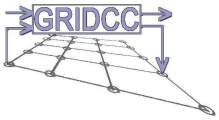

Grid Enabled Remote Instrumentation with
Distributed Control and Computation

Logging Collector 3.0
User Guide

Version 1.0
10/11/2005

Version: 1.0
Date: 10/11/2005
Authors: E. Frizziero
CI identifier

INFN Istituto Nazionale di Fisica Nucleare
Laboratori Nazionali di Legnaro

Information Society
Infrastructures

Revision History

Date	Version	Description	Author
October 04, 2005	1.0	Document creation	E. Frizziero

CI Record

Field	Description
CI Identifier	
Description	
Submission Date	
Submitted By	
Components	
Dependencies/Related	
External Identifier	
Point of Contact	
Comments	
Physical Location	

Table of Contents

1 Overview 6

2 Logging Collector architecture 7
2.1 Input section 7
2.2 Output section 7
2.2.1 SocketAppender 7
2.2.2 JMSAppender 8
2.2.3 DBAppender 8
2.3 Command interface section 8
2.3.1 Commands 8

3 Logging Collector core 9
3.1 Logging Collector state machine 9
3.2 Logging Collector working commands 9
3.3 Description of the transitions 10

4 Logging Collector Appender core 11
4.1 Appender state machine 11
4.2 Logging Collector Appender commands 11
4.3 Description of the transitions 12
4.4 Appender working modes 13
4.4.1 NO_QUEUE working mode 13
4.4.2 QUEUE_WITH_LOST_MSG working mode 13
4.4.3 QUEUE_WITH_NO_LOST_MSG working mode 13
4.4.4 Example 13

5 Logging Collector web surfing 14

6 Logging Collector command interface 15
6.1 Command web page 15
6.1.1 Logging Collector working commands 15
6.1.2 Refresh button and Message from Collector 15
6.1.3 Logging Collector Appender commands 15
6.2 Command web service 16
6.2.1 Available Web Service commands 16

7 Logging Collector configuration 17
7.1 Input configuration section 18
7.1.1 TCP input 18
7.1.2 TCP XML input 18
7.2 Output configuration section 18
7.2.1 SocketAppender output 18
7.2.2 DBAppender output 19
7.2.3 JMSAppender output 19

8 Logging Collector functioning monitor 20
8.1 Description of the monitoring web page fields 20
8.1.1 Input section 20
8.1.2 Output section 21

9 How applications send logging information to Logging Collector 22
9.1 Java application 22
9.2 Non-Java application 22

10 How to display collected logging information 24

11 Examples of the Logging Collector use 25
11.1 Logging Collettor – Chainsaw 25
11.2 Logging Collector, JMS + Chainsaw and DB 25

12 Frequently asked questions 28

13 Appendix A: Configuration file 29

14 Appendix B: Appender in detail 31
14.1 Appender structure 31
14.2 Appender working modes in detail 32
14.2.1 NO_QUEUE working mode 32
14.2.2 QUEUE_WITH_LOST_MSG working mode 33
14.2.3 QUEUE_WITH_NO_LOST_MSG working mode 34

15 Appendix C: Example.java 36

16 Appendix D: Some useful web application parameters of the Logging Collector 37

17 References 38

This is a note, e.g.: Always put originator and targetAddr in your SOAP messages when sending SOAP messages to other XDAQ applications

This is a help item, e.g: How do I restart an executive, if I lost connectio to the host

This is an orientation item,e g.: You have the choice between using a Semaphore or polling the channel. The advantage of the first is bla bla, the advantage of the second is bla bla.

This is a tip,e.g.: To query a parameter several times, use the timer in the window “Properties” of the slected application.

Logging Collector

1 Overview

The Logging Collector is a software component designed and developed to collect logging information from log4j compliant applications.
The Logging Collector allows also to store logging information in a persistent way in database and/or to distribute/publish it through a real time message system (JMS).

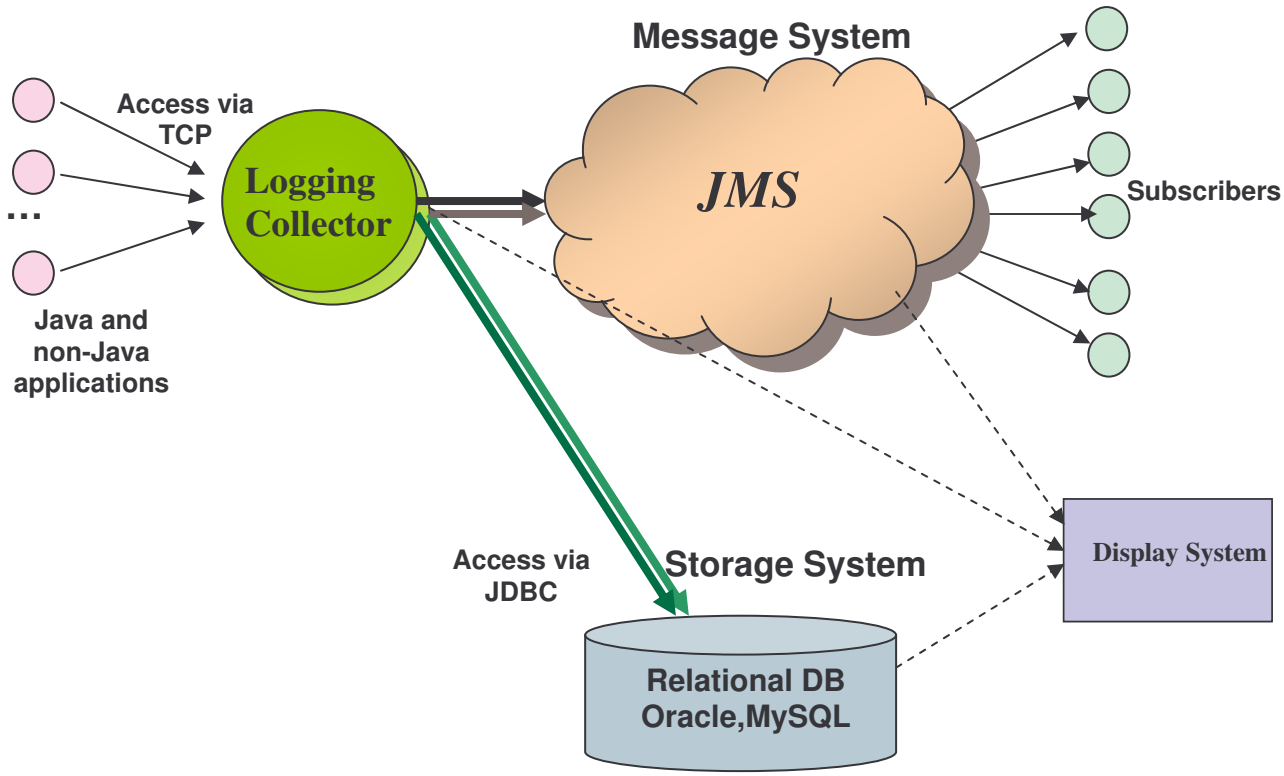

Figure 1.1 Logging Collector overview.

What's logging information?
Logging information is a message:

- written by developers in their software code
- can have a priority/level (i.e. DEBUG, INFO, WARN, ERROR, FATAL) used for debug purposes in software applications
- human readable

2 Logging Collector architecture

Logging Collector is compound by:

- an input section
- an output section
- a command interface section.

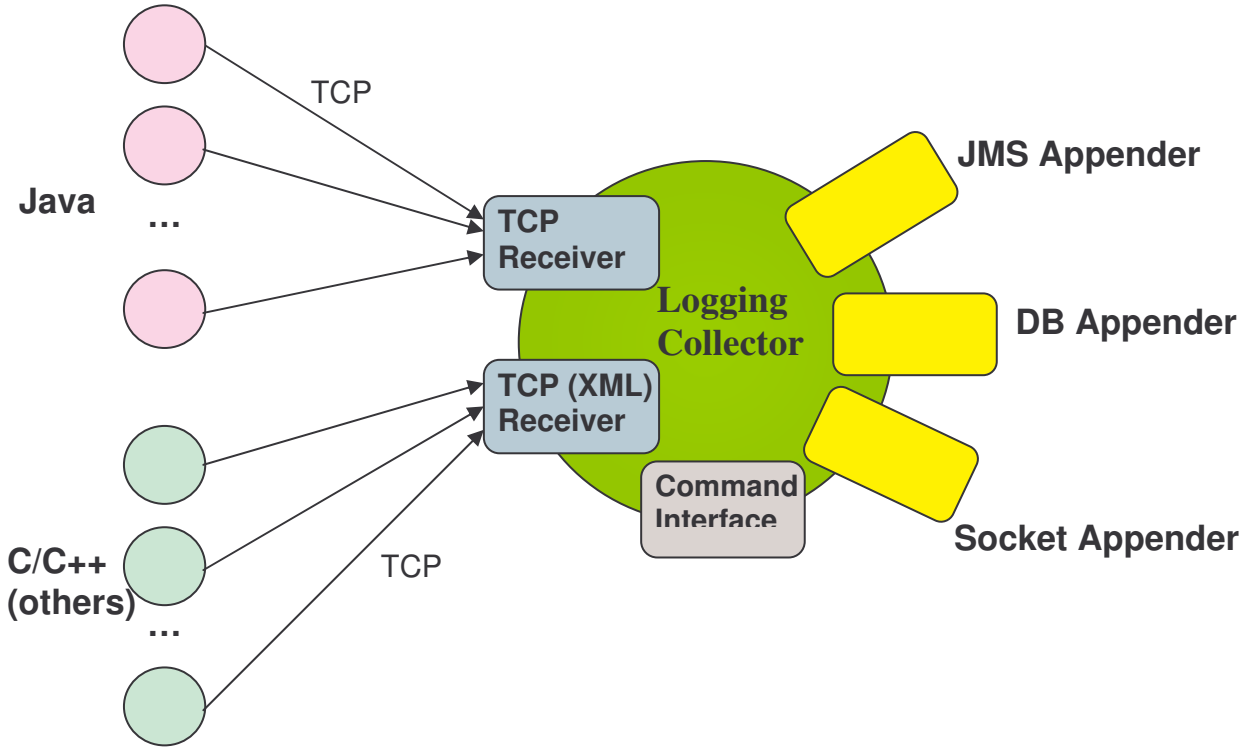

Figure 2.1 Logging Collector architecture.

2.1 Input section

The Logging Collector provides two TCP inputs. The former receives java log4j LoggingEvent objects (*org.apache.log4j.spi.LoggingEvent*) from java applications. The latter instead more generic can receive logging information in XML format (log4j compliant) from non-java applications such as C or C++ applications.

2.2 Output section

The outputs of the logging collector called **appenders** are:

- Socket Appender
- JMS Appender
- DB Appender

The collector's outputs are used to allow the persistent storage, the display and the distribution of the logging information.

2.2.1 SocketAppender

The SocketAppender allows to send the logging information to a TCP port.

The SocketAppender, correctly configured , can be used for example to send the logging information to Chaisaw [Chaisaw – site] to display it in real time. Otherwise it can be used to send the collected logging information to another Logging Collector instance to build a possible hierarchic structure of collectors.

2.2.2 JMSAppender

The JMSAppender can be used to send the logging information to a message system (JMS).
In this way, JMS's subscribers can receive the real time logging information and, if necessary, filter it by defining filter conditions at the subscription time.
By the way, two types of JMS's subscribers have been implemented and are available in the Logging Collector release (*clientCollector.zip* package):

1. **clientJMSAppender**: a generic subscriber receiving the whole logging information.
2. **clientSelectorJMSAppender**: a selector subscriber which filters the logging information according to the particular level specified (if message selectors are enabled in the JMSAppender configuration).

2.2.3 DBAppender

The DBAppender can be used to store the collected logging information to a database in order to ensure the persistent storage and so build an historical archive.
The connection database is made via JDBC.

2.3 Command interface section

The command interface section allows to send commands to the Logging Collector. The Logging Collector can receive commands both by a command web service and by a command web page. For more details see the next paragraphs.

2.3.1 Commands

Commands are of two types:

- Commands for the Logging Collector functioning

- Start
- Stop
- Configure
- Monitor

- Commands for Logging Collector appenders

- Add
- Remove
- View Message

3 Logging Collector core

3.1 Logging Collector state machine

The Logging Collector has got an internal state machine whose transitions are determined by the current state and the command sent.

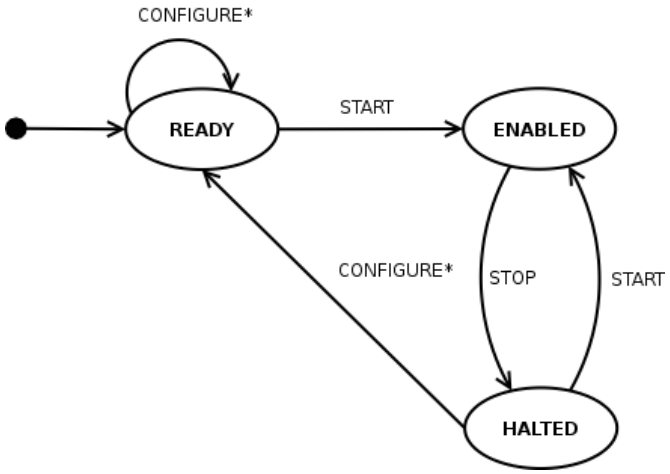

Figure 3.1 Logging Collector state machine.

State	Description
READY	Logging Collector is ready. It can be configured and/or started.
ENABLED	Logging Collector is collecting the logging information.
HALTED	Logging Collector has been stopped.

Table 3.1 Logging Collector's states.

3.2 Logging Collector working commands

Logging Collector working commands are:

- **Start**

  Start command allows to start the collection of the logging information by creating the input receivers (TCP Receiver and/or TCPXML Receiver)

- **Stop**

  Stop command allows to stop the collection of the logging information by killing the input receiver. The Logging Collector holds its current configuration and its outputs enabled.

- **Configure**

  Configure command allows to display the configuration web page.

Configure*: When you press the "*Save&Load*" button in the configuration web page, the Logging Collector is cleared (disabled both inputs and appenders). It will start from scratch with the new configuration when the start command is sent.

- **Monitor**

  Monitor command allows to display the monitoring web page.

3.3 Description of the transitions

Command	From State	To State
Start	READY	ENABLED
Description		
After this transaction the inputs of the Logging Collector are enabled and the Logging Collector is enabled to collect logging information. When the start command is sent, the Logging Collector configuration used is: • The one previously saved (in the configuration file, see appendix A). It is automatically loaded in the Logging Collector (there is no need to use the configuration web page if the configuration isn't changed from the last use of the Logging Collector); • The one just updated by the configuration web page by pressing "*Save&Load*" button.		

Table 3.2 From READY to ENABLED transition.

Command	From State	To State
Stop	ENABLED	HALTED
Description		
After this transaction the inputs of the Logging Collector are disabled whereas the appenders continue to be enabled. Logging Collector is stopped to collect the logging information.		

Table 3.3 From ENABLED to HALTED transition.

Command	From State	To State
Start	HALTED	ENABLED
Description		
After this transition the inputs are enabled. (appenders are already enabled). Logging Collector is restarted to collect logging information.		

Table 3.4 From HALTED to ENABLED transition.

Command	From State	To State
Configure*	HALTED	READY
Description		
This transition happens when the Logging Collector configuration is updated (see the note in the paragraph 3.2).		

Table 3.5 From HALTED to READY transition.

4 Logging Collector Appender core

4.1 Appender state machine

Each appender has got an internal state machine.

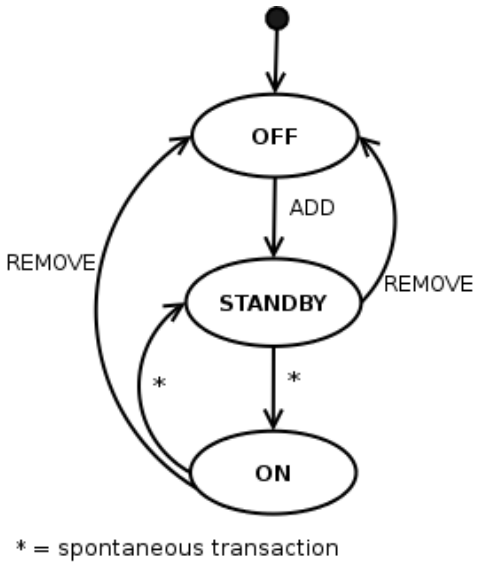

Figure 4.1 Appender state machine.

State	Description
OFF	Appender is neither enabled nor active.
STANDBY	Appender is enabled, i.e. it is tempting to connect to its output client.
ON	Appender is active, i.e. it can send the logging information to its output client.

Table 4.1 Appender states.

Output appender clients are for example:

- A **Chainsaw** for the socket appender
- A **database** for the DB appender
- A **JMS** for the JMS appender.

4.2 Logging Collector Appender commands

Each Logging Collector appender can accept the following commands:

- **Add**

The *Add* appender command allows to enable an appender of the Logging Collector.

- **Remove**

The *Remove* appender command allows to remove an appender of the Logging Collector.

- **ViewMessage**

The *ViewMessage* appender command allows to display a possible message from the appender.

4.3 Description of the transitions

Command	From State	To State
Add	OFF	STANDBY
Description		
After this transition the appender is enabled.		

Table 4.2 From OFF to STANDBY transition.

Command	From State	To State
---	STANDBY	ON
Description		
In the STANDBY state an appender is enabled but is not active, because it still must open a connection to its output client. For the SocketAppender, it means that it is tempting to create a connection to the host on the port number specified in its configuration. Once the connection has been opened the appender with a spontaneous transition enters the ON state and becomes active.		

Table 4.3 From STANDBY to ON transition.

Command	From State	To State
---	ON	STANDBY
Description		
If the appender during its functioning looses the connection to its output client, it transits spontaneously from the ON state to STANDBY state. The appender spontaneously comes back to the ON state if the connection is re-established.		

Table 4.4 From ON to STANDBY transition.

Command	From State	To State
Remove	STANDBY	OFF
Description		
The appender is disabled.		

Table 4.5 From STANDBY to OFF transition.

Command	From State	To State
REMOVE	ON	OFF
Description		
The appender is disabled.		

Table 4.6 From ON to OFF transition.

4.4 Appender working modes

Each appender can work in one of the following modes:

- **NO_QUEUE**
- **QUEUE_WITH_LOST_MSG**
- **QUEUE_WITH_NO_LOST_MSG**

For the details, see appendix B.

4.4.1 NO_QUEUE working mode

The appender in this working mode doesn't use its queue, but it sends directly the logging information to its output client. The appender doesn't handle the next logging information until it has not sent the previous one. So, in this mode there is no de-coupling between the inputs and the outputs.

When the appender is in STANDBY state the logging information is lost.

4.4.2 QUEUE_WITH_LOST_MSG working mode

The appender in this working mode inserts the logging information in its queue and handles immediately the next logging information. The *Sender* process of the appender extracts the logging information from the queue and sends it to the appender output client. When the appender queue is full and a new logging information arrives to the appender, the logging information is lost.
So in this mode there is de-coupling between the inputs and the outputs.

The appender in the STANDBY state (after the ON to STANDBY state transaction) inserts the logging information into its queue until it isn't full. When the appender queue becomes full the logging information is lost.

4.4.3 QUEUE_WITH_NO_LOST_MSG working mode

The appender in this working mode inserts the logging information in its queue and handles immediately the next logging information. The *Sender* process of the appender extracts the logging information from the queue and send it to the appender output client. When the appender queue is full and a new logging information arrives to the appender, the input client goes slow and the logging information isn't lost.
So in this mode there is de-coupling between the inputs and the outputs only when the appender queue isn't full.

The appender in the STANDBY state (after the ON to STANDBY state transaction) inserts the logging information into its queue until it isn't full. When the appender queue becomes full the logging information isn't lost, because the input client slackens its speed.

4.4.4 Example

It is possible to configure the SocketAppender in the NO_QUEUE working mode and the DBAppender with the QUEUE_WITH_NO_LOST_MSG working mode.
By these appender configurations, the rundown caused by the DBAppender is cancelled out by the DBAppender queue. In this way, the rundown of the DBAppender doesn't involve the SocketAppender functioning.

5 Logging Collector web surfing

The following page displays the Logging Collector web surfing schema.

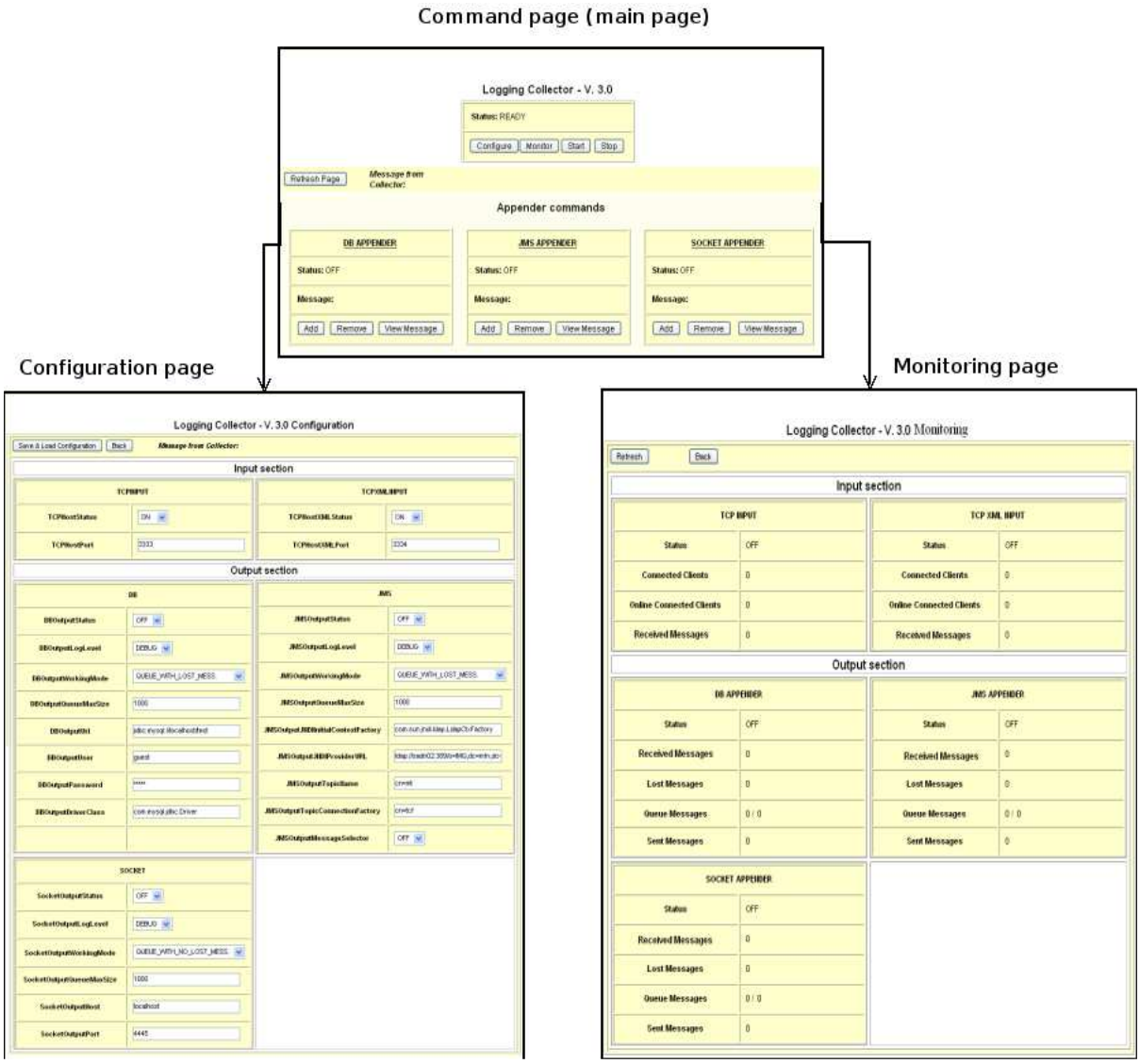

Figure 5.1 Logging Collector web surfing schema.

The url of the Logging Collector web application is:

http://<Host>:<Port>/Collector

where **<Host>** and **<Port>** are respectively the host name and the port of the web server, where the Logging Collector is installed.

In the command page, by pressing the **Configure** button you can surf to the Configuration web page, whereas by pressing the **Monitor** button you can surf to the Monitoring web page. Each web page is described in detail in the next paragraphs.

6 Logging Collector command interface

The Logging Collector can receive command both by a command web page and by a command web service.

6.1 Command web page

Figura 6.1 Command web page.

The command web page is split into three sections:

- Logging Collector working commands
- Refresh button and Message from Collector
- Logging Collector Appender commands

6.1.1 Logging Collector working commands

This section displays the current Logging Collector's state and allows to send configure, monitor, start and stop commands to the Logging Collector.

6.1.2 Refresh button and Message from Collector

This section displays messages from Logging Collector. For example if a stop command is sent when the Logging Collector is in the Ready state, a message "This command is disabled" is displayed.
In this section there is also a refresh button to get the updated collector's state and the updated appender's states. This because there could be a concurrent use of the Logging Collector for example by the command web service.

6.1.3 Logging Collector Appender commands

This section displays the state and allows the adding/removing of each Logging Collector's appender (add/remove appender command). It also allowes to view possible appender message if there is some problem to connect to the appender client (viewMessage appender command).

6.2 Command web service

The Logging Collector can be commanded by a command web service. The Logging Collector exposes a command web service called "***CollectorService***" whose WSDL can be retrieved from the url:

http://<Host>:<Port>/Collector/services/CollectorService?wsdl

<Host> and **<Port>** are respectively the host name and the port of the web server, where the Logging Collector is installed.

In the **collectorClientWebServiceCommand.zip** of the Logging Collector's release, there are some clients of the "CollectorService" that can be executed by a script.

In the case it is necessary to develop an application to send commands to the Logging Collector by the "CollectorService", it is necessary to pick up the web service WSDL, to create the stub and to call the methods of the web service (see [Axis Doc – site]).

6.2.1 Available Web Service commands

Through the web service it is possible to send only some commands to the Logging Collector.

Web Service command	Web Service command description
Start	To start the Logging Collector.
Stop	To stop the Logging Collector.
AddAppender <*AppenderName**>	To add the "*AppenderName*" appender.
RemoveAppender <*AppenderName**>	To remove the "*AppenderName*" appender.
ViewAppenderMessage <*AppenderName**>	To retrieve the message of the "*AppenderName*" appender.

Table 6.1 Logging Collector web service command.

* The <*AppenderName*> parameter can be:

- SOCKET_APPENDER
- JMS_APPENDER
- DB_APPENDER

7 Logging Collector configuration

The Logging Collector configuration is performed by the following configuration web page.

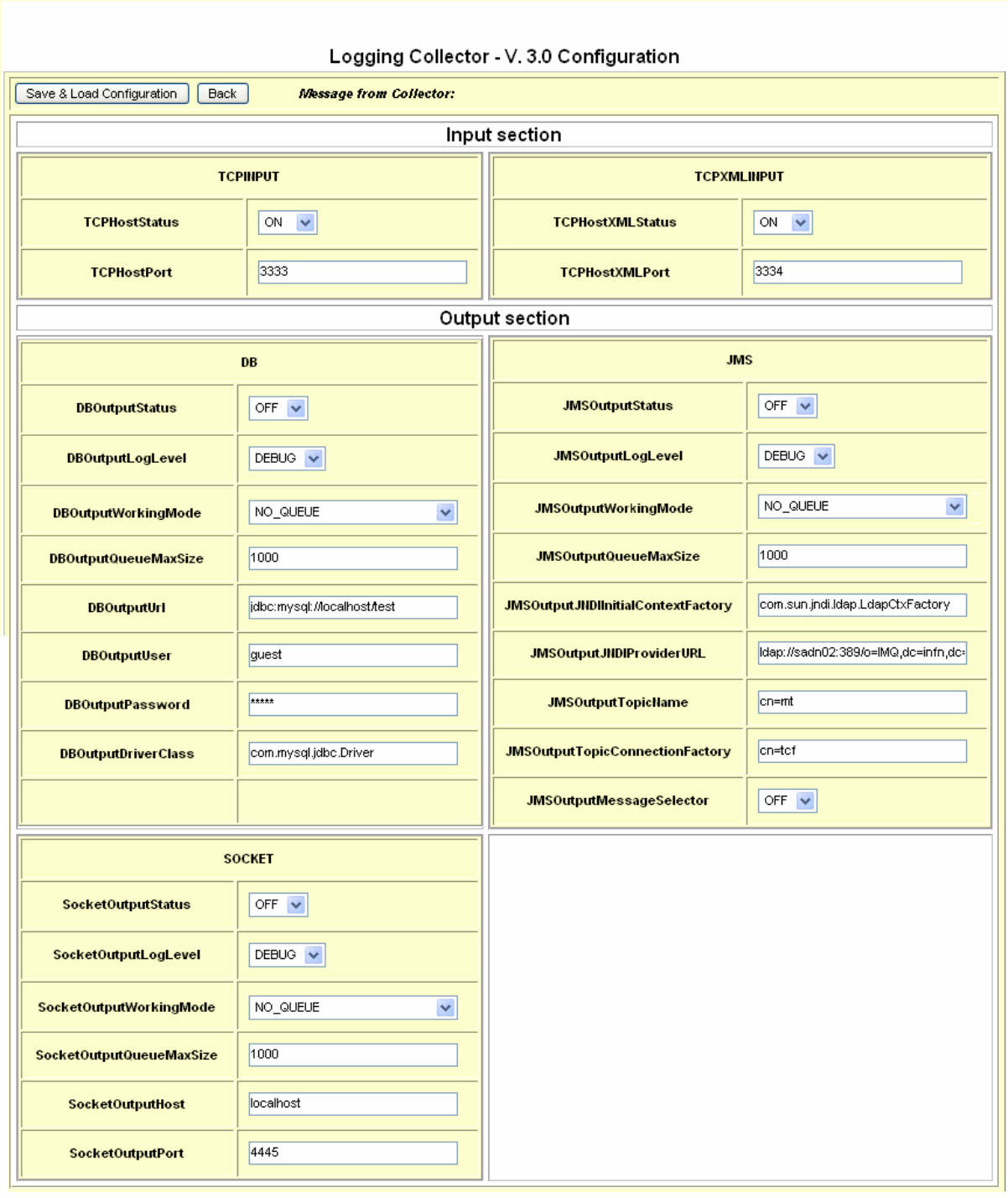

Figure 7.1 Logging Collector configuration page.

By pressing the "***Save & Load Configuration***" button you can save in a permanent way the modifications to the configuration (see Appendix A) and load the updated configuration into the Logging Collector. The new configuration becomes the actual Logging Collector configuration.
When a new Logging Collector configuration is loaded, the Logging Collector appenders belonging to the previous configuration are destroyed and new Logging Collector appenders referring to the new Logging Collector configuration are added.

The Logging Collector configuration consists of two parts:

- the input configuration section
- the output configuration section.

7.1 Input configuration section

7.1.1 TCP input

Field	Field Description
TCPHostStatus	To enable/disable the TCP input of the Logging Collector. Allowed values: ON/OFF.
TCPHostPort	The port number of the TCP input.

Table 7.1 TCP input configuration fields.

7.1.2 TCP XML input

Field	Field Description
TCPHostXMLStatus	To enable/disable the TCP XML input of the Logging Collector. Allowed values: ON/OFF.
TCPHostXMLPort	The port number of the TCP XML input.

Table 7.2 TCP XML input configuration fields.

7.2 Output configuration section

7.2.1 SocketAppender output

Field	Field Description
SocketOutputStatus	To enable/disable the socketAppender output. Allowed values: ON/OFF.
SocketOutputLogLevel	Minimum log level enabled for the socketAppender output. Allowed values: DEBUG, INFO, WARN, ERROR and FATAL (see log4j levels [Log4j Manual]).
SocketOutputWorkingMode	To select the socketAppender working mode. Allowed values: NO_QUEUE, QUEUE_WITH_LOST_MSG and QUEUE_WITH_NO_LOST_MSG.
SocketOutputQueueMaxSize	To set the number of the logging messages that can be inserted in the queue (queue size).
SocketOutputHost	IP address or hostname to which the logging information is sent.
SocketOutputPort	The port number of the host to which the logging information is sent.

Table 7.3 Socket appender configuration fields.

7.2.2 DBAppender output

Field	Field Description
DBOutputStatus	To enable/disable the DBAppender output. Allowed values: ON/OFF.
DBOutputLogLevel	Minimum log level enabled for the DBAppender output. Allowed values: DEBUG, INFO, WARN, ERROR and FATAL (see log4j levels [Log4j Manual]).
DBOutputWorkingMode	To select the DBAppender working mode. Allowed values: NO_QUEUE, QUEUE_WITH_LOST_MSG and QUEUE_WITH_NO_LOST_MSG.
DBOutputQueueMaxSize	To set the number of the logging messages that can be inserted in the queue (queue size).
DBOutputUrl	The url of the database to which to connect. Url in jdbc format: subprotocol : subname / nameDatabase.
DBOutputUser	The database user for the connection.
DBOutputPassword	The user's password.
DBOutputDriverClass	The JDBC driver class.

Table 7.4 DB appender configuration fields.

7.2.3 JMSAppender output

Field	Field Description
JMSOutputStatus	To enable/disabile the JMSAppender output. Allowed values: ON/OFF.
JMSOutputLogLevel	Minimum log level enabled for the JMSAppender output. Allowed values: DEBUG, INFO, WARN, ERROR and FATAL (see log4j levels [Log4j Manual]).
JMSOutputWorkingMode	To select the JMSAppender working mode. Allowed values: NO_QUEUE, QUEUE_WITH_LOST_MSG and QUEUE_WITH_NO_LOST_MSG.
JMSOutputQueueMaxSize	To set the number of the logging messages that can be inserted in the queue (queue size).
JMSOutputJNDIInitialContextFactory	The JNDI Initial Context Factory.
JMSOutputJNDIProviderURL	The JNDI Provider Url.
JMSOutputTopicName	The topic's name.
JMSOutputTopicConnectionFactory	The topic connection factory.
JMSOutputMessageSelector	To enable/disable the JMS message selector.

Table 7.5 JMS appender configuration fields.

You must enable the JMS message selector with the JMSOutputMessageSelector property if you use a JMS selective subscription.

8 Logging Collector functioning monitor

To monitor the Logging Collector functioning you can use the Logging Collector monitoring web page.

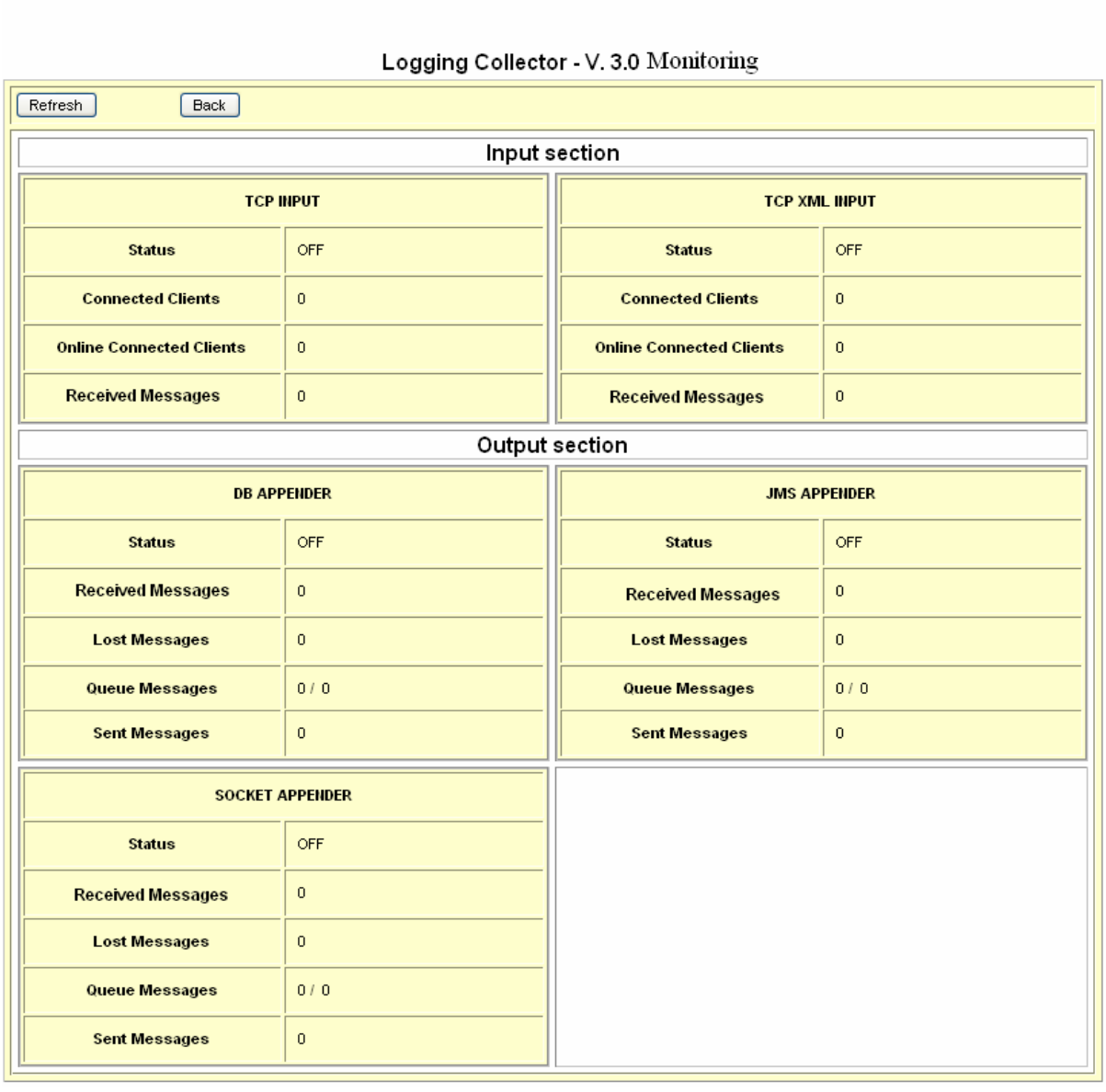

Figura 8.1 Monitoring web page.

This page is a snapshot of the Logging Collector functioning. To update the snapshot you can press the refresh button.

8.1 Description of the monitoring web page fields

The monitoring web page is split into two parts:

- Input section
- Output section

8.1.1 Input section

For each of the two Logging Collector inputs the following information is available.

Field	Field description
Status	Status of the input.
Connected Clients	Number of the connected clients (since the Logging Collector has been started).
Online Connected Clients	Number of the connected clients at the present time.
Received Messages	Number of the received messages.

Table 8.1 Input monitoring fields.

The input monitoring fields are reset to zero when the Logging Collector is stopped.

8.1.2 Output section

For each of the Logging Collector appenders the following information is available.

Field	Field description
Status	Status of the appender.
Received Messages	Number of received messages by the appender.
Lost Messages	Number of lost messages into the appender.
Queue Messages	Number of messages in the appender queue. This field is in the format of "*Number of the Queue messages / Queue size*".
Sent Messages	Number of messages sent to the output appender client.

Table 8.2 Appender monitorig fields.

The appender monitoring fields are reset to zero when the appender is removed.

The Queue Messages field isn't used (set to 0/0) if the NO_QUEUE appender working mode is selected.

The Lost Messages field is increased when a logging message is received by the appender and one of the following conditions occurs:

- The appender is in the STANDBY state (after the first OFF to STANDBY transition)
- The appender is in QUEUE_WITH_LOST_MSG working mode and its queue is full.

9 How applications send logging information to Logging Collector

Logging Collector is enabled to receive logging information through java log4j LoggingEvent objects (*org.apache.log4j.spi.LoggingEvent*) from java applications or through a stream in log4j compliant XML format from non-java applications.

9.1 Java application

In appendix C, there is the code of a dummy java application producing logging information. In the constructor is defined and configured the logger and the socketAppender used to send the logging information to the Logging Collector.

It must be used the NDC object (*org.apache.log4j.NDC*) to set the necessary information to identify unambiguously the logging information origin.

In the NDC of the dummy application is only set the IP address of the host. In some cases this information alone couldn't be enough.

LocationInfo property must be set to true value, to include the location information (method, class, line information) in the logging information too.

In some cases it is necessary to set the ReconnectionDelay property. Default values: 30 seconds.

The appender can be also defined and configured from a property file in text format or in XML format (see [Log4j Manual]).
It is advised to use the XML format.

9.2 Non-Java application

A non-java application can send the logging information by opening a TCP socket to the TCP XML input of the Logging Collector.
The Logging information must be in XML format log4j compliant.
Example:

```
<log4j:event logger="ProductLog" timestamp="181256984" level="WARN"
thread="10241">
<log4j:message>The store pipe is broken</log4j:message>
<log4j:NDC>192.165.12.128</log4j:NDC>
<log4j:throwable></log4j:throwable>
<log4j:locationInfo class="ProductLog" method="insertPipe" file="Productlog.cc"
line="134"/>
</log4j:event>
```

It must be used the NDC object (*org.apache.log4j.NDC*) to set the necessary information to identify unambiguously the logging information origin.

LocationInfo property must be set to true value, to include the location information (method, class, line information) in the logging information too..

In some case it is necessary to set the ReconnectionDelay property. Default values: 30 seconds.

10 How to display collected logging information

For a graphical display of the collected logging information a log4j application, called **Chainsaw**, is available.
Chainsaw can be download from url: http://logging.apache.org/log4j/docs/chainsaw.html.
Chainsaw has many functionalities, for example it can filter the received logging information.
For more information you can see [Chainsaw –site].

The next Logging Collector version probably will provide a new graphical interface to retrieve, to display and to filter the logging information.

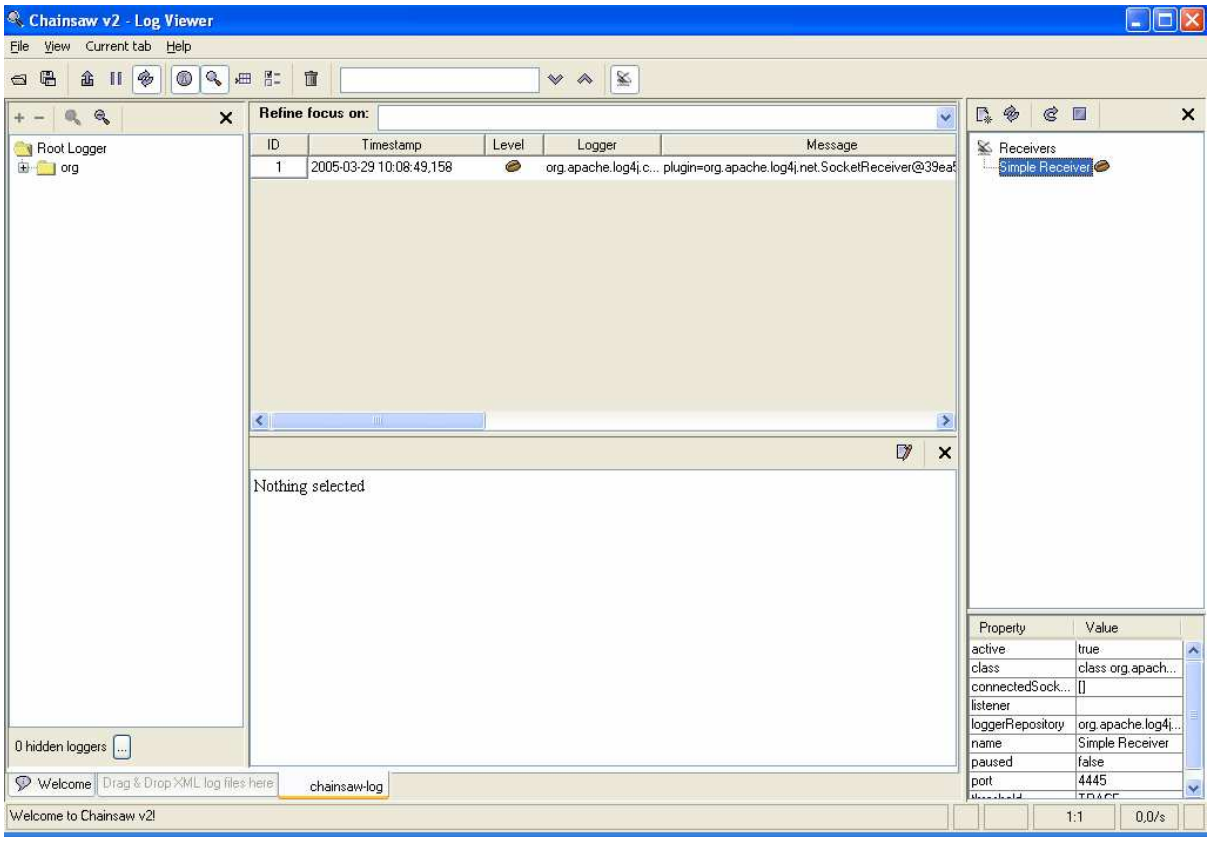

Figure 10.1 Chainsaw.

11 Examples of the Logging Collector use

Now let's see some examples how to use the Logging Collector, supposing to have already istalled all the necessary components (see [LoggingCollector – Installation Guide]).

11.1 Logging Collettor – Chainsaw

In this example, we want to display by Chainsaw the logging information produced by ProductorLog application (***collectorClient.zip*** package).

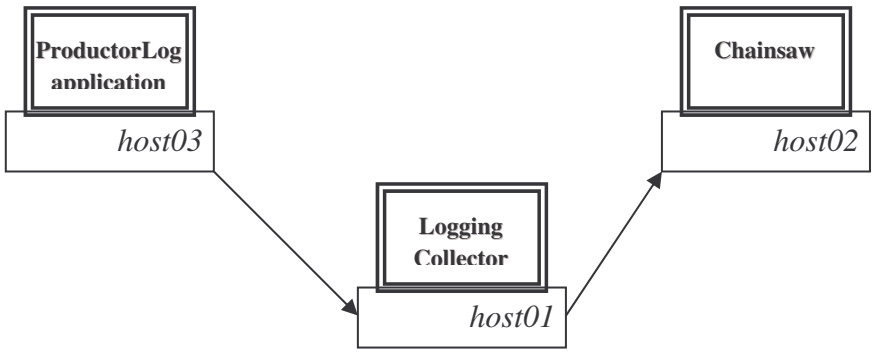

Figure 11.1 Example: Logging Collector – Chainsaw.

- **ProductorLog application**

It must be able to send the logging information to the Logging Collector, so its configuration file (*collectorClient\scripts\configFiles\ ProductorLog.config*) is:
- remoteHost=host01
- portHost=3333

- **Logging Collector**

It must be configured to send the collected logging information to Chainsaw, so its output section for the SocketAppender output is:

SocketOutputStatus = ON
SocketOutputHost = host02
SocketOutputPort = 4445

- **Chainsaw**

Following the indications of the on-line documentation [Chainsaw –site], it must be defined an SimpleReceiver on the port number 4445.

Now, you can start the ProductorLog application and in a few time in Chainsaw will be displayed the logging information.

11.2 Logging Collector, JMS + Chainsaw and DB

In figure 11.2 is shown an example of use of the Logging Collector with:
- a DB, to build an historical archive of the logging information;
- an JMS, to distribute in real time logging information to JMS's subscribers;
- Chainsaw, for a graphical display of the logging information.

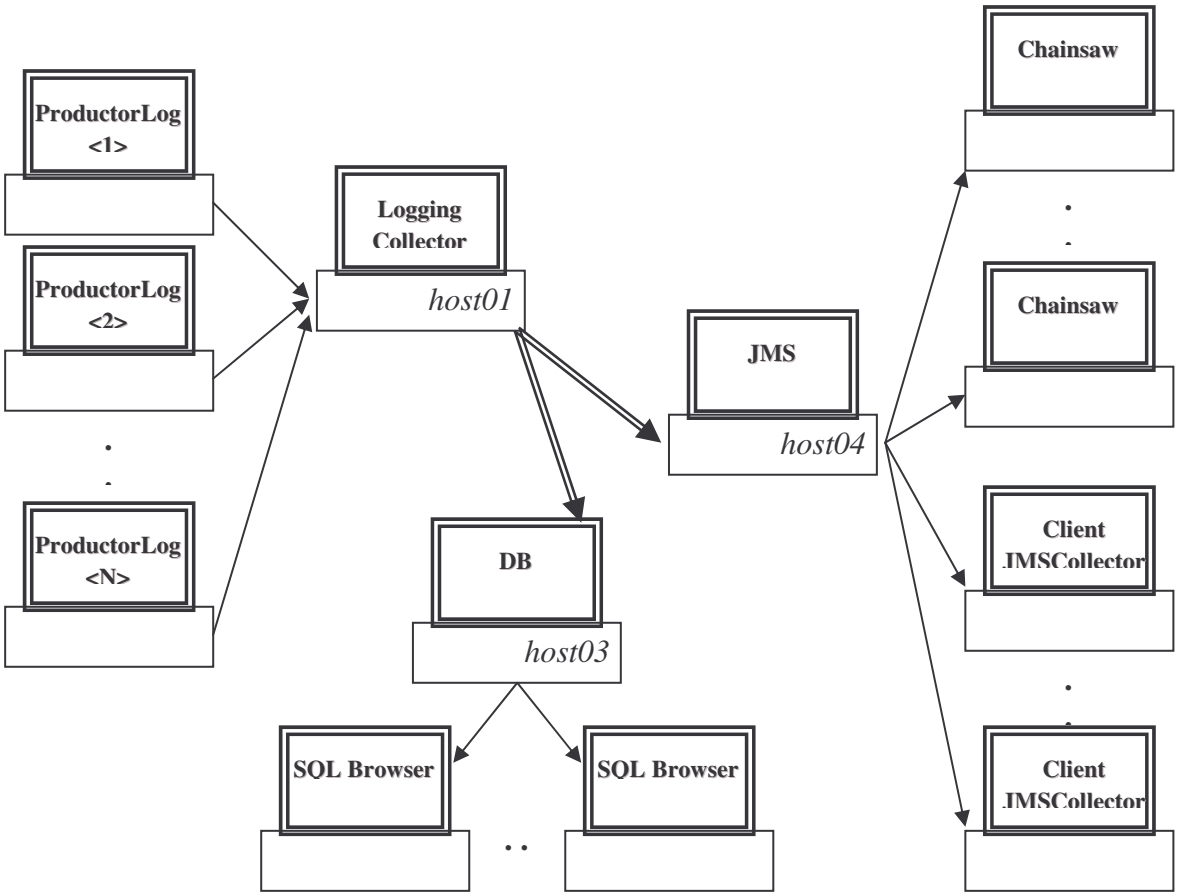

Figure 11.2 Example: Logging Collector, JMS + Chainsaw and DB.

- **ProductorLog applications**

They must be able to send the logging information to the Logging Collector, so their configuration files (*collectorClient\scripts\configFiles\ ProductorLog.config*) are:
remoteHost=host01
portHost=3333

- **Logging Collector**

It must be configured to send the collected logging information to JMS + Chainsaw and to DB. The output section of the configuration file will be of the type:

```
JMSOutputStatus = ON
JMSOutputJNDIInitialContextFactory = com.sun.jndi.ldap.LdapCtxFactory
JMSOutputJNDIProviderURL = ldap://host04:389/o=IMQ,dc=infn,dc=it
JMSOutputTopicName = cn=mt
JMSOutputTopicConnectionFactory = cn=tcf

DBOutputStatus = ON
DBOutputUrl = jdbc:mysql://host03:4306/testDB
DBOutputUser = test
DBOutputPassword = test
DBOutputDriverClass = com.mysql.jdbc.Driver
```

- **Chainsaw**

To use Chainsaw to receive and/or to filter the logging information from JMS it's necessary to refer to ***chainsaw.zip*** package in the Logging Collector's release.
After starting the Chainsaw by the *startChainsaw.sh* script, it is necessary to define an JMSReceiver through the "*Creates and configures a new receiver*" button and "*New JMSReceiver*" link and to configure it specifying:
jndiPath: path of the jmsPlugin.properties file (see *chainsaw.zip*) properly configured;
name: JMSReceiver's name (only a label);
threshold: minimum log level enabled for the JMSReceiver;
topicFactoryName: topic connection factory;
topicName: topic's name.

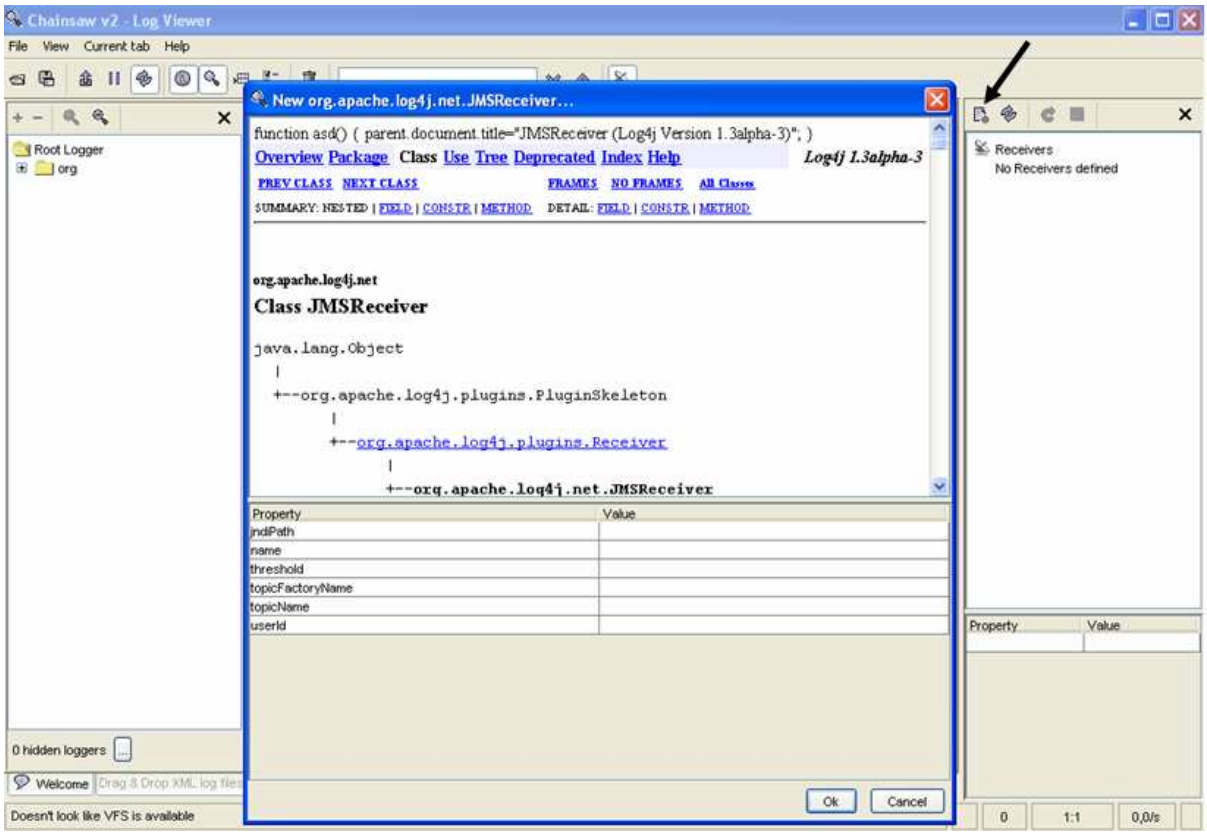

Figure 11.3 Chainsaw: how to create a JMSReceiver.

- **ClientJMS**

They must be able to receive logging information from JMS, so their configuration file (*collectorClient\scripts\configFiles\ ClientJMSCollector.config)* is:

```
initial_context_factory=com.sun.jndi.ldap.LdapCtxFactory
provider_url= ldap://host04:389/o=IMQ,dc=infn,dc=it
topicConnectionFactory=cn=tcf
topicName=cn=mt
```

12 Frequently asked questions

1. *When I press a button in the command web page nothing happens.*
Check the browser settings.
The javascript is enabled?
You should enable the javascript in the browser you are using.

2. *In the collected logging information isn't displayed the location information.*
Have you set the LocationInfo property?
The LocationInfo property must be set to true.

3. *After restarting of the Logging Collector, the logging information isn't collected anymore.*
You must check if applications which send logging information are running and the ReconnectionDelay property has been opportunely set.

4. *Why in the monitoring web page there is a mismatch between the ReceivedMessages to the inputs and the ReceivedMessages to the appenders?*
Because the monitoring web page is built dynamically and so it's possible happen:
Number of ReceivedMessages to the inputs = 100
Number of ReceivedMessages to one appender = 105
That it is correct!!! It means that in the time gap between the retrieving of the ReceivedMessages number to the inputs and the retrieving of the ReceivedMessages number to the appender, the Logging Collector has collected another five pieces of logging information.

5. *How do I select the appender queue size?*
This is a delicate issue, because on the one hand you would like to use a large appender queue size to decouple the input from the output of the Logging Collector on the other hand you would like to avoid an "Out Of Memory" error. So, it is advised to set an acceptable queue size.
In some performance tests, for example, it has been used a queue size of 15000 elements.

13 Appendix A: Configuration file

The Logging Collector configuration (CollectorConfiguration.xml file) is saved in the **<TOMCAT_HOME>/*Collector/FileConfiguration/*** folder, where <TOMCAT_HOME> is the TOMCAT installation folder.

Below the complete configuration file of the Logging Collector.

```
<?xml version="1.0" encoding="UTF-8" ?>
<CONFIGURATION>

 <!-- INPUT CONFIGURATION -->
 <INPUT>
  <TCPHostName>TCPINPUT</TCPHostName>
  <TCPHostStatus>ON</TCPHostStatus>
  <TCPHostPort>3333</TCPHostPort>
 </INPUT>

 <INPUT>
  <TCPHostXMLName>TCPXMLINPUT</TCPHostXMLName>
  <TCPHostXMLStatus>ON</TCPHostXMLStatus>
  <TCPHostXMLPort>3334</TCPHostXMLPort>
 </INPUT>

 <!-- OUTPUT CONFIGURATION -->
 <OUTPUT> <!-- JMS  -->
   <JMSOutputName>JMS</JMSOutputName>
   <JMSOutputStatus>OFF</JMSOutputStatus>
   <JMSOutputLogLevel>DEBUG</JMSOutputLogLevel>
   <JMSOutputWorkingMode>NO_QUEUE</JMSOutputWorkingMode>
   <JMSOutputQueueMaxSize>1000</JMSOutputQueueMaxSize>
   <JMSOutputJNDIInitialContextFactory>
      com.sun.jndi.ldap.LdapCtxFactory
   </JMSOutputJNDIInitialContextFactory>
   <JMSOutputJNDIProviderURL>
    ldap://sadn02:389/o=IMQ,dc=infn,dc=it
   </JMSOutputJNDIProviderURL>
   <JMSOutputTopicName>cn=mt</JMSOutputTopicName>
   <JMSOutputTopicConnectionFactory>cn=tcf</JMSOutputTopicConnectionFactory>
   <JMSOutputMessageSelector>NO</JMSOutputMessageSelector>
 </OUTPUT>

 <OUTPUT> <!-- DB -->
   <DBOutputName>DB</DBOutputName>
   <DBOutputStatus>OFF</DBOutputStatus>
   <DBOutputLogLevel>DEBUG</DBOutputLogLevel>
   <DBOutputWorkingMode>NO_QUEUE</DBOutputWorkingMode>
   <DBOutputQueueMaxSize>1000</DBOutputQueueMaxSize>
   <DBOutputUrl>jdbc:mysql://localhost/test</DBOutputUrl>
   <DBOutputUser>guest</DBOutputUser>
   <DBOutputPassword>guest</DBOutputPassword>
   <DBOutputDriverClass>com.mysql.jdbc.Driver</DBOutputDriverClass>
 </OUTPUT>

  <OUTPUT> <!-- SOCKET -->
```

```
    <SocketOutputName>SOCKET</SocketOutputName>
    <SocketOutputStatus>OFF</SocketOutputStatus>
    <SocketOutputLogLevel>DEBUG</SocketOutputLogLevel>
    <SocketOutputWorkingMode>NO_QUEUE</SocketOutputWorkingMode>
    <SocketOutputQueueMaxSize>1000</SocketOutputQueueMaxSize>
    <SocketOutputHost>localhost</SocketOutputHost>
    <SocketOutputPort>4445</SocketOutputPort>
  </OUTPUT>

</CONFIGURATION>

```

14 Appendix B: Appender in detail

14.1 Appender structure

The following figure shows in detail the appender structure.

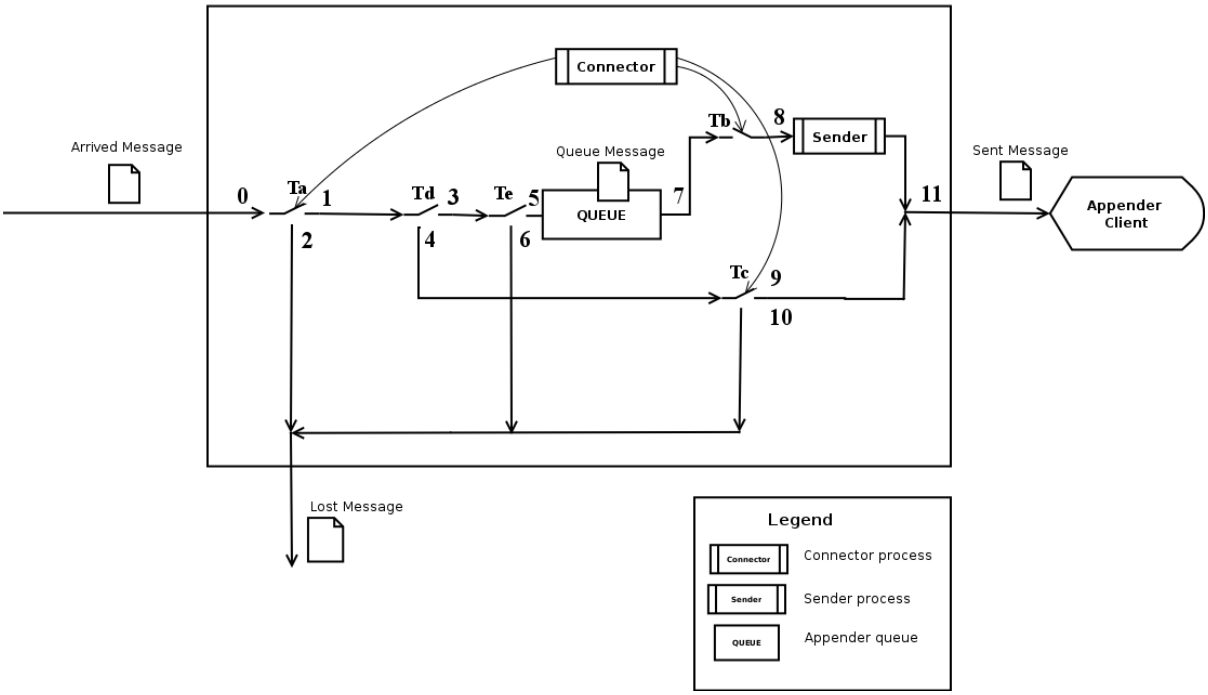

Figura 14.1 Appender structure.

When the appender is in the STANDBY state the *Connector* process is activated to open a new connection to the output appender client and to keep the switches (see figure 14.1) in the following positions:

- Ta switch in position 2 (only for the first activation of the Connector process)
- Tb switch open
- Tc switch in position 10

Once the connection is open, the *Connector* process is destroyed after having set the switches in the new positions:

- Ta switch in position 1
- Tb switch close
- Tc switch in position 9

The *Sender* process is activated at the end of the *Connector* process only if the appender is set to use its queue (QUEUE_WITH_LOST_MSG or QUEUE_WITH_NO_LOST_MSG working mode).

	The Ta switch is in position 2. After the first STANDBY to ON state transition will be always left in position 1.

	The Td switch is in position 3 when the appender is in QUEUE_WITH_LOST_MSG or QUEUE_WITH_NO_LOST_MSG working mode otherwise is in position 4.

The Te switch is in position 6 only when the appender is in QUEUE_WITH_LOST_MSG working mode and the appender queue is full.

14.2 Appender working modes in detail

Each appender can function in one of the following modes:

- **NO_QUEUE**
- **QUEUE_WITH_LOST_MSG**
- **QUEUE_WITH_NO_LOST_MSG**

14.2.1 NO_QUEUE working mode

In this appender working mode the appender queue isn't used.

Transition	State	Condition	Path
OFF to STANDBY	STANDBY	---	0 - 2
Notes			
Ta switch in position 2 Td switch in position 4 Tc switch in position 10			

Table 14.1 NO_QUEUE mode: From OFF to STANDBY transition.

Transition	State	Condition	Path
STANDBY to ON	ON	---	0 -1 – 4 – 9 – 11
Notes			
Ta switch in position 1 Td switch in position 4 Tc switch in position 9			

Table 14.2 NO_QUEUE mode: From STANDBY to ON transition.

Transition	State	Condition	Path
ON to STANDBY	STANDBY	---	0 – 1 – 4 – 10
Notes			
Ta switch in position 1 Td switch in position 4 Tc switch in position 10			

Table 14.3 NO_QUEUE mode: From ON to STANDBY transition.

Transition	State	Condition	Path
ON to OFF	OFF	---	---
Notes			
The appender is destroyed.			

Table 14.4 NO_QUEUE mode: From ON to OFF transition.

14.2.2 QUEUE_WITH_LOST_MSG working mode

Transition	State	Condition	Path
OFF to STANDBY	STANDBY	---	0 - 2
Notes			
Ta switch in position 2 Tb switch open Td switch in position 3 Te switch in position 5			

Table 14.5 QUEUE_WITH_LOST_MSG mode: From OFF to STANDBY transition.

Transition	State	Condition	Path
STANDBY to ON	ON	Appender queue isn't full.	0 – 1 – 3 – 5 – 8 – 11
Notes			
Ta switch in position 1 Tb switch close Td switch in position 3 Te switch in position 5			

Table 14.6 QUEUE_WITH_LOST_MSG mode: From STANDBY to ON transition.

Transition	State	Condition	Path
STANDBY to ON	ON	Appender queue is full.	0 – 1 – 3 - 6
Notes			
Ta switch in position 1 Tb switch close Td switch in position 3 Te switch in position 6			

Table 14.7 QUEUE_WITH_LOST_MSG mode: From STANDBY to ON transition.

Transition	State	Condition	Path
ON to STANDBY	STANDBY	Appender queue isn't full.	0 – 1 – 3 - 5
Notes			
Ta switch in position 1 Tb switch open Td switch in position 3 Te switch in position 5 The logging information is left into the appender queue.			

Table 14.8 QUEUE_WITH_LOST_MSG mode: From ON to STANDBY transition.

Transition	State	Condition	Path
ON to STANDBY	STANDBY	Appender queue is full.	0 – 1 – 3 - 6
Notes			
Ta switch in position 1 Tb switch open Td switch in position 3 Te switch in position 6			

Table 14.9 QUEUE_WITH_LOST_MSG mode: From ON to STANDBY transition.

Transition	State	Condition	Path
ON to OFF	OFF	---	---
Notes			
The appender is destroyed.			

Table 14.10 QUEUE_WITH_LOST_MSG mode: From ON to OFF transition.

14.2.3 QUEUE_WITH_NO_LOST_MSG working mode

Transition	State	Condition	Path
OFF to STANDBY	STANDBY	---	0 - 2
Notes			
Ta switch in position 2 Tb switch open Td switch in position 3 Te switch in position 5			

Table 14.11 QUEUE_WITH_NO_LOST_MSG mode: From OFF to STANDBY transition.

Transition	State	Condition	Path
STANDBY to ON	ON	---	0 – 1 – 3 – 5 – 7 – 8 - 11
Notes			
Ta switch in position 1 Tb switch close Td switch in position 3 Te switch in position 5 The appender input client slows down if the appender queue becomes full.			

Table 14.12 QUEUE_WITH_NO_LOST_MSG mode: From STANDBY to ON transition.

Transition	State	Condition	Path
ON to STANDBY	STANDBY	---	0 – 1 – 3 - 5
Notes			
Ta switch in position 1 Tb switch open Td switch in position 3 Te switch in position 5 The logging information is left into the appender queue and the appender input client slows down if the appender queue becomes full.			

Table 14.13 QUEUE_WITH_NO_LOST_MSG mode: From ON to STANDBY transition.

Transition	State	Condition	Path
STANDBY to OFF	OFF	---	---
Notes			
The appender is destroyed.			

Table 14.14 QUEUE_WITH_NO_LOST_MSG mode: From STANDBY to OFF transition.

Transition	State	Condition	Path
ON to OFF	OFF	---	---
Notes			
The appender is destroyed.			

Table 14.15 QUEUE_WITH_NO_LOST_MSG mode: From ON to OFF transition.

15Appendix C: Example.java

```
import java.net.InetAddress;
import java.net.UnknownHostException;

import org.apache.log4j.Level;
import org.apache.log4j.Logger;
import org.apache.log4j.NDC;
import org.apache.log4j.net.SocketAppender;

public class Example {
  SocketAppender socketAppender = null;
  Logger         log            = null;

  /**
   * Constructor
   */
  public Example() {
    // define the appender
    socketAppender = new SocketAppender();
    socketAppender.setRemoteHost("localhost");
    socketAppender.setPort(3333);
    socketAppender.setThreshold(Level.DEBUG);
    socketAppender.setLocationInfo(true);
    socketAppender.activateOptions();

    // define the logger
    log = Logger.getLogger(SumClass.class);
       try {
    NDC.push(InetAddress.getLocalHost().getHostAddress());
    } catch (UnknownHostException uke){
       // do nothing.
    }
    log.setLevel(Level.DEBUG);
       // Attached the appender to logger
    log.addAppender(socketAppender);
  }
  /**
   * @param a
   * @param b
   * @return the sum
   */

  public  int sum(int a, int b) {
    log.debug("First number: " + a);
    log.debug("Second number: " + b);
    return a + b;
  }

  public static void main(String[] args){
    Example example = new Example();
    example.sum(5, 6);
  }
}
```

16 Appendix D: Some useful web application parameters of the Logging Collector

The Logging Collector has got the **<TOMCAT_HOME>/webapps/Collector/WEB-INF/web.xml** web application configuration file. In this file there are some parameters you can change.
The parameters are written in the following format:

<init-param>
<param-name>***ParameterName***</param-name>
<param-value>***ParameterValue***</param-value>
</init-param>

Parameter Name	Parameter Default Value	Available Parameter Values
PingCommandEnabled	YES	YES / NO.
Parameter Description		
YES: To enable the ping check if you are using the SocketAppender. NO: To disable the ping check if you are using the SocketAppender.		

Table 16.1 PingCommandEnabled: web application parameters of the Logging Collector.

Parameter Name	Parameter Default Value	Available Parameter Values
StateStartMode	CONFIGURE	CONFIGURE / START.
Parameter Description		
START : if you want to start the Logging Collector at the Tomcat startup time; CONFIGURE: if you want to start the Logging Collector by the Command web page.		

Table 16.2 StateStartMode: web application parameters of the Logging Collector.

Parameter Name	Parameter Default Value	Available Parameter Values
CollectorInternalLogLevel	DEBUG	OFF / DEBUG / INFO / WARN / ERROR / FATAL.
Parameter Description		
To select the internal logging level of the Logging Collector.		

Table 16.3 CollectorInternalLogLevel: web application parameters of the Logging Collector.

For more details see [Logging Collector – Installation Guide].

17 References

1. [Logging Collector – Installation Guide] Logging Collector - Installation Guide.
2. [Log4j Manual] CeKi Gulcu: log4j - the complete manual.
3 . [Log4j -site] http://logging.apache.org/log4j/docs/index.html
4. [Chainsaw –site] http://logging.apache.org/log4j/docs/chainsaw.html
5. [Axis Doc-site] Documentation for Axis Users:
http://ws.apache.org/axis/java/index.html.