgrade.ru.html.
- system
  - client – client-side executable code
  - database – database sql scripts:
    - create_general.tmpl.sql, create_operations.tmpl.sql
    - create_general.mssql65-8.sql, create_operations.mssql65-8.sql
    - create_general.pgsql, create_operations.pgsql.sql
    - drop_general.sql, drop_operations.sql
  - server – server-side executable code
  - install.cmd, install.sh.

Legal notice. jpcap.dll, libjpcap.so and net.sourceforge-jpcap-0.01.13.jar are distributed under terms of Mozilla Public License version 1.1. Source code of supplied executable code is available at http://jpcap.sourceforge.net/ or http://www.rathedg.com/products/cachalot/download/net.sourceforge.jpcap-0.01.13.zip under terms of Mozilla Public License version 1.1.

This product includes software developed by the Apache Software Foundation http://www.apache.org/.

5.2. Installation Sequence

Optimal sequence of installation and configuration:

- create database (see «Database» chapter)
- configure and run UIU (see «User Interface Unit» chapter)
- register through UIU hosts and users, set the prices
- configure and run ERU/S (see «Event Registration Unit» chapter)
- setup ERU client on client hosts (see «Event Registration Unit» chapter), configure clients
- configure and run TRU.

5.3. Database

To setup database use create*.sql files located in db folder of system package. Main files are create*.tmpl.sql, they are templates containing meta-names instead of data type names. System has been tested with MSSQLServer 6.5, MSSQLServer 2000 and PostgreSQL 7.2.3 and thus there are prepared sql scripts[1] for these databases: create_*.mssql65-8.sql[2] and create_*.pgsql.sql. If you wish to use system with another database, say X, you must create your own create_*.x.sql scripts by replacing meta-names with appropriate values of data types in create_*.tmpl.sql files:

pattern	**value**
%VARCHAR%	variable length string
%BYTE%	integer number, 2^{8} values
%SHORT%	integer number, 2^{16} values
%FLOAT%	real number, any one can be used
%DATETIME %	date and time
%INTEGER%	integer number, 2^{32} values
%CHAR%	character (single byte)
%BIT%	boolean or bit (or any integer number if none of boolean and bit is supported by database)
%ALLOWSNULL%	a string which tells database that field can contain NULL

---

[1] Here script term is used in sense of a sequence of queries.

[2] Note that MSSQL Query Analyser can not be used to execute the script file completely because of CREATE VIEW-related restriction

	(e.g. '*NULL*' string for MSSQL)
%TRUE%	'true' constant (typically '*true*' or '*1*', depending on %BIT%)
%FALSE%	'false' constant (typically '*false*' or '*0*', depending on %BIT%)

You may choose data types with different value ranges or types having the internal database representation different from required[3] (acting at your own risk), but consider the following:

- choosing the bigger values range you will not achieve bigger values as this will not change the way system works with these values
- you may choose smaller ranges (e.g. %BYTE% in place of %SHORT%) to reduce the size of the database, but be aware that some ODBC and JDBC drivers can fail to convert data types that differ from the requested by application (it can be deliberate behavior of a driver as well as an error in driver), thus it can result in error (e.g. when application will try to get a %SHORT% from resultset while it contains %BYTE%).

To setup database you have to sequentially execute two scripts: create_general.x.sql, create_operations.x.sql. Omitting the execution of second script you will not install 'Operations' plug-in. There are also drop*.sql scripts, they may be useful to drop all the Cachalot's data and tables (order of their execution is reverse to the order of creation).

To make your database accessible to system you must have appropriate JDBC driver, know its name and syntax of JDBC URL for that driver. Below is a list of some drivers:

database	**driver**	**URL**
MSSQLServer 7 MSSQLServer 2000	MS JDBC Driver for MSSQL (successfully tested with MSSQLServer 2000)	http://www.microsoft.com/sql/downloads/2000/jdbc.asp
MSSQLServer 6-7 Oracle Informix4	BEA jDriver 5.1.0 (successfully tested with MSSQLServer 6.5)	http://commerce.beasys.com/downloads/weblogic_server.jsp#wlsjdbc http://www.beasys.com/products/weblogic/drivers.shtml
PostgreSQL	~~PostgreSQL JDBC~~ 7.2 (failed to do some required data type conversions, seems to be a bug; 7.3 beta 3 – the same)	http://jdbc.postgresql.org/
PostgreSQL	PostgreSQL JDBC 7.3 (successfully tested with PostgreSQL 7.2.3)	http://jdbc.postgresql.org/
PostgreSQL	jxDBCon-doc-0.9z (successfully tested with PostgreSQL 7.2.3)	http://sourceforge.net/projects/jxdbcon
Oracle	Oracle JDBC	http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html
Others		http://industry.java.sun.com/products/jdbc/drivers

If there's no JDBC driver available for your database or if it is too buggy, you may use JdbcOdbc driver, which is part of JRE. A popular database has at least one ODBC driver. It makes it possible to work from JRE with ODBC datasource. To use JdbcOdbc driver you have to create a DSN. For this, use

'sun.jdbc.odbc.JdbcOdbcDriver' as name of driver and

'jdbc:odbc:<DSN>;UID=<username>;PWD=<password>;' as JDBC URL (without angle brackets).

---

[3] If you are making these changes you are acting at your own risk and RatheDG is free to decide whether to provide technical support in such cases.

5.4. Event Registration Unit

5.4.S. Installation of ERU/S

Installation of ERU/S is in its configuration. Below is list of configuration file parameters:

parameter	**description**	**possible values**
port	TCP/IP port to listen to	*1..65535*
address	address to bind to	host address
connections	maximum length of incoming connections queue[4]	*1..65535*
threads	number of threads in pool[5]	*1..65535*
shutdown.password	password for server shutdown	any string
log.*	Log configuration. See appendix A.	
socket.timeout	socket i/o timeout, ms, *0* is infinity	*0..2147483647*
socket.linger	socket linger time, ms	*0..2147483647*
socket.nodelay	NODELAY socket option	*true*, *false*
password	password for client authentication	not empty string
db.*	Database connection configuration. See appendix B.	
cachalot.keyfile	Full name of a keyfile. Example: "*c:\cachalot\key.keyfile*"	filename

Configuration requires some understanding of TCP/IP and JDBC. Below are the most important tips for choosing right values for some of configuration parameters:

parameter	**tips**
connections	A number specifies maximum simultaneous connections that may be established with the server. If maximum is reached client will receive an immediate "connection refused" error. This number is directly proportional to expected maximum number of simultaneous client requests (i.e. events). It is recommended to not to save resources at the expense connections queue. If clients often fail due to "connection refused" error, it is highly recommended to increase number of connections.
threads	A number specifies maximum simultaneous requests that may be handled without allocating expensive resource – thread. If maximum is reached, a handler will create new threads. Thus, request will be processed slower than if maximum is not reached. I.e. increasing number of threads in pool speeds up handling of requests for expense of memory usage. This number is directly proportional to the expected typical number of simultaneous requests and to the time of request handling. When server creates a new thread (running out of pool), it writes message to log file; thus it is possible to analyze whether number of threads needs to be increased or not. It makes no sense to set number of threads more than number of connections specified in 'connections' parameter.

---

[4] socket backlog

[5] i.e. preallocated and reusable

socket.timeout	It is not recommended to use the infinite timeout. Timeout value depends only on speed of the network (which depends on network load) and server load. In case of unloaded local network and unloaded server, the value of *1000* should be enough. For slow or overloaded network or server the recommended minimum value is *3000*.
socket.linger	Same as for timeout since ERU/S doesn't sends much data.
socket.nodelay	For unloaded local network it is better to set '*true*'. In case of usual network overloads or in case of slow network it is better to set '*false*'.

If you hesitate about some parameters of your configuration or use less than safe values it is recommended to observe server and client logs for errors to get aware whether the configuration needs to be improved or not.

After you have made all necessary changes in the supplied configuration file with your configuration you can run ERU/S from command line using sample start/shutdown script (edit variables in beginning of each script). If you wish to run ERU/S as Windows NT service you have to use third-party products that allow running usual applications as services (e.g. srvany).

5.4.C. Installation of ERU/C

5.4.C.1. Command-line Clients

Actually, Java command-line ERU/C implementation and native command-line ERU/C implementation do not require installation at all. To get detailed information about available configuration parameters, refer to the description of 'server.*' parameters of Windows Integrated Client configuration, parameters are the same.

5.4.C.2. Windows Integrated Client

Use installer rdgcswic.exe for Windows integrated client installation. Installer has command line interface. It has four functions:

first parameter	**function**	**command line**
view	View current configuration.	rdgcswic.exe view
install	Installation (installation over installed earlier version is possible). Configuration file name must be specified as second parameter.	rdgcswic.exe install rdgcswic.conf
configure	Configuration change. Configuration file name must be specified as second parameter.	rdgcswic.exe configure rdgcswic.conf
uninstall	Uninstallation.	rdgcswic.exe uninstall

When installing, rdgcswic.dll must be in current folder. Configuration file contains configuration parameters described below. System package contains sample configuration file – rdgcswic.conf. After installing and configuring it is necessary to reboot the computer for changes to take effect.

parameter	**description**	**possible values**
log.normal	A name of the log file where the events information is stored. If the path is omitted, the file will be placed to %SystemRoot%\system32\.	name of file
log.error	A name of the log file where the errors information is stored. If the path is omitted, the file will be placed to %SystemRoot%\system32\.	name of file
server.name	Host name locating ERU/S.	address of host
server.port	Port number locating ERU/S.	*1..65535*
server.password	Password to authenticate in ERU/S.	any string

server.timeout	Socket i/o timeout, ms, *0* is infinity.	*1..65535*
server.attempts	A number of attempts to send an event before giving up in case of a failure.	*1..65535*
neterror.allow	Comma-separated list of user or/and group names, which are allowed to login if error has occurred during sending the logon event. Other users (not listed and not belonging to groups listed) will be denied to logon. Leave empty to allow any user to logon in case of error sending an event (e.g. unplugged network cable). Example: "*administrator, mydomain\Admin, mydomain\Managers*"	comma-separated list of user or/and group names

Below are the most important tips for choosing right values of some parameters:

server.timeout	Timeout depends only on speed of the network (which depends on its load) and server load. In case of unloaded local network and unloaded server, the value of *1000* should be enough. For slow network or often network or server overloads, the recommended minimum value is *3000*.
server.attempts	A number of maximum attempts to send an event to the server. Recommended minimum value is *3*.

If you hesitate about some parameters of your configuration or use less than safe values it is recommended to observe server and client logs for errors to get aware whether the configuration needs to be improved or not.

5.5. Traffic Registration Unit

TRU requires Sourceforge JPcap 0.01.13 library to be installed.

To properly install TRU you just need to configure it. Below is an available configuration file parameters:

parameter	**description**	**possible values**
device.name	A name of the network adapter device. Execute tru_listdevices.cmd or tru_listdevices.sh to see the names of available devices. IP address (numeric) appropriate to adapter can be specified instead of name of device. Example: '*\Device\Packet_NdisWan4*', '*eth0*', '192.168.0.1'.	device name
device.promiscuous	Defines whether to open device in promiscuous mode, or not.	*true* or *false*
interval	A number of seconds before accumulated and summarized traffic information will be flushed to the database. Specify *0* (zero value) for immediate flushing of the information about each packet to the database.	*0..65535*
log.*	Log configuration. See appendix A.	
db.*	Database connection configuration. See appendix B.	

Tips for choosing 'interval' parameter: bigger interval means less precise information about session traffic (because traffic information will possibly be filled to database after session traffic has been finally calculated) but makes more robust the work of whole system because significantly decreases amount of data in database (this can be seen well in change of duration of UIU reports generation).

After you have filled supplied configuration file with your configuration you may run TRU from command line using supplied sample start/shutdown script (edit variables set in beginning of script). If you wish run TRU as Windows NT service you have to use third-party products that allow running usual applications as services (e.g. srvany).

5.6. User Interface Unit

To install UIU configure it and deploy to the JSP container.

UIU load its configuration in two ways – from the configuration file or from JNDI.

5.6.1. Configuring via Configuration File

To specify the configuration file, set a JVM variable UIU_CONF; its value must be the absolute path of the configuration file (e.g., UIU_CONF=*c:\cachalot\uiu.conf)*. This file contains parameters of UIU configuration.

Some containers provide the ability to specify JVM variable for container. For other containers '-D' parameter must be added to java command line. To set JVM variable, specify '-Dname=value' in java command line, i.e. '-DUIU_CONF=filename'. For example 'java –DUIU_CONF=c:\cachalot\uiu.conf mycontainerclassname'.

5.6.2. Configuring via JNDI

At first be sure that UIU_CONF environment variable is undefined or contains empty value before setting up JNDI configuration for UIU.

There are many ways to specify parameter values via JNDI; they depend on particular container. Most widely used ones are represented below:

- container user interface for JNDI
- specifying parameters in container configuration file
- specifying parameters in web.xml (web-application configuration file).

Refer to the container documentation for the detailed information about first two options.

Below is the description of a third option.

xml-file WEB-INF/web.xml of archive uiu/uiu.war may contain `<env-entry>` sections of the following structure:

```
<env-entry>
     <env-entry-name>parameter name</env-entry-name>
     <env-entry-value>parameter value</env-entry-value>
     <env-entry-type>parameter type</env-entry-type>
</env-entry>
```

Parameter type must be '*java.lang.String*'. Thus, to use this method of UIU configuration, specify such section for each parameter of configuration. Example:

```
<env-entry>
     <env-entry-name>cachalot.log.console</env-entry-name>
     <env-entry-value>off</env-entry-value>
     <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

For editing the WEB-INF/web.xml file web-application archive file uiu/uiu.war has to be unpacked and then packed back (using unzip/zip or jar).

5.6.3. Configuration Parameters

parameter	**description**	**possible values**
cachalot.db.*	Database connection configuration. See appendix B (prefix each parameter with 'cachalot.').	
cachalot.db.capacity	A number of database connections in pool. This value specifies how many simultaneous requests to database can be executed without allocating expensive resource – database connection. If maximum is reached, a handler will create new connection. Thus, request will be processed slower than if maximum is not reached. I.e. increasing number of connections in pool speeds up handling of requests for expense of memory and other resources usage. This number is directly proportional to expected typical number of simultaneous requests and a	*0..65535*

	time of request handling. When server creates a connection running out of pool, it writes a message to the log file; thus, it can be analyzed by examining the log whether to increase number of connections, or not. Set to *0* (zero) to completely disable pooling of database connections.	
cachalot.db.bigInt	Pattern of SQL function of your database, which performs converting of an integer value into big integer (to make the sum of integers produce correct value for big numbers). '%0%' must be specified in the place of converted number argument. If you need to use '%' character anywhere in function pattern, use double '%' character, i.e. '%%' will be transformed to '%'. For example, for MSSQL the pattern is '*CONVERT(NUMERIC,%0%)*', for PostgreSQL it is '*%0%*'.	string containing *'%0%'*
cachalot.db.dateDiff	Pattern of SQL function of your database, which performs calculating of the difference in milliseconds between two DATETIME (TIMESTAMP) values. '%0%' and '%1%' must be specified in place of DATETIME arguments (%0% is earlier date). If you need to use '%' character anywhere in function pattern, use double '%' character, i.e. '%%' will be transformed to '%'. For example, for MSSQL the pattern is '*DATEDIFF(ms,%0%,%1%)*', for PostgreSQL it is '*date_part('epoch',%1%-%0%)*1000*'.	string containing *'%0%'* and *'%1%'*
cachalot.log.events	Event codes separated with comma. Specifies what events to audit in log file. Possible codes are: - 'auth.success','auth.failure' – successful/ failed login (on UIU authentication page) - 'logout' – user logout (in UIU) - 'entity' – entity (e.g. user) management (creating, removing, update) - 'system' – change of system settings (e.g. price) - 'data.removed' – use of 'remove data' page. Example: '*entity, system, data.removed*'.	any of codes of events separated with comma
cachalot.log.*	Log configuration. See appendix A (prefix each parameter with 'cachalot.').	
cachalot.keyfile	Absolute path to the keyfile. Example: "*c:\cachalot\key.keyfile*"	filename

After database was created, there is one default user registered in UIU with name '*admin*' and password.

5.6.4. Deployment

The process of deployment depends on particular JSP container. System has been tested with Apache Tomcat 4.0 and 4.1.18 (http://jakarta.apache.org/tomcat/) and Jetty 4.2.5 (http://jetty.mortbay.com/jetty/index.html).

Below is the brief description of a deployment process for these containers. If you deploy to another server, refer to its documentation.

Tomcat 4.x Deployment

If you configure UIU through the configuration file, file set an environment variable CATALINA_OPTS to '-DUIU_CONF=path to configuration file' value.

You can deploy UIU to Tomcat 4.x by several ways, including the following:

- (Simply) By copying uiu/uiu.war into %CATALINA_HOME%/webapps. UIU will be accessible at http://yourhost:port/uiu/ URL. If you with to access it at server root path, rename webapps/ROOT folder to ROOT.orig and rename webapps/uiu.war to ROOT.war. After that, UIU will be accessible at http://yourhost:port/ URL.
- By adding the following <Context> section into <Host> section of %CATALINA_HOME%/conf/server.xml file:

  ```
  <Context
       path="/cachalot"
       docBase="c:/RatheDG/cachalot/uiu.war"
  />
  ```

  where *path* is a part of UIU URL relatively to your server document root, *docBase* is an absolute path to uiu.war file. UIU will be accessible at http://yourhost:port/cachalot/ URL.

Your JDBC may be copied to %CATALINA_HOME%/lib (for Tomcat 4.0.x) or %CATALINA_HOME%/shared/lib (for Tomcat 4.1.x). It will automatically become available for UIU.

For more detailed information on deployment refer to the corresponding documentation located at

%CATALINA_HOME%/webapps/tomcat-docs/config/context.html or http://jakarta.apache.org/tomcat/tomcat-4.1-doc/index.html

Jetty 4.2.5 Deployment

Current version of Jetty doesn't support JNDI configuration. You can deploy UIU to Jetty 4.2.x in several ways, including the following:

- (Simply) By copying uiu/uiu.war to %JETTY_HOME%/webapps folder. UIU will be accessible at http://yourhost:port/uiu/ URL. If you with to access it at server root path, rename webapps/root folder to root.orig and rename webapps/uiu.war to root.war. After that UIU will be accessible at http://yourhost:port/ URL.
- By adding <Call> section to <Configure> section of %CATALINA_HOME%/etc/jetty.xml file:

  ```
  <Call name="addWebApplication">
       <Arg>/cachalot</Arg>
       <Arg>c:/RatheDG/cachalot/uiu.war</Arg>
       <Set name="extractWAR">false</Set>
  </Call>
  ```

  where first *Arg* element is the part of UIU URL relatively to your server document root, second *Arg* element is the absolute path to uiu.war file. UIU will be accessible at http://yourhost:port/cachalot/ URL.

Your JDBC may be copied to %JETTY_HOME%/ext. It will automatically become available for UIU.

After you finish the configuration, you may run Jetty by changing the current folder to %JETTY_HOME% and executing the following command:

```
java –DUIU_CONF=configuration_file -jar start.jar etc/jetty.xml
```

For more detailed information on deployment refer to the corresponding documentation at

http://jetty.mortbay.com/jetty/index.html

6. Security

The environment is supposed to be the following:

- unauthorized person can not access server files
- unauthorized person can not access database
- unauthorized person can not access ERU client password kept on client-side
- unauthorized person can not get ERU client password by sniffing the network connections
- unauthorized person can not modify client host IP address
- UIU user password is known to the user only and nobody else.

As long as these conditions are followed the system stays absolutely secure, and unauthorized person is unable to: change system settings, modify data, supply wrong incoming data, etc. UIU authentication is brute force resistant, however it is desirable to observe logs periodically to get aware about the brute force attacks attempts.

Appendices

Appendix A. Log configuration

Log configuration parameters:

log.filename	A name of log file; it may contain path; leave empty or commented to disable logging to file. Example: ‘*c:\logs\eru.log*’ or ‘*/opt/cachalot/logs/eru.log*’.	filename
log.period	A period of log file rotation.	*hour, day, month*
log.prefix	Specifies whether to add the date before the log file name.	*true, false*
log.usefileserparator	Specifies whether to use file separator in datestamp in the log file name instead of ‘.’, doesn’t matter if log.prefix=false.	*true, false*
log.console	Specifies whether to duplicate log output to the console	*on, off*

Tips for choosing values for some parameters:

log.period	Smaller period lightens regular review of log files and searching for the particular place in log file for very little expense of resources.
log.usefileserparator	If ‘*true*’, log files will be automatically created in tree-like structure of folders, e.g. year/month/day for ‘*hour*’ period of rotation.
log.console	Useful for debug and analysis purposes. It is recommended to turn off when running as a background process.

Appendix B. Database connection configuration

Database connection parameters configuration:

db.driver	JDBC database driver classname; class must be accessible from CLASSPATH. Example: ‘*weblogic.jdbc.mssqlserver4.Driver*’ or ‘*org.sourceforge.jxdbcon.JXDBConDriver*’.	classname
db.url	JDBC url to database, it must be suitable for specified driver and must locate the database. Refer to your JDBC driver documentation for URL syntax details. Example: ‘jdbc:weblogic:mssqlserver4:localhost:1433;user=cachalot;password=cachalotpass;’ or ‘*jdbc:postgresql:net//pguser:pgpass@myhost:5438/cachalotdb*’.	JDBC URL
db.transaction	Database transaction isolation level.	*none,* *read_uncommitted,* *read_committed,* *repeatable_read,* *serializable*

Below is the explanation of ‘db.transaction’ parameter.

Usually changing level of transactions isolation leads to a different proportion of performance and risk of data inconsistency. It is obvious that more performance leads to more risk and less risk leads to less performance. It is supposed that there’s no extra important data in Cachalot System. The maximum loss is missing the events in case of some hardware or software failure; it doesn’t make considerable damage.

Current implementation of Cachalot System works in auto-commit mode and it is supposed that transaction isolation level is “none”, i.e. no transactions. But some database engines do not support this level (e.g. MSSQL). There is a configuration parameter in each system unit, which must be configured by administrator accordingly to database capabilities and/or personal preferences.

A set of supported transaction isolation levels is specific to particular database. Below are technical descriptions of five widely used transactions isolation levels in order of increasing the safety and decreasing the performance. Administrator has to specify one of them in configuration of each unit.

- isolation level "none"

  Indicates that transactions are not supported.

- isolation level "read_uncommitted"

  Dirty reads, non-repeatable reads and phantom reads can occur. This level allows a row changed by one transaction to be read by another transaction before any changes in that row have been committed (a "dirty read"). If any of the changes are rolled back, the second transaction will have retrieved an invalid row.

- isolation level "read_committed"

  Dirty reads are prevented; non-repeatable reads and phantom reads can occur. This level only prohibits a transaction from reading a row with uncommitted changes in it.

- isolation level "repeatable_read"

  Dirty reads and non-repeatable reads are prevented; phantom reads can occur. This level prohibits a transaction from reading a row with uncommitted changes in it, and it also prohibits the situation where one transaction reads a row, a second transaction alters the row, and the first transaction rereads the row, getting different values the second time (a "non-repeatable read").

- isolation level "serializable"

  Dirty reads, non-repeatable reads and phantom reads are prevented. This level includes the prohibitions in TRANSACTION_REPEATABLE_READ and further prohibits the situation where one transaction reads all rows that satisfy a WHERE condition, a second transaction inserts a row that satisfies that WHERE condition, and the first transaction rereads for the same condition, retrieving the additional "phantom" row in the second read.

http://www.rathedg.com/

support@rathedg.com