JFreeMon

Integration Guide

11.3.2007

Tuomas Tikka

Contents

1. Overview

This document is a technical integration guide to the open source general purpose resource monitoring software, JFreeMon.

Purpose of the Document

This document is an integration guide, which focuses on the way clients connect to the server in the JFreeMon resource monitoring system. It covers the following main topics:

- Technical overview of JFreeMon
- The JFreeMon protocol
- Out-of-the-box JFreeMon clients
- Client side error codes
- Frequently asked questions

Intended Audience

This document is intended for technical personnel involved in evaluating JFreeMon, developing the server itself or developing extensions for it, for example in the way of creating new clients. The list below identifies key areas that the reader should have a fair knowledge of:

- Java 2 SDK Standard Edition
- UDP and datagrams
- The client/server model

Used Terms

The table below explains the terms used in this document.

Term	**Explanation**
Client	Refers to clients transmitting data to the JFreeMon server.
JFreeMon	Open source general purpose resource monitoring software.
Server	Refers to the JFreeMon desktop server application.
UDP	User Datagram Protocol. Connectionless packet switched.

Version History

The table below shows the version history of this document.

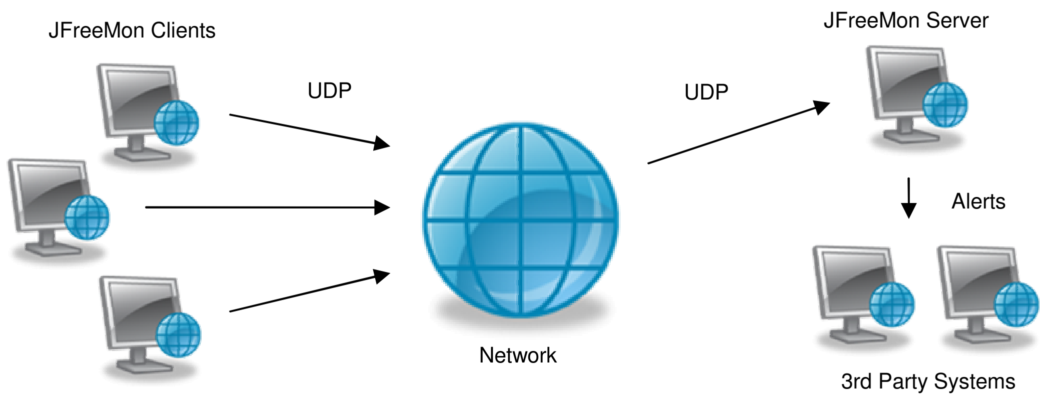
Version	Created	Description
0.1	11.3.2007	First version of document.

2. General

This chapter provides general information on JFreeMon. It provides a brief overview of how the software system works as a whole and goes deeper into two of the major components, the client and server.

What is JFreeMon?

JFreeMon is a general purpose resource monitoring software with an emphasis on ease-of-use. A centrally managed server is used to monitor incoming resources from clients. These clients can transmit practically any kind of data - it is the server's responsibility to understand it.



As well as normal monitoring and logging capabilities, the server has some nice features to act on the incoming data also. Special warning and alert levels can be configured per client for its data, and when these levels are reached, certain actions can be taken. These actions are rules, which are used to dynamically automate tasks in the server. For example, if a client reached a warning level, a rule might show a notification popup, and if the alert level is reached, the server might send an email notification or even push the alert to a 3rd party system.

JFreeMon Server

The JFreeMon server is a Java desktop application which centrally manages the input coming from its clients. The server functionality can be roughly divided into:

- Configure clients to accept incoming data
- Configure warning and alert levels per client data
- Configure script-based rules to trigger actions if these levels are reached
- Provide overall logging and monitoring of client data

JFreeMon Clients

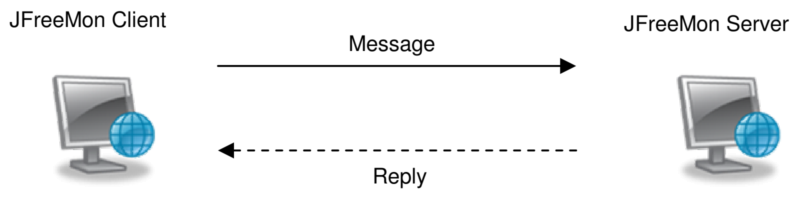
JFreeMon clients can be basically anything, as long as they communicate to the JFreeMon server using the agreed upon communication standard. The client/server communication is a UDP based messaging protocol, which is designed to be easy to take into use, simple and efficient. The protocol is described in detail later on this document, but what’s worth mentioning here is that clients can be created for practically any environment, from Perl or Python to C/C++ or Java. The idea is that these “thin” clients are simple enough to be used from existing projects and the real power is left to the JFreeMon server.

3. JFreeMon Protocol

This chapter describes the protocol used by the clients when communicating with the server.

Messages & Replies

In the JFreeMon model, each transaction between the client and server consists of at least one message. This message always originates from the client and, depending on whether the communication is synchronous or asynchronous, is responded with a reply.



The message and reply contain the same header fields, which are specified in the table below:

Field	Length	Description
Encryption	1	0 if not encrypted, otherwise ID of encryption key.
Version	1	Object version (1, 2, 3, etc.).
Timestamp	7	Timestamp (year, month, day, hour, minute, second).
Source	16	Source identifier (client identifier).
Data		List of data objects in the message/reply.

Encryption

The encryption field specifies if the object data is encrypted or not. A value of 0 means that the data is encrypted and a value in the range of 1-4 specifies that the data is encryption as well as the encryption key number configured in the server. Any other value is erroneous.

Version

The version field specifies the object version. Valid values are 1-255. Note that the server always makes the decision of accepting or rejecting a message by its version.

Timestamp

The timestamp field defines the object creation time. The 7 bytes are used in the following manner:

- bytes 1-2 define the year using the formula (byte 1 * 100) + byte 2
- byte 3 defines the month (1-12)
- byte 4 defines the day of month (1-31)
- byte 5 defines the hour of day (0-23)
- byte 6 defines the minute (0-59)
- byte 7 defines the second (0-59)

Source

The source field identifies the client. Each client has a unique source identifier, which is also used in the server to identify the incoming data.

Example Header

The example below contains an example header, with individual bytes separated by whitespaces. The example reads out:

- Encryption: 0 (no encryption)
- Version: 1
- Timestamp: 23.2.2007 12:00:00
- Source: 0000000000000001

```
0 1 20 07 2 23 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Data Objects

As defined in the previous section, messages and replies can contain data objects. These objects contain the actual transmitted information in a name/value type structure. A single data object structure is defined in the table below:

Field	Length	Description

Size	1	The length of the data field.
Code	1	The data code identifier.
Type	1	The data type.
Data		The actual data bytes.

Size

The size field specifies the length of the data object in bytes. The size always includes the data headers (size, code and type).

Code

The code field identifies the data being transmitted. A single client can transmit several codes, for example in a server monitoring case, codes could resemble:

- Code 1 for CPU usage
- Code 2 for memory usage
- Code 3 for network status
- Code 4 for hard disk usage
- etc.

Type

The type field indicates the type of data in the object. Possible data types are:

- Integer (4 bytes)
- Long (8 bytes)
- Float (4 bytes)
- Double (8 bytes)
- String (1-N bytes)
- Date (7 bytes)

Data

The data field contains the actual information (value) that is transmitted from the client to the server.

Example Data

The table below contains an example data object, with individual bytes separated by whitespaces. The example reads out:

- Size: 10
- Code: 1
- Type: 50 (string type, see JFreeMon Commons API for details)
- Data: “Testing” (from the ASCII character table)

```
10 1 50 84 101 115 116 105 110 103
```

4. Out-of-the-box Clients

This chapter describes two client-side components, which are available as part of the JFreeMon software system. With these ready-made components, it is easy to integrate your software to JFreeMon without needing to know details of the underlying communication protocol.

JFreeMon Client

The JFreeMon Desktop Client is a library for the Java desktop and enterprise environments. It can be practically used from any Java Runtime Environment and requires no additional 3rd party libraries.

Requirements

Software	Version
Java SDK	1.4 or later

Sample Code

```
// Initialize message object:
//
// Encryption: 0 (none)
// Version   : 1
// Timestamp : Convert from current time
// Source    : 0000000000000001
message = new MessageDTO();
message.setMessageEncryption((byte) 0);
message.setMessageVersion((byte) 1);
message.setMessageTimestamp(Converter.date2Bytes(new Date()));
message.setMessageSource(new byte[]{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1});

// Initialize data object:
//
// Size: 7 (calculated automatically in constructor)
// Code: 1
// Type: Integer (determined by constructor)
// Data: 1
data = new DataDTO((byte) 1, 1);

// Append data to message.
message.addData(data);

// Send message:
//
// Server address: localhost
// Port number   : 9001
try {
    JFreeMonClient client = JFreeMonClient.getInstance();
    client.sendAsynchronous(InetAddress.getLocalHost(), 9001, message);
} catch (Exception e) {
}
```

JFreeMon Mobile Client

The JFreeMon Mobile Client is a library for the Java mobile environment. It can be used to integrate mobile clients to JFreeMon. The component design is simple allowing it to be run from practically any Java Mobile Environment and requires no additional 3rd party libraries.

Requirements

Software	Version
Java Micro Edition	*MIDP 2.0 or later

**The JFreeMon Mobile Client uses UDP, which is not required in MIDP 2.0. It depends on the mobile terminal if UDP is supported.*

Sample Code

```
// Initialize message object:
//
// Encryption: 0 (none)
// Version   : 1
// Timestamp : Convert from current time
// Source    : 0000000000000001
message = new MessageDTO();
message.setMessageEncryption((byte) 0);
message.setMessageVersion((byte) 1);
message.setMessageTimestamp(Converter.date2Bytes(new Date()));
message.setMessageSource(new byte[]{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1});

// Initialize data object:
//
// Size: 7 (calculated automatically in constructor)
// Code: 1
// Type: Integer (determined by constructor)
// Data: 1
data = new DataDTO((byte) 1, 1);

// Append data to message.
message.addData(data);

// Send message:
//
// Server address: localhost
// Port number   : 9001
try {
    JFreeMonClient client = JFreeMonClient.getInstance();
    client.sendAsynchronous("localhost", 9001, message);
} catch (Exception e) {
}
```

5. Client-side Error Codes

When using the out-of-the-box JFreeMon clients, a standard set of error codes are used that apply to each client. This chapter provides a reference to these error codes and typical situation when they occur.

The table below describes the possible status codes, which are present in every log generated by a JFreeMon client.

Code	**Description**
0	Success, no error occurred. Used mainly in INFO level logs.
1000	Connection related error. Mainly related to network I/O problems.
1001	Server response error. Mainly related to errors parsing server replies.

6. Further Reading

This chapter provides links for further reading.

1. Java SDK Standard edition
   http://java.sun.com/javase/

2. Java SDK Enterprise edition
   http://java.sun.com/javaee/

3. Java SDK Micro Edition
   http://java.sun.com/javame