JENNIFER®
Enterprise Web Business Monitoring

JENNIFER User's Guide

JENNIFER'S KEY FEATURES

- Comprehensive Dashboard
- Real-time Resource/Service Monitoring
- Performance Problem Diagnosis and Resolution
- Application Tracing and Tuning
- Statistical Analysis and Reporting on the Performance Data
- Dynamic Profiling and StackTrace

JENNIFER v4
User's Guide

JS-0810-03

JENNIFER v4 User’s Guide

Written By **Sunjo Kim, Kevin Jung**

Published By **JENNIFERSOFT,Inc**

Edited By **Amie Song**

First Edition **Oct, 30th 2008**

Recent Update **April, 12th 2011**

Document version **4.5**

Document Number **JS-0810-03**

Reference Book **JENNIFER v4 Installation Guide**

Table of Contents

Copyright

Trademarks

JENNIFER™ and JENNIFERSOFT® are the trademarks of JenniferSoft,Inc in the United States, other countries, or both. All Rights Reserved under Copyright and intellectual property rights. As a condition of your use of the trademarks, you will not use the trademarks for any purpose that is unlawful or prohibited by these terms, conditions, and notices.

JENNIFERSOFT® JENNIFER™

Other company, product, and service names may be trademarks or service marks of others. Any rights not expressly granted herein are reserved.

Upload, or otherwise make available, files that contain images, trademarks or other material protected by intellectual property laws, including, by way of example, and not as limitation, copyright or trademark laws (or by rights of privacy or publicity) unless you own or control the rights thereto or have received all necessary consent to do the same.

Sun, Sun Microsystems, Solaris, Java, JavaEE, EJB, GlassFish, JMX, JNI, JSP, JDBC, JNDI, and JVM are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

IBM, WebSphere, AIX, iSeries, z/OS, iSeries, DB2, CICS and CTG are trademarks of IBM Corporationin the United States, other countries, or both.

BEA, WebLogic, WLI, ALSB, TUXEDO, Jolt and WTC are trademarks of Oracle, Inc the United States, other countries, or both.

Microsoft, Window 2000, Vista, XP, Windows Server 2003, 2008 and NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

The following terms are trademarks of other companies:

Fujitsu® Interstage®
Hitachi® Cosminexus®
Sybase® EAServer®
Macromedia® JRun®
Tmaxsoft® JEUS® WebT®
Apache® Apache Derby® Jakarta Tomcat® DBCP®
Caucho Technology® Resin®
RedHat® JBoss®

Apache® JServ®
HP-UX® Itanium®
Sun Solaris®
Linux®
Compaq® True64®
Unipoint® J*Link®

Preface

This document comprehensively provides the details of the JENNIFER server and agent architecture, the organization methods, the user interfaces, and the performance data collection and analysis method necessary for operation and management of JENNIFER v4, after or before it is installed.

People who should read this document include technical support engineers for JenniferSoft, JENNIFER customer operators who operate JENNIFER on a regular basis, application developers who are attempting to fine-tune their application through JENNIFER in the application development or testing phases, as well as people who simply wish to understand the detailed functions of JENNIFER.

JENNIFER is an application performance management solution. We believe that JENNIFER can deliver special values and results to people who are in charge of performance tuning for capacity estimation and problem analysis, database performance tuning and the extraction of SQL queries from an application, as well as WAS operators for Java EE/WAS performance management and application developers for application performance improvements.

Before reading this document, it is essential for you to have an understanding of the basic environment for the Java Virtual Machine (JVM), be able to develop Java applications and analyze source code, and have sufficient knowledge regarding the installation, organization and operation of the Java Application Server (JAS). In addition, a basic knowledge of TCP/IP and UDP communication network protocols for inter-system communication is required, as well as a minimal understanding of TCP/UDP port setting for systems. In addition, readers should fully understand the concept of a JDBC connection between the WAS and the database and have sufficient knowledge of the specific JDBC connections applied to the system. Readers should have basic knowledge regarding the preparation of database SQL queries.

Furthermore, readers should have some general understanding of the architecture of the Web, WAS and DB servers operating in a web-based system and fundamental terminologies for performance analysis, such as visitors, concurrent users, throughputs and response time.

The script file extensions are different depending on the operating systems. A script designed to run at the UNIX and Rinux has sh file extension and Micresoft's Windows has bat. This manual is written for UNIX and Linux users. Therefore, if you use Microsoft Windows, we allow you to use the script file with bat extension name.

This document does not include any content related to JENNIFER installation. For specifics on installation, you are recommended to refer to the JENNIFER installation guide. Since this document is an official JENNIFER 4 User's Guide, we tried to describe all contents, including a variety of options. For further details of the application or specific case studies, you are recommended to refer to an additional guideline or technical notes provided on the JenniferSoft website.

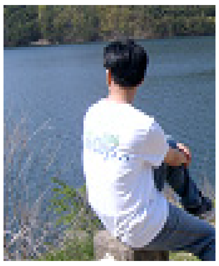
Finally, I would like to thank all our customers and supporters, who have given positive feedback on JenniferSoft and its products prior to the release of JENNIFER v4 to the market, as well as the engineers at our partner companies who are fully devoted to providing technical support.

The team that wrote this book

SungJo Kim Director of JenniferSoft R&D Institute

Sung-Jo Kim is a performance analysis consultant and developer who studies Java-based software architectures and performance. He has studied software development methodologies and has taught software architecture as a software architect at LGCNS. He has conducted performance tuning of Java applications and WAS for dozens of research projects. In 2005, he joined the JenniferSoft R&D center. At present, he is a leader of the development of the JENNIFER core engine and low-level modules. In particular, he delivered the concept of transaction profiling based on a distribution graph of response times (X-View), and combined it with the actual product..

Kevin Jung Senior Researcher of JenniferSoft R&D Institute



Based on the wide range of field experiences in .NET programming, Kevin Jung is mainly working on developing application performance monitoring solution, JENNIFER .NET and writing a technical documents overing .NET architecture.

Albert Jeon Senior Researcher of JenniferSoft R&D Institute

He was charge in UI and network development at TmaxCore R&D and moved to the JenniferSoft, Inc in 2010. Aalbert is developing JENNIFER UI and likes to read books in various areas. He regard himself as a happy developer.

Khalid Saeed Junior Researcher of JenniferSoft R&D Institute

Khalid is a junior developer in the R&D team, he is specialized in Java programming. Khalid came to Korea in 2008 for his postgraduate studies and after obtained his M.Sc. he joined Jennifer R&D in 2010 and since then he is working in java programming, Jennifer Testing, and writing technical documents.

In addition, Senior Manager Kyoung-sik Yoon(Sam) of the JenniferSoft R&D team have also made significant contributions by verifying, proofreading and testing the content of this document.

In addition, Marketing Manager Amie Song from the overseas marketing division, helped to prepare and edit this guide as a coordinator.

Comments welcome

Your comments will be most beneficial to us!
Issues, concerns, and/or any other comments that you have regarding this book can be mailed to:
E-mail: manual@jennifersoft.com

Mail your comments to:

360 Fairview Way Milpitas, CA 95035

Chapter introduction

This document is organized in a certain sequence to help readers to understand the concepts. However, you may skip to the chapters that are most relevant to you if you wish, as each chapter is independent from each other. This document has the following chapters.

1. "Getting Started" This chapter describes details about JENNIFER's concept, features and benefits.

2. "JENNIFER Architecture" This chapter describes details about JENNIFER architecture, module diagram, performance data collected by JENNIFER, and network organization.

3. "JENNIFER Java Agent Configuration & Management" This chapter describes the main components for the JENNIFER Java agent and how to install/configure/manage JENNIFER Agent.

4. "JENNIFER .NET Agent Configuration & Management" This chapter describes the main components for the JENNIFER .NET agent and how to install/configure/manage JENNIFER Agent.

5. "User Interface" JENNIFER client runs on the web-browser. This chapter provides guidance for the browser environment and information about JENNIFER client's interface.

6. "Service & User Monitoring" This chapter describes application monitoring, peak load control, rea-time active service monitoring, service dump and user concept (Visit user, active user, concurrent user, etc) and its monitoring.

7. "Chart & Dashboard" This chapter describes how to configure a dashboard by using various charts offered by JENNIFER.

8. "X-View & Profiling" This chapter describes the concept and example of response time scatter graph, X-View, service profiling and how to analyze X-View data. Especially, this chapter introduces JENNIFER 4.0 new features such as dynamic profiling and dynamic stackTrace.

9. "Java Resources, External Transactions and JDBC Monitoring" This chapter describes the method for monitoring resources such as CPU and memory, and external transactions and JDBC monitoring in interconnection with external applications.

10. "NET Resources and DB Connection Monitoring" This chapter describes alerts and exceptions that occur in the Java application, and analyzes the status of the Java application to generate proper alerts. It also guides how to check the alerts and analyze the contents of previous alerts through the user interface in real time.

11. "Alerts and Exception Monitoring"This chapter describes the all the errors and exceptions that is defined bu the JENNIFER.

12. "JENNIFER Server Configuration & Management" This chapter describes the manner in which the network of the JENNIFER server is organized to process various network communications between the JENNIFER client or JENNIFER agent and the JENNIFER server. This chapter mainly deals with the port number configuration.

13. "Performance Status & Reports" This chapter describes methods for monitoring and analyzing the diverse performance data collected by the JENNIFER agent.

14 "Java Application Advanced Monitoring" This chapter describes methods for expanding the JENNIFER agent by programming and interconnecting the performance data collected by the JENNIFER server with other applications.

15 "REMON" The REMON module is one of the JENNIFER independent agents that collects and processes non-standard data and sends it to the JENNIFER server. This chapter describes that REMON collects the performance data from another adaptor which is programmed by shell or class program. It also explains the real-time LogWatcher.

16 "APPENDIX" JENNIFER agent/server configuration options and JENNIFER error codes are in the APPENDIX.

About marks

The terminologies and nomenclatures below are frequently used in this document. Before reading this guide, make sure to refer to the following content to help your understanding of the material in this document.

Mark	Description
Arial (Bold)	Main font, method/class/file/directory name,
Body Box Courier	`http_service_class = mysys.AServlet;mysys.BServlet`
Courier Bold	`http_service_class = mysys.AServlet;mysys.BServlet`
[Menu]-Bold	**[Dashboard | JENNIFER Dashboard]**
[Main menu 〕Sub menu]	Main/Sub menu order
[Delimiter]	[,]
Notice/Warning	Tipbox
Reference	[Transaction Data(32 page)]
Option name	**config_refresh_check_interval**

1 Getting Started

1.1. Application Performance Management

Application Performance Management(APM) is a process of developing a progressive system that effectively monitors the availability of enterprise applications, identifies and resolves the application performance problems, and predicts/prevents future application performance problems.

Unlike the traditional system management(SMS and NMS), APM enhances the organization’s ability to monitor the enterprise application service status and resolve application performance problems, resulting in reduced Total Cost of Ownership(TCO) and enhanced customer service.

1.2. About JENNIFER

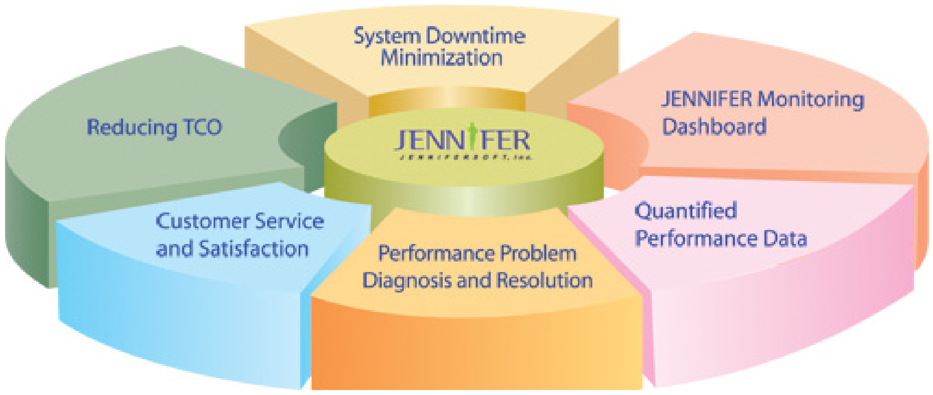
JENNIFER developed by JenniferSoft.Inc is the APM solution running on Application Server to manage the performance of enterprise Java applications.

Notice: JENNIFER is also able to monitor the Java application that runs independently instead of running on the application server..

1.3. Benefits

The following are the major benefits of JENNIFER.

Figure 1-1:Benefits of JENNIIFER


System Downtime Minimization
JENNIFER Monitoring Dashboard
Reducing TCO
JENNIFER
JENNIFERSOFT, Inc.
Quantified Performance Data
Customer Service and Satisfaction
Performance Problem Diagnosis and Resolution

1.3.1. System Downtime Minimization

If you adopt the JENNIFER, JENNIFER immediately provides the critical information for diagnosis and resolution of the performance problem with just a few clicks of the mouse, achieving fast and efficient service recovery.

1.3.2. JENNIFER Monitoring Dashboard

JENNIFER produces the comprehensive system resource data, application performance data, and business data via dynamic and intuitive performance monitoring dashboard which can be used to build AMS project or ITSM system.

1.3.3. Performance Problem Diagnosis and Resolution

Through progressive system monitoring and response time scatter graph (X-View), JENNIFER identifies the performance problems and dispenses alerts automatically via email or SMS system when the performance problem occurs. JENNIFER then provides the root-cause analysis on the identified performance problems for speedy resolution.

1.3.4. Customer Service and Satisfaction

Customer satisfaction is enhanced through minimized system downtime, reduced performance problem, and established Peak Load Control (PLC), ensuring maintenance services and stable operation for the customers.

1.3.5. Reducing TCO

JENNIFER reduces Total Cost of Ownership(TCO) in IT environments: reducing costs by minimizing system downtime, preventing performance problems, maximizing system performance through tuning, increasing efficiencies in IT operations, etc.

1.3.6. Quantified Performance Data

Through statistical analysis of resources, applications, and business data, quantified evidentiary data can be established for use in capacity planning, system tuning, and internal communication.

1.4. JENNIFER Features

The following are the JENNIFER core features.

1.4.1. Service Monitoring

When monitoring the status of the application services, it is important to retrieve the performance data in real-time and grasp the mutual relationship between each service transactions.

JENNIFER provides following features in service monitoring.

- Concurrent User
- Active Service (Currently Running Service)
- Throughput(TPS)
- Real-Time Service Response Time Graph
- SQL Trace (Including External Transaction and BIND Variable)
- HTTP Request Parameter Tracking

1.4.2. Resource Monitoring

Resource monitoring includes managing the physical and logical resources used in running an application service.

JENNIFER provides following features in resource monitoring.

- DB Connection Pool Status
- System and Process CPU Usage (KERNEL/USER/IO)
- System and Process Memory Usage
- Process Heap Memory Usage
- FILE Read/Write Status
- TCP/IP SOCKET INPUT/OUTPUT Tracking
- Collection/Live Object Count
- Status of Application Server Resource Used by JMX(Java Management Extension)

1.4.3. Performance Data Analysis

The performance problem in a web system can be categorized into two types: conditional performance problems caused by anomalous circumstances or relative performance problems caused by an increase in load and throughput that is more than the amount system can handle. JENNIFER possesses various types of data collection and analysis modules for detecting all types of performance problems

- Java Memory Leak Detection (Collection/Live Object)
- Unreturned DB Connection Tracing
- Unreturned JDBC(Statement/ResultSet) Tracing

- Unprocessed JDBC Transaction (commit/rollback) Tracing
- Application Exception Tracing
- SQL Exception Tracing
- Dump on Running Service
- HTTP Session Dump Capability
- Finding Loaded Class

1.4.4. Real-time Active Service Monitoring

JENNIFER transmits the performance data through reverse-direction protocol that connects from JENNIFER Server to JENNIFER Agent whenever user requests for the data. Using this method, JENNIFER can extract the snapshot information of the running active services by unit of time. Active service data shown in JENNIFER is not a summarized data of events that occurred in the past but the real-time data concerning active services.

Notice: The active service data offered by JENNIFER is a real-time transaction data, not a past one.

1.4.5. Response Time Scatter Graph(X-VIEW) and Profiling

JENNIFER's response time scatter graph, called X-View, presents the response time of all service transactions as plots in a scatter graph. The vertical axis is the response time of an individual transaction and the horizontal axis is the end time of each transaction's runtime.

Using the X-View, user can not only detect the delay in response time for the specific transaction(s) but also the root-cause behind the delay in the response time.

Notice: The plots may form different patterns in X-View that user can use to identify or predict the performance problem. The X-View is a powerful and intuitive tool that is more useful than using any other line graphs combined.

The X-View displays the detailed information in following areas: SQL queries, External system and interface (including Legacy), accessed files and sockets, and Tier and Layer high-level class/method profiling information.

- Method Parameter, Return Data Tracing
- High-level Class/Method Dynamic Profiling for All Transactions
- JDBC and SQL Query Tracing (Including BIND Variable)

- Legacy, External System Interface
- File/Socket Connection Tracing
- Dynamic Profiling for the Tier/Layer Class/Method

1.4.6. Dynamic Profiling

JENNIFER can register additional package, class, method and/or activate/deactivate transaction profiling without restarting the web application server.

1.4.7. Dynamic StackTrace

The traditional method for extracting Java Full StackTrace is intentionally causing an exception/error for an application resource and outputting it onto the stacktrace; JENNIFER can register a class/method during operation, allowing dynamic full stacktrace of additional class/method without changing the application source code.

1.4.8. Monitoring Dashboard per Domain

In a large scale of enterprise environment, many different business systems may exist, triggering a need for a solution that individually monitors each business system under integrated one view. JENNIFER provides performance management capability per domain that allows the user to allocate multiple business systems into different domain and manage each system under one umbrella.

1.4.9. Application/SQL Tuning

The response time of executed queries can be traced without affecting the system performance. The relative connection between SQL Query and application that used it, and the proportion of response time spent on SQL over the total response time of the application can be viewed and analyzed via JENNIFER.

1.4.10.Automatic Alert System

JENNIFER has classified many different error types that can exist in application which negatively affects the system performance. Error/Exception is grouped into “Critical”, “Error”, and “Warning” and managed by date/time. User can also add a new error/ exception type using extension adaptor.

1.4.11.Extended Monitoring Adaptor

JENNIFER features Extended Monitoring Adaptors Functionality(EMAF) that allows performance data from other system devices to be extracted and inputted into JENNIFER for analysis and reporting. Broadly, JENNIFER offers 3 types of Extended Monitoring Adaptor Functionality.

- REMON - The REMON module is one of the Jennifer independent agents that collects and processes non-standard data and sends it to the Jennifer server. Using the REMON module, you can execute an arbitrary script on the OS or an arbitrary SQL on the database, or can execute a Java application that implements a specific SQL. Using these methods, you can collect non-standard data and send it to the Jennifer server.
- ExtraAgent- ExtraAgent is imbedded in WAS interface and communicates directly with REMON (Extended Monitoring Adaptor). It collects the performance data in Java application and sends them into JENNIFER server or REMON.
- LogWatcher - LogWatcher extracts the data from various formats of user log files and provides real-time analysis, monitoring, and alert services.

1.4.12.Statistical Analysis and Reporting

JENNIFER allows user to analyze its performance problem by providing more flexible statistics because JENNIFER stores the performance data into the DB. In addition, it shows the data in customizable reports.

1.5. Supported Platform

The following are the platform information that can be supported by JENNIFER.

1.5.1. JENNIFER Agent

The following are the supported OS.

- AIX 4.3.3, 5.x 32bit, 64bit
- HP-UX 11.x 32bit, 64bit, Itanium 64bit
- Sun Solaris 2.8, 2.9, 10 32bit, 64bit, x68

- Intel Linux 32bit, Redhat Itanium 64bit
- Compaq Tru64 UNIX OSF1
- Microsoft Windows 2000, XP, 2003, Vista
- IBM iSeries(AS400) for WebSphere
- IBM z/OS for WebSphere, zLinux

The JENNIFER Agent can only monitor the Java applications using Java 1.3 or higher. The following are the supported application servers.

- BEA WebLogic 5.1, 6.x, 8.x, 9.x, 10.x
- IBM WebSphere Application Server 3.5, 4.x, 5.x, 6.x
- TmaxSoft JEUS 3.x, 4.x , 5.x
- Oracle Application Server 9i AS, 10g AS, OC4J, ERP
- SUN Application Server 7.x, 8.x
- Fujitsu Interstage 5.x, 6.x, 7.x, 9.x
- Hitachi Cosminexus 7
- Sybase EAServer 4.x, 5.x
- Macromedia JRun 4.x
- Apache Jakarta Tomcat 3.x, 4.x, 5.x, 6.x
- Caucho Technology Resin 2.x, 3.x
- RedHat JBoss Application Server 3.x, 4.x
- Apache JServ

1.5.2. JENNIFER Server

The following are the minimum hardware specifications for running the JENNIFER Server under the conditions : monitoring less than 10 of JENNIFER Agents with maximum 100 TPS(Transaction Per Second), storing the collected performance data for a month.

- More than 2 CPU
- More than 2 GB of Memory
- More than 300GB of Hard Disk storage
- More than 100 Mbps of broadband network between the JENNIFER agent and the JENNIFER server.

In addition, you have to download the Java 1.5 or higher to install JENNIFER server.

1.5.3. JENNIFER .NET Agent

The following are the supported operating system.

- Windows Server 2003 32bit, 64bit
- Windows Server 2008 32bit, 64bit
- Windows Server 2008 R2

.NET

- 2.0 or higher

1.5.4. JENNIFER Client

In order to use the JENNIFER Client, you have to install Microsoft Windows XP or higher and web browser Microsoft IE 6 and IE 7, and Mozilla Firefox 3.0 or higher. Java Plug-in 1.6 should be installed as well.

The following are the minimum hardware specifications for running the JENNIFER Client under the conditions : monitoring less than 10 of JENNIFER Agents with maximum 100 TPS(Transaction Per Second).

- More than 1.8 GHz of CPU
- More than 2 GB of Memory

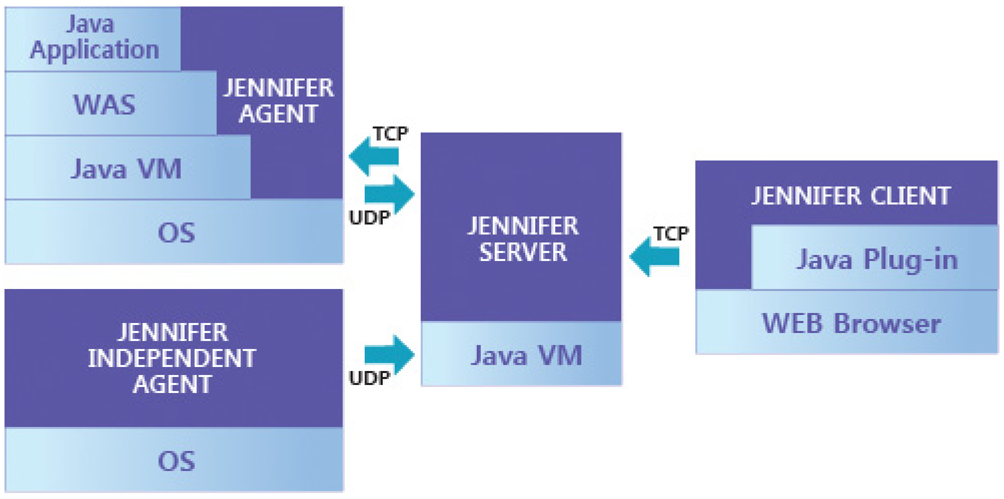
2 JENNIFER Architecture

The basic goal of the JENNIFER architecture is to perform real-time monitoring of the enterprise Java application without causing any load to the target systems.

2.1. Basic JENNIFER Architecture

The JENNIFER architecture is comprised of a JENNIFER agent, a JENNIFER independent agent, a JENNIFER server, and a JENNIFER client.

Figure 2-1:JENNIFER Architecture Diagram



The JENNIFER agent and the JENNIFER independent agent collect various performance data and send them to the JENNIFER server. The JENNIFER server processes the received data and stores them in a file and/or database. The JENNIFER server also manages configuration information such as users, privileges, menu, etc. Finally, the JENNIFER client provides a web-based user interface that enables intuitive monitoring of the performance data.

2.1.1. JENNIFER Agent

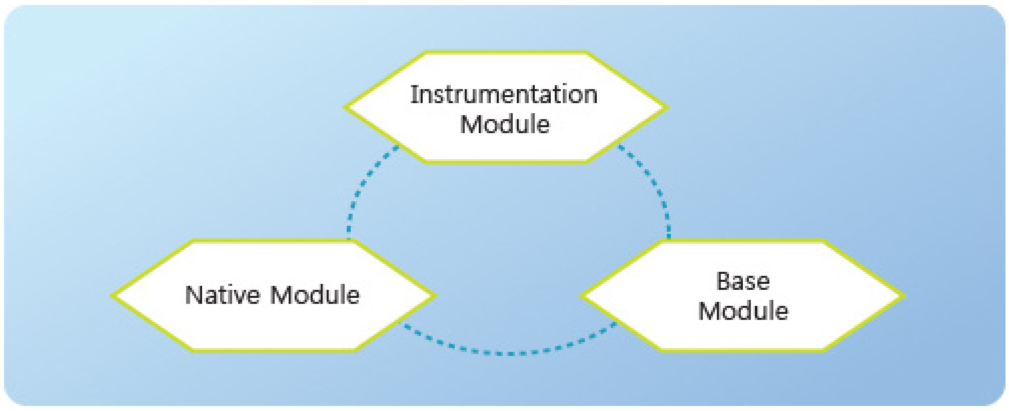
Along with the JENNIFER independent agent, the JENNIFER agent collects various performance data and sends them to the JENNIFER server. The performance data collected by the JENNIFER agent comes from the Java application. In general, the Java application is a web-based application that operates on the WAS.

Notice: In addition to the Java application operating on the WAS, the independent Java applications can be monitored by the JENNIFER agent. However, this is possible only for applications that operate on JDK 1.3 or higher.

Therefore, the JENNIFER agent is not an independent process, but operates in an embedded environment with the Java application. The JENNIFER agent and the Java application operate on the same Java Virtual Machine (JVM).

The JENNIFER agent is comprised of three modules -instrumentation, native and base modules.

Figure 2-2:JENNIFER Agent's Module



- Instrumentation module - The instrumentation module injects the tracking code for performance data collection and the profile information extraction code into the classes constituting the Java application, without changing the source code of the monitored Java application. This is the role of the instrumentation module, which is sometimes referred to as the LWST (Light Weight Stack Trace) module.
- Native module - The native module is a C library invoked by the JNI (Java Native Interface). This module is responsible for collecting information relating to CPU and system memory usage by the Java application in which the JENNIFER agent is installed. The native module is compiled in accordance with the OS, and is included in the JENNIFER package as an 'so' or 'dll' file.
- Base module - The base module collects various performance data, and processes it, and sends it to the JENNIFER server.

2.1.2. JENNIFER .NET Agent

The JENNIFER .Net Agent is responsible for collecting various performance data and sending them to the JENNIFER Server.

The complied module consists of DLL and if the web application operating at the IIS(Internet Information Services) is to be monitored, it operates altogether with the web application in the same address space of the w3wp.exe process.

The JENNFIER .Net Agent is internally comprised of two modules, in other words, the NET profiler com object that directly changed the IL code and the .Net agent module that is responsible for communication with the JENNIFER Server.

- .NET profiler com module – It is a native module that implements the profiler com interface provided by the CLR and changes the IL code.
- .Net agent module - .When executed by the NET profiler com object, it is connected to the application program which is the dynamic profiling target in order to collect and

process various performance data in the unit of AppDomain and send them to the JENNIFER Server.

2.1.3. JENNIFER Independent Agent

The JENNIFER agent operates internally, in combination with the monitored Java application. However, it is often necessary to collect performance data from a region outside the Java application. For instance, the hardware CPU usage by the business database or the results of executed script in an arbitrary OS should also be collected as performance data. The JENNIFER independent agent supports such tasks.

Notice: The JENNIFER agent collects performance data according to certain rules and formats. In contrast, the JENNIFER independent agent collects non-standard performance, data without pre-arranged formats or rules.

The JENNIFER independent agent does not refer to a specific entity, but instead logically refers to modules like WMOND, REMON and LogWatcher.

2.1.3.1. WMOND

This module collects the system CPU usage information of arbitrary hardware. This module is developed in a C language, and it is very simple to use.

2.1.3.2. REMON

The method for collecting non-standard performance data varies depending on the source of the data. However, REMON module can be used to manage the collected data and send it to the JENNIFER server. The REMON module provides capability to flexibly expand the methods of performance data collection and to send the performance data to the JENNIFER server.

For instance, you can use the REMON module to execute an arbitrary script in an arbitrary OS, or execute an arbitrary SQL in an arbitrary database, or to execute the Java class with a specific interface. Through these methods, you can easily collect non-standard performance data and send it to the JENNIFER server.

Notice: REMON is an application implemented in the Java language.

2.1.3.3. LogWatcher

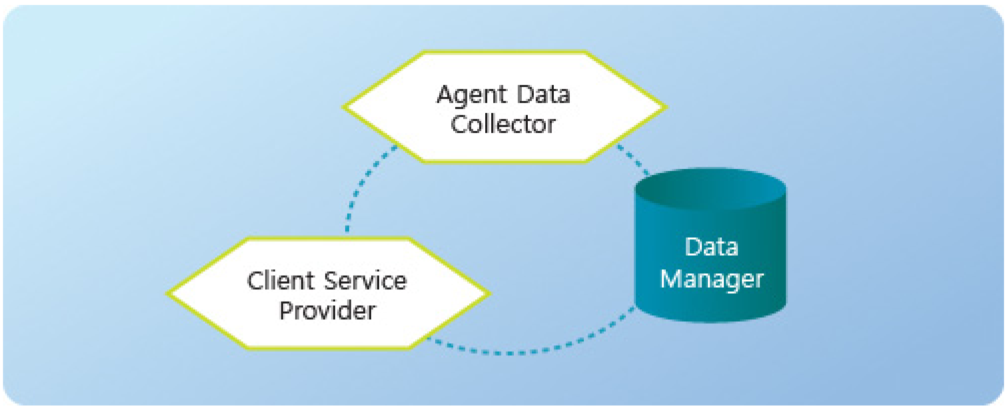
This module plays the role of a log monitor. It searches a text log file for a designated pattern, and generates an event to send the relevant data to the JENNIFER server. The user can then monitor it through the JENNIFER server.

2.1.4. JENNIFER Server

The JENNIFER server processes the performance data that it receives from the JENNIFER agent and the JENNIFER independent agent, and saves it to a file and/or database. In addition, it manages the configuration of the menu, the user and the privileges.

A single JENNIFER server can handles multiple JENNIFER agents. The JENNIFER server is designed to process large amount of performance data. JENNIFER server is comprised of three internal modules - the Agent Data Collector, the Data Manager and the Client Service Provider.

Figure 2-3:JENNIFER Server's Module



- Agent Data Collector - The agent data collector receives the performance data that has been sent by the JENNIFER agent and the JENNIFER independent agent through the UDP, and repeatedly sends a TCP request to the individual JENNIFER agent every ten minutes, so that it can collect the statistical data regarding the status of application processing.
- Data Manager - The data manager saves the performance data collected by the agent data collector module in a file and in a database. By default, the JENNIFER server uses a built-in database (Apache derby), but depending on the situation, it may use an Oracle database, IBM DB2, and others instead.
- Client Service Provider - The client service provider provides the necessary service and data for the JENNIFER client to organize the user interface.

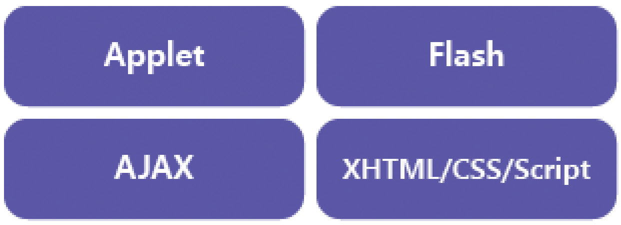
In addition to the three main modules above, the JENNIFER server includes the Scheduler, Reporter and Organization Management modules.

2.1.5. JENNIFER Client

JENNIFER provides a user interface for intuitive monitoring of the performance data, which is called the JENNIFER client. To improve accessibility, the JENNIFER agent is implemented with XHTML, CSS, Java scripts and AJAX and a user can use the displayed through the web browser such as Internet Explorer or Mozilla Firefox.

The JENNIFER Client generates and displays various charts, graphs and tables using the Java applets, so that the user can easily monitor the changes in performance data in real time. JENNIFER client requires a web browser with Java 6.0 or higher.

Figure 2-4:Implementation of the JENNIFER Client



Notice: To ensure that the client is compatible with the majority of the available web browsers, the JENNIFER client does not use MS ActiveX technology.

If the clock in the JENNIFER server and in the JENNIFER client is different, the JENNIFER client displays the time of the JENNIFER server.

2.2. Performance Data Collected by JENNIFER

This chapter provides an overview of the performance data that the JENNIFER server collects from the JENNIFER agent and the JENNIFER independent agent.

2.2.1. Source of Data

The JENNIFER server collects various performance data from the JENNIFER agent and the JENNIFER independent agents, such as WMOND, REMON and LogWatcher.

- JENNIFER Java/ .NET agent - The JENNIFER agent collects various general performance data such as service request rates, mean response time and Java heap memory usages, as well as profile data, status of processing of active service, SQL and external transactions, alarms and exception data and X-view transactions.

- WMOND - The WMOND module collects an arbitrary system CPU usage.
- REMON - The REMON module collects non-standard data. There is no limitation on the format or source of such data.
- LogWatcher -The LogWatcher module collects main events from an arbitrary text log file.

2.2.2. Saving Data

The JENNIFER server saves the collected performance data in a file and/or database.

The JENNIFER server first saves large-sized data, such as profile information and items related to the execution of individual transactions in a file. Most of the remaining performance data is saved in a database.

2.2.3. Performance Data Collected by the JENNIFER Agent

The following is a list of the various types of performance data collected by the JENNIFER agent.

2.2.3.1. General Performance Data

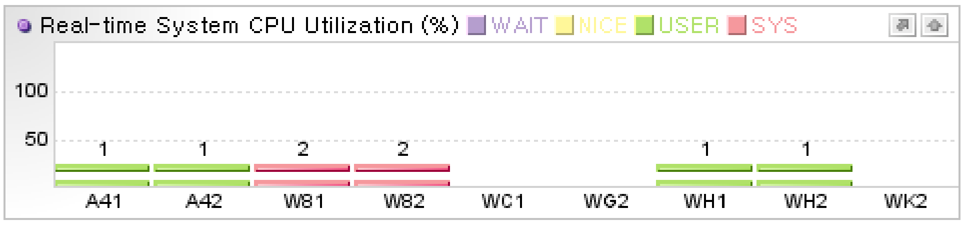
Service performance data is related to user and service processing, such as the number of concurrent users or the service request rate, while resource performance data is related to system status items such as CPU and memory usage. Together, the service and resource performance data are referred as the general performance data.

The JENNIFER agent collects the general performance data once every second and sends it to the JENNIFER server. The collected data reflects the mean value over the thirty-seconds period just prior to the moment of data collection, For instance, the mean response time collected at 10:28:45 is the mean value from 10:28:15 to 10:28:45, while the mean response time collected at 10:28:46 is the mean value from 10:28:16 to 10:28:46..

Notice: However, for cumulative quantitative data, such as the number of calls or the number of concurrent users, the accumulated value during the interval is used, rather than the mean value. In addition, the number of active services and that of JDBC connections is the value of the certain time period, not an mean value of 30-sec.

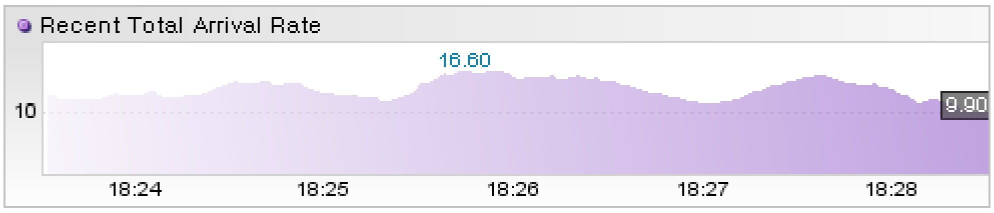
The reason behind using a mean value rather than an point value is that to statistically adjust the discrepancy that fluctuates after time of data collection. The collected data representing the 30-sec mean value can be viewed in an equalizer chart or in a run-time line chart in real time.

Figure 2-5:Equalizer Chart



The equalizer chart represents the 30-sec mean value recently collected. But, the number of active services and that of JDBC connections is the value of the certain time period, not an average value of 30-sec.

Figure 2-6:Run Time Line Chart



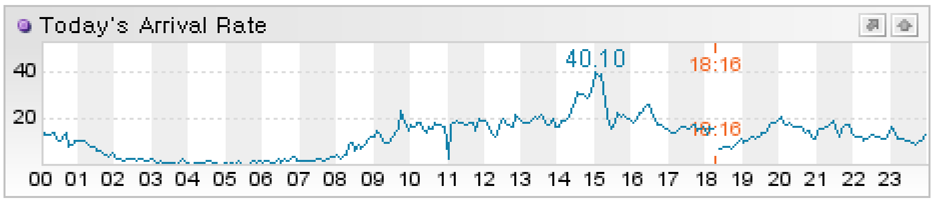
The run time line chart represents the 30-sec mean values collected over a five-minute period. The run time line chart displays the 30-sec mean value of the number of active services but, it represents the number of JDBC connections collected over a certain time period.

However, the JENNIFER server does not save the 30-sec mean value collected from the JENNIFER agent at one-second intervals, as this would overload the database and degrade performance. Consequently, the JENNIFER server re-processes the 30-sec mean values into 5-min mean values, and stores them in the database every 5 minutes.

Notice: The JENNIFER server saves the general performance data in the PERF_X_01~31 table. If you partition a 24-hour period into five-minute segments, you have 288 entities. Therefore, each JENNIFER agents stores 288 sets of general performance data in the PERF_X_01~31 table.

The 5-min mean data can be viewed in a line chart.

Figure 2-7:Line Chart



The line chart represents a total of 288 five-min mean values in one line.

Service Performance Data

The following is the description of service performance data.

Table 2-1: Service Performance Data

Performance Data	Descriptions
# of concurrent users	This performance data can only be collected from a Java web application that supports cookies (client is a web browser), and it represents the total number of users who are simultaneously using the Java web application. Since a unique cookie is assigned to each user in the clustering environment, no user is counted twice.
# of active services	This value represents the total number of services being processed by the Java application.
# of active users	It is possible that some services may share the same users. This value assumes that if multiple services are provided to the same user, the user is only counted once.
Arrival rates	The rate of incoming requests per second
Service rate	The rate of service processing per second
Average response time	The unit is in seconds. The total response time is divided by the number of requests to yield the value.
Think time	The mean difference between the most recent time of service request and the current time of service request.
# of calls	The number of service requests by the user
Daily visitors	This implies the number of unique visitors per day.
Hourly visitors	This implies the number of new visitors in a given time interval.

Notice: In general, the user is a web browser process. Therefore, if the same computer uses multiple web browser processes to call the Java application, the JENNIFER agent assumes that each web browser process is a different user. Also, after closing the web browser, if a new web browser is launched to call the Java application, the JENNIFER agent will considers it to be another user.

For more details on the service performance data, please refer to [Service performance monitoring].

Resource Performance Data

The following is the description of the resource performance data.

Table 2-2: Resource Performance Data

Performance Data	Description
System CPU usage	This is the system CPU usage that the JENNIFER agent collects. It is expressed as percentage (%).
Process CPU usage	This is the process CPU usage that the JENNIFER agent collects. It is expressed as percentage (%).
System memory usage	This is the system memory usage that the JENNIFER agent collects. It is expressed in MB.
Process memory usage	This is the amount of memory the JENNIFER agent collects. It is different from the heap memory usage.
Total heap memory usage	This is the full capacity of Java heap memory. It is expressed in MB.
Total physical memory usage	This is the total physical memory. It is expressed in MB.
Process heap memory usage	This is the heap memory usage. It is expressed in MB.
# of standby DB connections	The number of DB connections waiting in the DB connection pool.
# of assigned DB connections	This is the number of DB connections that are not currently performing SQL, of all the DB connections assigned to the Service thread by the DB connection pool.
# of DB connections in use	This is the number of DB connections that are currently performing SQL queries among all the DB connections that are assigned to the Server thread by the DB connection pool.

For more details on the resource performance data, please refer to [Resource and JDBC monitoring].

2.2.3.2. Statistical Data on the Status of Application Processing

The JENNIFER agent collects statistical data on the status of the processing active services, applications, SQL, external transactions, exceptions and JDBC. This data is integrated and collected every 10 minutes, and can be divided into real-time processing data for the most recent 10-minute interval and previous processing data for interval before then. After 10 minutes have elapsed, the real-time processing data is saved in the database.

Notice: Statistical data related to active services and JDBS is not stored in the database.

The data mentioned here is a summary of statistics for the most recent 10-minute interval. For instance, it provides information such as the total number of calls and the mean response time of the login.jsp application over a 10-minute period, but does not provide information on the request and response time for individual login.jsp transactions. If you want to analyze individual transactions, please refer to [X-view and Profiling].

Notice: The real-time processing data does not mean 10 minutes upto the present moment. It only means that the time is partitioned into 10-minute segments. Therefore, if the current time is 09:18, then the real-time processing data will show the performance data from 09:10 to 09:20.

2.2.3.3. Alerts and Exception Data

The JENNIFER agent detects exceptions that occur in the Java application, and analyzes the condition of the Java application to generate appropriate alerts. The alerts have three levels - critical, error and warning. The user can check the alerts through the JENNIFER client in real time, and analyze the content of past alerts.

For more details on alerts and exception data, please refer to [Alerts and Monitoring].

2.2.3.4. X-View Transactions and Profile Data

The JENNIFER agent collects the transaction information processed by the Java application and provides the user with the information on an X-view chart. The transaction-related data collected by the JENNIFER agent is called X-view data, and it is categorized into transaction data and profiling data.

For more details on X-View transactions and profiling data, please refer to [X-view and Profiling].

2.2.4. Performance Data Collected by WMOND

The WMOND module collects an arbitrary system CPU usage data. For more details on the WMOND, please refer to [System CPU monitoring (WMOND)].

Notice: The WMOND module classifies each of the physically separated CPUs, and collects CPU usage data from each CPU. The JENNIFER agent collects the logical CPU usage data for the entire computer.

2.2.5. Performance Data Collected by REMON

The REMON module collects various types of non-standard performance data. To learn about collecting non-standard performance data using REMON, please refer to [REMON]. For the data monitoring method, please refer to [User defined dashboard].

2.2.6. Performance Data Collected by LogWatcher

The LogWatcher module searches an arbitrary text log file for a designated pattern, and collects the relevant events. For more details on LogWatcher, please refer to [Log monitor (LogWatcher)].

2.3. Network Organization and Key Java Threads

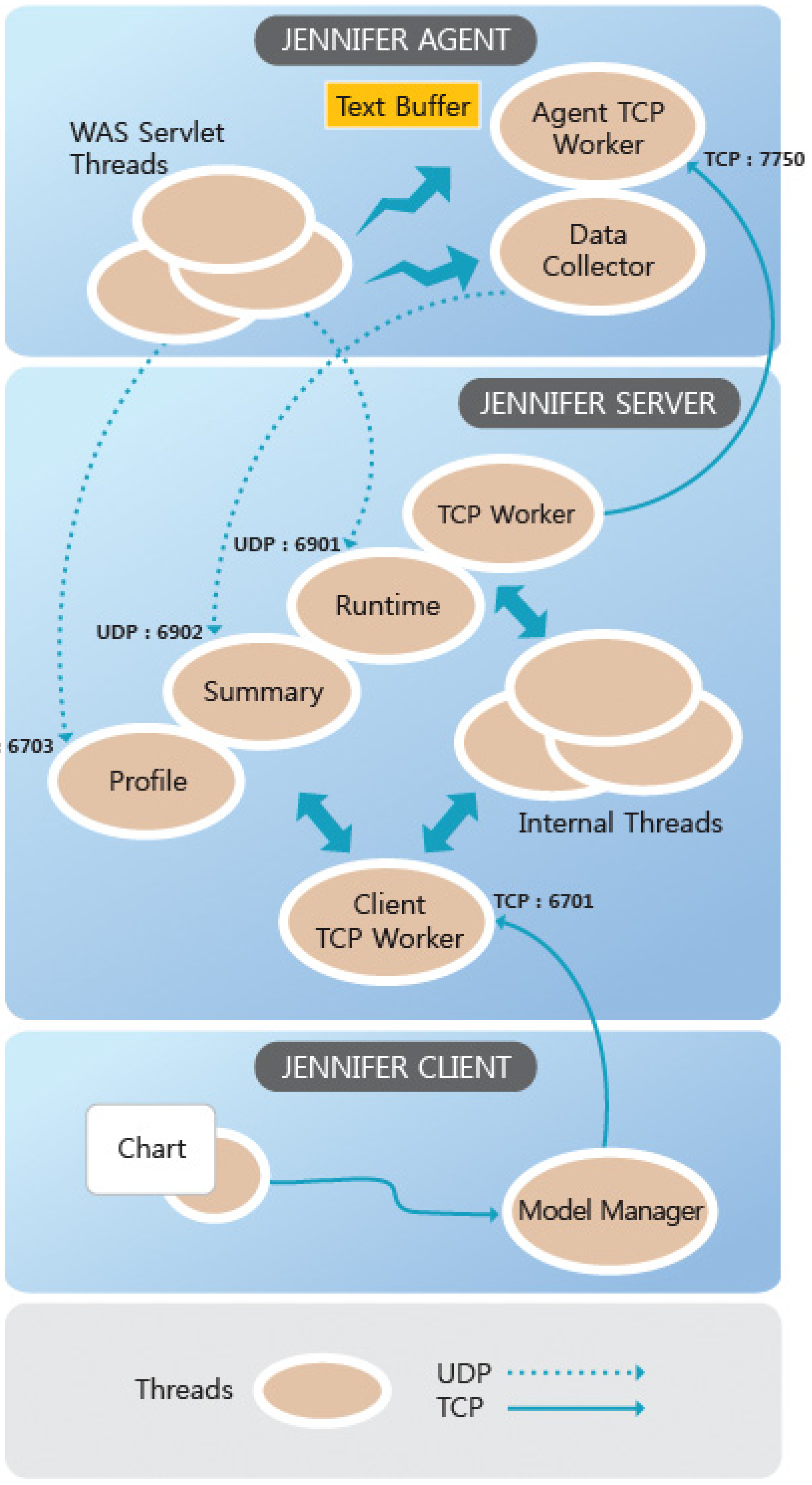
The JENNIFER agent, the JENNIFER independent agent and the JENNIFER client that constitute JENNIFER are processes that operate independently from each other. Therefore, data transmission between these modules is achieved through UDP or TCP network communication.

Notice: Even if you have installed the JENNIFER server and the JENNIFER agent in the same computer, data communication between them is achieved through network communication.

Data communication is only achieved between the JENNIFER agent and the JENNIFER server, the JENNIFER independent agent and the JENNIFER server, and the JENNIFER server and the JENNIFER client. There is no data communication between the JENNIFER client and the JENNIFER agent, or between the JENNIFER agent and the JENNIFER agent.

Various Java threads exist in the JENNIFER agent and server that can process the received data.

Figure 2-8:Network and Java Thread Structure in JENNIFER



2.3.1. JENNIFER Agent and JENNIFER Server

The JENNIFER agent sends performance data to the JENNIFER server in UDP. The JENNIFER server receives the performance data through three separate UDP ports. Each data port receives unique performance data.

The JENNIFER agent sends the data about the beginning and end of every transaction to the UDP port set by the server_udp_runtime_port option of the JENNIFER server. This data is very small, and is used to display the X-view chart. The default port number is 6901.

```
server_udp_runtime_port = 6901
```

The JENNIFER server's Java thread that processes the performance data coming from the port set by the server_udp_runtime_port option of the JENNIFER server is called Runtime UDP Worker. The number in Runtime UDP Worker is set by the number_of_udp_runtime_workers option of the JENNIFER server. The default is 10.

```
number_of_udp_runtime_workers = 10
```

The JENNIFER agent sends general performance data such as the service request rate and the mean response time to the UDP port set by the server_udp_listen_port option of the JENNIFER server once every second. The default port number is 6902.

```
server_udp_listen_port = 6902
```

The JENNIFER server's Java thread , which processes the performance data coming from the port set by the server_udp_listen_port option of the JENNIFER server, is called Summary UDP Worker. The number in Summary UDP Worker is set by the number_of_udp_listen_workers option of the JENNIFER server. The default is 10.

```
number_of_udp_listen_workers = 10
```

The JENNIFER agent sends the X-view transaction profile data to the UDP port set by the server_udp_lwst_call_stack_port option of the JENNIFER server every two seconds. The default port number is 6703.

```
server_udp_lwst_call_stack_port = 6703
```

And the JENNIFER server's Java thread, which processes the performance data coming from the port set by the server_udp_lwst_call_stack_port option of the JENNIFER server is called Profile UDP Worker. The number in Profile UDP Worker is set by the number_of_udp_callstack_workers option of the JENNIFER server. The default is 30..

```
number_of_udp_callstack_workers = 30
```

If the size of the performance data that the JENNIFER agent sends to the JENNIFER server in UDP exceeds the max UDP transmission value set in the OS, the data is not sent, and is lost. However, the error messages will be saved in a log file. If there is an additional network located between the JENNIFER agent and server, and the size of the performance data sent to the JENNIFER server exceeds the max UDP transmission value, then no error message will be recorded, and the data will be lost.

Notice: The max data size that can be sent in UDP is (UDP Send Buffer Size) 64 KB. The default value for Sun Solaris and HP HP-UX is 64kB but a smaller value is used for IBM AIX.

Various options, including server_udp_runtime_port, server_udp_listen_port and server_udp_lwst_call_stack_port should be set in both the JENNIFER server and agent. According to these options, the JENNIFER server opens the UDP to receive the performance data from the JENNIFER agent and the JENNIFER agent designates the port of the JENNIFER server to transmit/receive the performance data.

In addition, it is necessary to set the JENNIFER server's IP address using the udp_server_host option of the JENNIFER agent.

```
udp_server_host = localhost
```

JENNIFER puts a higher priority on the reliable monitoring of target Java applications than reliable data transmission. Therefore, even if data transmission fails, it will not send the data again.

Sometimes, the JENNIFER server establishes a reverse connection to the TCP port of the JENNIFER agent. This setting is mostly used for searching for large data that is not stored in the database or file, such as the active service lists, the10-min application processing status, the loading class list and the socket/file lists. Using the agent_tcp_port option of the JENNIFER agent, you can set the receiving port of the JENNIFER agent for the TCP request from the server. The default port number is 7750.

```
agent_tcp_port = 7750
```

The JENNIFER agent's Java thread that processes the JENNIFER server's request coming through the port set by the agent_tcp_port of the JENNIFER agent is called Agent TCP Worker. The number in Agent TCP Worker can be set by the number_of_tcp_workers option of the JENNIFER agent. The default value is 5.

```
number_of_tcp_workers = 5
```

This is called a reverse connection because the network connection is made for the JENNIFER server that searches for the data, not for the JENNIFER agent.

In the event that the reverse TCP connection from the JENNIFER server to the JENNIFER agent fails, you may set the timeout option using agent_tcp_io_timeout of the JENNIFER server. The default value is 5000 and the unit is in milliseconds.

```
agent_tcp_io_timeout = 5000
```

2.3.2. JENNIFER Independent Agents and JENNIFER Servers

The JENNIFER server receives the performance data from the JENNIFER independent agent through the UDP port.

Therefore, the WMOND, REMON and LogWatcher modules send the performance data to the UDP port set by the server_udp_listen_port option of the JENNIFER server.

2.3.3. JENNIFER Servers and JENNIFER Clients

To provide a web-based user interface, the JENNIFER server and client use the HTTP protocol. The default port number is 7900.

And the Java applet receives the data necessary for organizing the charts through TCP communication from the JENNIFER server. The default TCP port number for the JENNIFER server is 6701, and it is set by the server_tcp_port option of the JENNIFER server.

```
server_tcp_port = 6701
```

The JENNIFER server's Java thread that processes the JENNIFER client's requests coming through the port set by the server_tcp_port option of the JENNIFER server is called Client TCP Worker. The number in Client TCP Worker is set by the number_of_tcp_pooled_workers option of the JENNIFER server. The default value is 80.

```
number_of_tcp_pooled_workers = 80
```

However, there is no TCP call from the JENNIFER server to the JENNIFER client.

2.3.4. Cautions for Network Organization

The following is a list of items to watch for in network organization.

- Avoid duplicate use of the same port - If multiple JENNIFER agents are installed in the same computer, then you should use the agent_tcp_port option of the JENNIFER agent to prevent duplicate use of the port. If multiple JENNIFER agents are installed in the same hardware, then use the following options to prevent duplicate uses: server_udp_runtime_port, server_udp_listen_port, server_udp_lwst_call_stack_port, server_tcp_port .

- Check the firewall - If there is a firewall between the main modules of JENNIFER, you should check to ensure that a port is opened properly.

2.4. Features of JENNIFER Architecture

The main features of the JENNIFER architecture are as follows.

- The JENNIFER architecture minimizes the load that occurs as a result of monitoring, and provides an effective means to inject tracking code without changing the source code.
- To reduce the network overload and acquire operational reliability, it is critical to minimize the amount of performance data that is monitored. Therefore, we prioritized performance data in term of usefulness to enterprise Java application monitoring. The performance data is collected on the basis of the priority.
- To efficiently utilize the network resource, the UDP method is used to collect data, and the active service list is browsed in real time. To analyze the causes of performance problems is delivered using reverse TCP connection.
- The JENNIFER architect provides a data storage structure that is suitable for efficient use of storage space and effective performance data analysis.
- The JENNIFER architect provides a flexible JENNIFER independent agent that helps to collect non-standard performance data from various systems and applications, as well as from Java applications.
- JENNIFER provides a web-based user interface for intuitive monitoring of the performance data.

3 Java Agent Management

This chapter describes the main components for the JENNIFER agent.

3.1. Java Agent Configuration File

All JENNIFER agent configurations for the Java version, including monitoring standards and methods, and network port number assignments, are set by the JENNIFER agent options. The JENNIFER agent configuration file is the file designated by the Java -Djennifer.config option when the JENNIFER agent is installed.

3.1.1. Format of the Configuration File

The JENNIFER agent option for the Java version is saved in a text formatted configuration file. The file has the following characteristics:

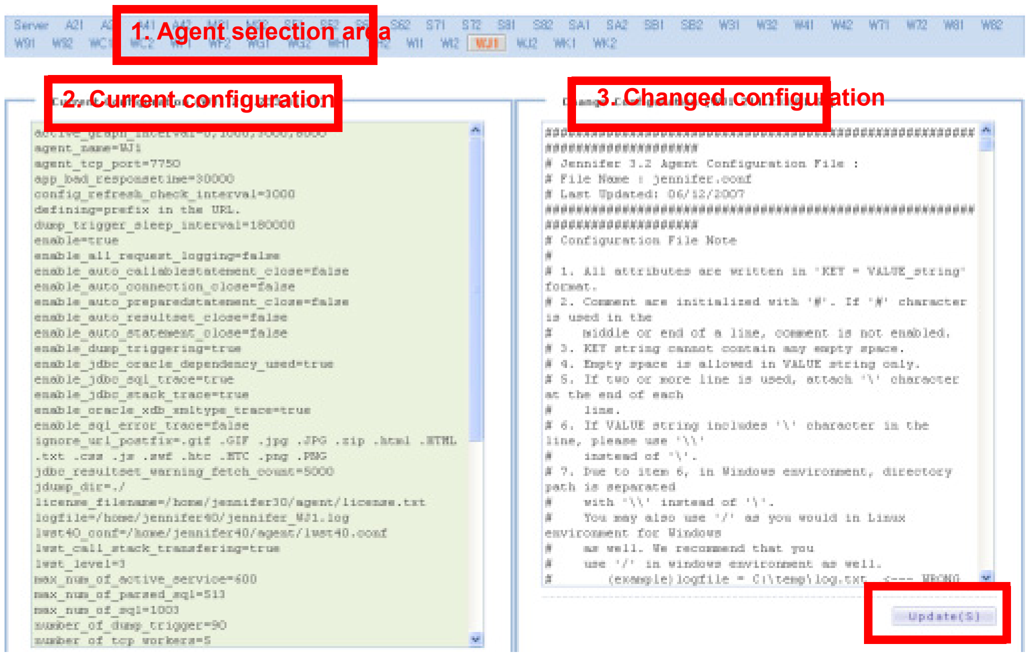
- The format is ‘key = value’. The key represents the JENNIFER agent option.
- Any lines starting with [#] are comments. However, the appearance of a [#] symbol in the middle of a text string will not indicate a comment.

- Instead of [=], you can use ‘SPACE’ to separate the key and the value.
- For the above reason, it is prohibited to use ‘SPACE’ as the key.
- However, you can include ‘SPACE’ in the value.
- If you need to use more than one line, add [\] at the end of the line. If you need to use [\] for text, then use [\\] to finish the line.
- For the reason shown above, you must use [\\], not [\] to distinguish the directory in the Microsoft windows environment. However, you can use [/] in Windows, as in the Unix environment, and you are recommended to do so.
- With a few exceptions, options change is reflected in the system as soon as the change is made. In addition, the option that can be used when you restart Java application installed the JENNIFER agent, is also described.

3.1.2. Changing the Configuration

The configuration file can be directly modified using a text editor, or using the **[Properties | Configuration]** menu in the JENNIFER client. However, only the users in Administrator group are allowed to modify the configuration options through the JENNIFER client.

Figure 3-1:JENNIFER Option Setting Screen



Server A21 A2 A41 A42 M21 M22 S51 S52 S6 S62 S71 S72 S81 S82 SA1 SA2 SB1 SB2 W31 W32 W41 W42 W71 W72 W81 W82
W91 W92 WC1 WC2 WF1 WF2 WG1 WG2 WH1 WH2 WI1 WI2 WJ1 WJ2 WK1 WK2
1. Agent selection area
2. Current configuration
3. Changed configuration
Update(S)

```
active_graph_interval=0,1000,3000,5000
agent_name=WJ1
agent_tcp_port=7750
app_bad_responsetime=30000
config_refresh_check_interval=3000
defining=prefix in the URL.
dump_trigger_sleep_interval=180000
enable=true
enable_all_request_logging=false
enable_auto_callablestatement_close=false
enable_auto_connection_close=false
enable_auto_preparedstatement_close=false
enable_auto_resultset_close=false
enable_auto_statement_close=false
enable_dump_triggering=true
enable_jdbc_oracle_dependency_used=true
enable_jdbc_sql_trace=true
enable_jdbc_stack_trace=true
enable_oracle_xdb_xmltype_trace=true
enable_sql_error_trace=false
ignore_url_postfix=.gif .GIF .jpg .JPG .zip .html .HTML
.txt .css .js .swf .htc .HTC .png .PNG
jdbc_resultset_warning_fetch_count=5000
jdump_dir=./
license_filename=/home/jennifer30/agent/license.txt
logfile=/home/jennifer40/jennifer_WJ1.log
lwst40_conf=/home/jennifer40/agent/lwst40.conf
lwst_call_stack_transfering=true
lwst_level=3
max_num_of_active_service=600
max_num_of_parsed_sql=513
max_num_of_sql=1003
number_of_dump_trigger=90
number of tcp workers=5
```

```
############################################################
####################
# Jennifer 3.2 Agent Configuration File :
# File Name : jennifer.conf
# Last Updated: 06/12/2007
############################################################
####################
# Configuration File Note
#
# 1. All attributes are written in 'KEY = VALUE_string'
format.
# 2. Comment are initialized with '#'. If '#' character
is used in the
#    middle or end of a line, comment is not enabled.
# 3. KEY string cannot contain any empty space.
# 4. Empty space is allowed in VALUE string only.
# 5. If two or more line is used, attach '\' character
at the end of each
#    line.
# 6. If VALUE string includes '\' character in the
line, please use '\\'
#    instead of '\'.
# 7. Due to item 6, in Windows environment, directory
path is separated
#    with '\\' instead of '\'.
#    You may also use '/' as you would in Linux
environment for Windows
#    as well. We recommend that you
#    use '/' in windows environment as well.
#       (example) logfile = C:\temp\log.txt  <--- WRONG
```

1. Agent selection area - To check or change the JENNIFER agent option, select the JENNIFER agent here.

2. Current configuration - In the current configuration on the left, review the JENNIFER agent option.
3. Changed configuration - Modify the existing option, or add a new one in the configuration change input form at the right. Click the [change] button at the bottom to apply the change.

You can use the config_refresh_check_interval option of the JENNIFER agent to periodically review the changes in the JENNIFER agent configuration file. It is expressed in Milliseconds.

```
config_refresh_check_interval = 3000
```

3.1.3. Making a New Configuration File

If you want to make a new JENNIFER agent configuration file, copy the existing configuration file (ex. w11.conf) in the JENNIFER_HOME/agent directory, and change the following option values:

```
agent_name = W11

agent_tcp_port = 7750
```

If you want to monitor more than one Java application on the same hardware, you must assign a new JENNIFER agent ID and use a different agent TCP listening port number.

However, if you need to monitor dozens of Java applications, it can be very tedious to copy each file and make a new configuration file. Therefore, the following utility can be used:

```
confutil.sh
```

The $JENNIFER_HOME/agent/confutil.sh utility reads the w11.conf file from the current directory, and automatically creates up to 50 configuration files, from c01.conf to c50.conf.

Notice: If you want to change the file name to be created or the number, modify the confutil.sh file directly.

3.2. Default Configuration and Management of the JENNIFER Java Agent

This section describes the exceptions that can occur during the JENNIFER agent installation and the options that need to changed after JENNIFER agent is installed.

3.2.1. JENNIFER Agent ID

A single JENNIFER server can collect performance data from multiple JENNIFER agents. For this reason, each JENNIFER agent should have a unique ID, so that the JENNIFER server can distinguish the performance data from different agents. This can be set using the agent_name option of the JENNIFER agent.

```
agent_name = w11
```

Warning: The JENNIFER agent ID should be composed of alphanumeric characters only, and must be three characters in length.

When the JENNIFER agent ID is duplicated, the JENNIFER client will show that the JENNIFER agent has been repeatedly stopped within a short time period, or it can issue the ERROR_MAYBE_GC_TIME_DELAY alert. In this case, you must change one of the duplicated JENNIFER agent ID's, and delete the [Agent list] and the [Real time data list] in the **[Properties | configuration | Real time management]** menu.

Notice: The change in the JENNIFER agent ID is reflected in real time.

3.2.2. Log File

The JENNIFER agent log file can be set using the logfile option of the JENNIFER agent.

```
logfile = jennifer.log
```

By default, the log is recorded in the WORKING_DIRECTORY/jennifer.log file. This log file contains exceptional cases, service dumps and debugging content that occur in the application.

If you want to record daily log files, set the enable_logfile_daily_rotation option of the JENNIFER agent to 'true'.

```
enable_logfile_daily_rotation = true
```

Using the logfile_encoding_characterset option of the JENNIFER agent, you can set the text encoding method for the JENNIFER agent log file.

In addition, the LWST-related log files can be set using the lwst_logfile option of the JENNIFER agent.

```
lwst_logfile = lwst.log
```

By default, the log is recorded in the WORKING_DIRECTORY/lwst.log file. This log file contains the class and method information set in the tx-server, tx-naming, tx-client and profile options of the JENNIFER agent, and the LWIST debugging contents.

3.2.3. License Key File

The JENNIFER agent license key is stored in a text file set by the license_filename option of the JENNIFER agent.

```
license_filename = license.txt
```

By default, the license key is saved in the WORKING_DIRECTORY/license.txt file.

3.2.4. License Key and JENNIFER Agent enablement

If you adjust invalid JENNIFR license key due to the wrong IP number or expired one, or set the enable option of JENNIFER agent to ‘false’, the JENNIFER agent runs as follows:

All functions of the JENNIFER agent (except for LWST) may stop, even though you restart the Java application. In other worlds, JENNIFER does not collect and send the performance data any more. Of course, this does not give any impact on Java application operation.

Although you do not restart Java application, LWST module operate properly as it runs before. Because the LWST module needs bite code change to run while loading the class. That means you can not stop all LWST functions including profiling process for the loaded class. Of course, if you restart the Java application, all LWST functions would stop.

3.2.5. Network Configuration

The JENNIFER agent sends the collected performance data to the JENNIFER server. In this section, we will describe the network configuration used for this task.

3.2.5.1. TCP Network Configuration

The JENNIFER server makes the reverse connection to the TCP port of the JENNIFER agent. JENNIFER usually uses the TCP reverse connection when it searches the large-scale of performance data such as active service lists, statistical data of application processing status for 10 minutes, loaded class lists, and socket/file lists that are not stored in the DB. Therefore, using agent_tcp_port of JENNIFER agent, set the port number of JENNIFER agent that allows the TCP request from JENNIFER server. The default port number is 7750.

```
agent_tcp_port = 7750
```

Agent TCP Worker is the Java thread of JENNIFER agent which processes such JENNIFER server's request. Using number_of_tcp_workers, set the number of Agent TCP Worker. The default value is 5.

```
number_of_tcp_workers = 5
```

If you want to change these two options, restart the Java application that installed JENNIFER agent.

3.2.5.2. UDP Network Configuration

The JENNIFER agent sends performance data to the UDP port of the JENNIFER server. First, using the udp_server_host option of the JENNIFER server, set the IP address for the JENNIFER server.

```
udp_server_host = 127.0.0.1
```

Use the UDP port number of the JENNIFER server to set a range of options for the JENNIFER agent.

```
server_udp_runtime_port = 6901
server_udp_listen_port = 6902
server_udp_lwst_call_stack_port = 6703
```

If these two options are modified, the changes will be applied without restarting the Java application in which the JENNIFER agent is installed.

3.2.5.3. Setting UDP Size for Operating System

This method is used to set the UDP send buffer size for each operating system. To change this option, you need system root privilege.

If you use IBM AIX, change the buffer UDP buffer size as follows.

```
Checking the setting value : # no -a | grep udp_sendspace
Changing the setting value : # no -o udp_sendspace=65535
Changing the permanent setting value : # no -p -o udp_sendspace
```

If you use Sun Solaris, change the UDP buffer size as follows.

```
Checking the setting value : # ndd -get /dev/udp udp_xmit_hiwat
Changing the setting value : # ndd -set /dev/udp udp_xmit_hiwat 65535
Changing the permanent setting value : You should add 'ndd -set /dev/
udp udp_xmit_hiwat 65535'in the RC script below '/etc/system' or '/
etc'.
```

If you use HP-UX, change the UDP buffer size as follows.

```
Checking the setting value : # sysconfig –q inet udp_sendspace
Changing the setting value : # sysconfig –i inet udp_sendspace = 65535
```

3.2.5.4. Network Tests

Sometimes, the restrictions imposed by the network equipment, such as a firewall, can hinder network communication. To resolve this problem, check whether or not the network is normal.

For a TCP connection, use a simple telnet program to determine whether the port is open.

```
telnet 127.0.0.1 7750
```

For a UDP connection, verify that the JENNIFER server is receiving the data correctly. The JENNIFER_HOME/agent/udptest.sh utility is provided for this purpose. As this utility is executed on the hardware in which the JENNIFER agent is installed, arbitrary data will be sent to the JENNIFER server to conduct UDP connection tests. Before running this utility, you must register the Java application in the path.

```
udptest.sh [jennifer_server_ip] [port] [datalen]
ex) udptest.sh 127.0.0.1 6901 100
```

If the following message is displayed in the JENNIFER server console, the UDP connection is normal.

```
RECV(6901) from=127.0.0.1 data=100 bytes
```

3.2.6. Installing JENNIFER on the Multiple Process

You have to set the JENNIFER agent per Java process because the -Djennifer.config option of each Java process need to be set as a different option.

However, the certain WAS or Java demon may execute the multiple Java processes by using setting value for execution commnad. In this case, you can not set the -Djennifer.config option differently for each different Java process.

Under this multiple Java process envirionment, you have to set -Dconfig.auto=true to the commonly used Java running option for installing the JENNIFER agent.

```
-Dconfig.auto=true
```

The JENNIFER agent check the TCP port while executing Java process and find out the proper JENNIFER agent configuration file by reading the file name set in the -Djennifer.config option.

```
-Dconfig.auto=true -Djennifer.config=x01.conf
```

If you set the option as above, the JENNIFER agent increases the last two numbers of the file name from x01.conf and then, adjusts it into the JENNIFER agent configuration file. The following is a guide for configuring the JENNIFER under multiple Java process.

- Create the multiple configuration files according to the number of Java process and designate it into the directory that has the JENNIFER agent configuration file(x01.conf) with -Djennifer.config option.
- The default configuration file(x01.conf) must have 2 numbers.
- Set the file that has earliest number on the -Djennifer.config option.
- When you give the JENNIFER agent configuration file name, the number in the file name should be serial number(for example, x01.conf, x02.conf ... x50.conf, x51.conf), after same name.
- The option value of the agent_tcp_port and agent_name option in each JENNIFER agent configuraton file should be different.

3.2.7. Changing Temp Directory for the JENNIFER Agent

The loaded class or performance data is temporarily stored in a directory associated with the OS in which the JENNIFER agent is installed. This directory is the WORKING_DIRECTORY that executed the Java application which installed the

JENNIFER agent. This directory has sub directories called .data &.class. The directory path can be changed through the JENNIFER agent configuration.

```
agent_fileroot=/jennifer/agent/data
```

Warning: The directory must exist and be accessible. The Java (WAS) process runtime can be modified, but some portions of data may be temporarily damaged (when modified). Making changes to the configuration before running the Java (WAS) process is recommended.

3.2.8. Exceptional Cases that can Occur during JENNIFER Agent Installation

• Broken SQL parameters on the X-View graph

If the file.encoding between the JENNIFER agent and JENNIFER server is different, you may not see the performance data collected by the JENNIFER server. In this case, set the file.encoding value of the JENNIFER server into the server_encoding option of the JENNIFER agent.

```
server_encoding = UTF8
```

• Native memory leakage in Sun Java 1.5, 6.0, etc.

If native memory leakage occurs in Sun Java 1.5 or 6.0, set the following Java option:

```
-XX:CompileCommand=exclude,org/apache/jennifer/bcel/classfile/
Attribute,readAttribute
```

• Lost application cookie information after JENNIFER installation

The JENNIFER agent uses cookies to determine the number of visitors and concurrent users. However, both the space allotted to cookies and the number of cookies that can be used simultaneously are limited, so if other Java applications have already used too many cookies, the cookie information could be lost. In this case, set the hotfix_remote_address_for_wmonid option of the JENNIFER agent to ‘true’.

```
hotfix_remote_address_for_wmonid = true
```

However, once this option is set to‘true’, you can no longer collect the number of visitors and simultaneous users.

Notice: If you set the hotfix_remote_address_for_wmonid option to ‘true’, you will see the number of visitors and concurrent users only within intranet system.

• Excessive occurrence of WARNING_JDBC_CONN_ILLEGAL_ACCESS

JENNIFER monitors the JDBC in thejava.sql.DriverManager and javax.sql.DataSource class. If the Java application uses another arbitrary connection pool ahead of these classes, it can cause excessive occurrences of WARNING_JDBC_CONN_ILLEGAL_ACCESS exceptions. To resolve this problem, please refer to [Connection Sharing Problems Caused by Multi-Threads].

• Inaccurate WARNING_JDBC_UN_COMMIT_ROLLBACK

If an inaccurate WARNING_JDBC_UN_COMMIT_ROLLBACK exception occurs, set the ignore_rollback_uncommited_error option of the JENNIFER agent to ‘true’.

```
ignore_rollback_uncommited_error = true
```

• Class Cast Exception in Oracle JDBC

If a java.lang.ClassCastException exception related to java.sql.Connection or java.sql.ResultSet occurs while you use Oracle JDBC, you should set the enable_jdbc_oracle_dependency_used option of the JENNIFER agent to ‘true’.

```
enable_jdbc_oracle_dependency_used = true
```

For the more detailed data, please refer to [Oracle Dependency].

• In the event that the HTTP session class can not be initialized after session dump

If the HTTP session class is not initialized after the session dump for HTTP session monitoring is set, remove the session_class option of the JENNIFER agent.

• In the event that CPU usage is high and too many network communications-related items are in the Java core dump

If the CPU usage is too high and there are too many network communications-related items in the Java core dump, set the socket_simple_trace option of the JENNIFER agent to ‘ true’.

```
socket_simple_trace = true
```

• In the event that the active service list is in the waiting status in IBM JDK 1.3

If IBM JDK 1.3 is installed, the searching task for the active service list in the **[Real-time Monitoring | Application]** menu can be delayed. That is because too many Java threads in IBM JDK 1.3 can trigger a problem while gathering Java thread data that processes avtive service.

Set the hotfix_disable_thread_active_count option of JENNIFER agent to ‘ true’. But, if you set this option, you can not see the Java thread data.

```
hotfix_disable_thread_active_count = true
```

• In the event that error ocurs when the getCallerClassLoader method of DriverManager class is called in JRockit 1.5.0_06.

The error can happen when the getCallerClassLoader of DriverManager class is called in JRockit 1.5.0_06. JENNIFER provides a patch file to help you solve this error. When you install the JENNIFER agent, set -Xbootclasspath as follows :

```
-Xbootclasspath/p:/jennifer/agent/lwst.jrockit150_06.jar:/jennifer/
agent/lwst.boot.jar:/jennifer/agent/lwst.jdk.jar
```

The lwst.jrockit150_06.jar file should be prior to the lwst.jdk.jar file.

3.3. Setting JENNIFER Agent Names

The JENNIFER agent ID is limited to three letters, such as ‘W11’. The ambiguity of such a name makes it very difficult for the JENNIFER client to clearly understand the meaning of the tasks. To resolve this problem, assign an arbitrary name to the JENNIFER agent, and allow the JENNIFER client to use this name.

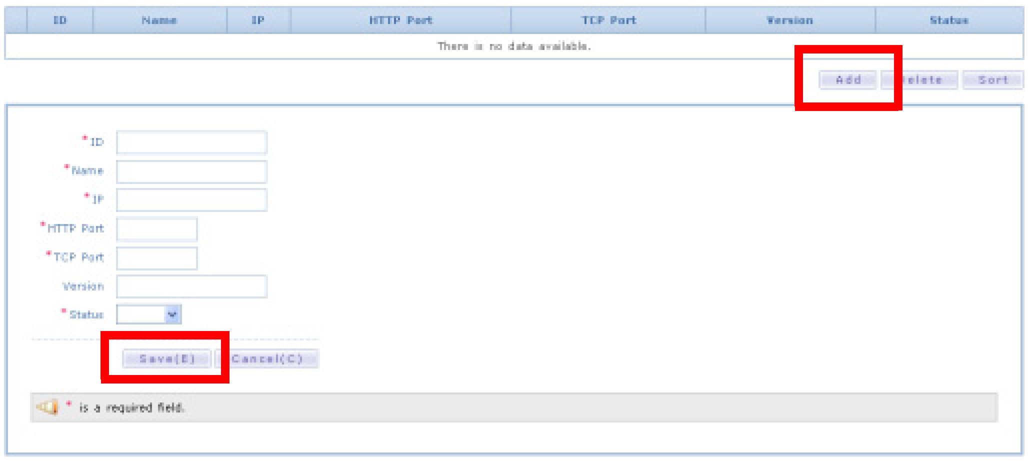
The following method is used to assign a name to the JENNIFER agent.

First, select the **[Properties | Configuration | Domain Setting]** menu, and enter the domain information.

Notice: The domain is a JENNIFER server. For this reason, you must enter the JENNIFER server-related information in the domain information.

When you click the [Add] button at the bottom of the domain list, the domain input form will appear. Enter your data in the form, and click the [Save] button at the bottom.

Figure 3-2:Domain Input Screen



There is no data available.

* is a required field.

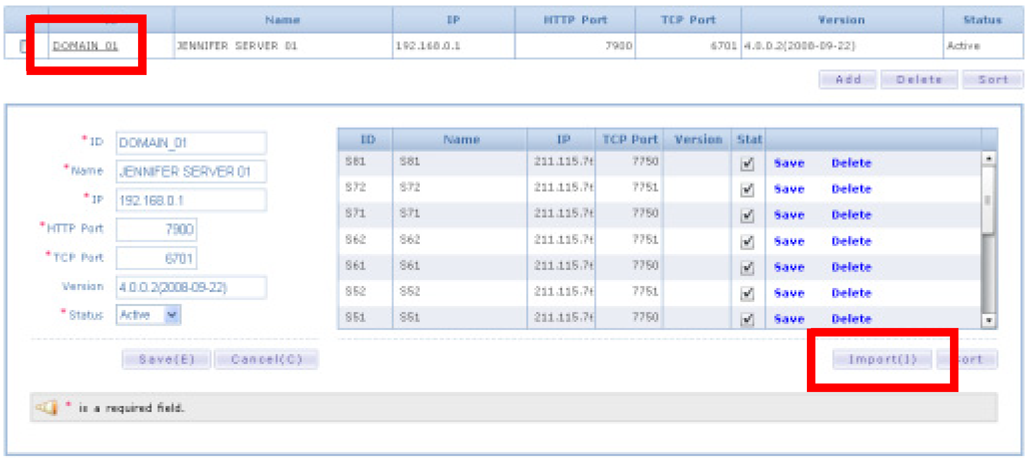
The descriptions of the fields in the domain input form are as follows:

Table3-1: Domain Input Form

Field	Descriptions
ID	The unique ID for the JENNIFER server. Only alphanumeric characters can be used, and it must contain no spaces.
Name	The JENNIFER server's name
IP	The JENNIFER server's IP address
HTTP port	The HTTP port number for the JENNIFER server
TCP port	The JENNIFER server's port number, as set by the server_tcp_port option
Version	Enter 4.0 for the JENNIFER server version. Versions lower than 4.0 cannot be integrated in the domain.
Status	Any JENNIFER server set to Inactive is not used.

The domain list will show the newly added domain information. As you click the domain information ID, the domain information will appear at the bottom. If you click the [Import] button in the right lower corner, the JENNIFER agent list will appear.

Figure 3-3:Importing the JENNIFER Agent



	ID	Name	IP	HTTP Port	TCP Port	Version	Status
☐	DOMAIN_01	JENNIFER SERVER 01	192.168.0.1	7900	6701	4.0.0.2(2008-09-22)	Active

Add Delete Sort

*ID	DOMAIN_01
*Name	JENNIFER SERVER 01
*IP	192.168.0.1
*HTTP Port	7900
*TCP Port	6701
Version	4.0.0.2(2008-09-22)
*Status	Active

Save(E) Cancel(C)

ID	Name	IP	TCP Port	Version	Stat		
S81	S81	211.115.7	7750		✓	Save	Delete
S72	S72	211.115.7	7751		✓	Save	Delete
S71	S71	211.115.7	7750		✓	Save	Delete
S62	S62	211.115.7	7751		✓	Save	Delete
S61	S61	211.115.7	7750		✓	Save	Delete
S52	S52	211.115.7	7751		✓	Save	Delete
S51	S51	211.115.7	7750		✓	Save	Delete

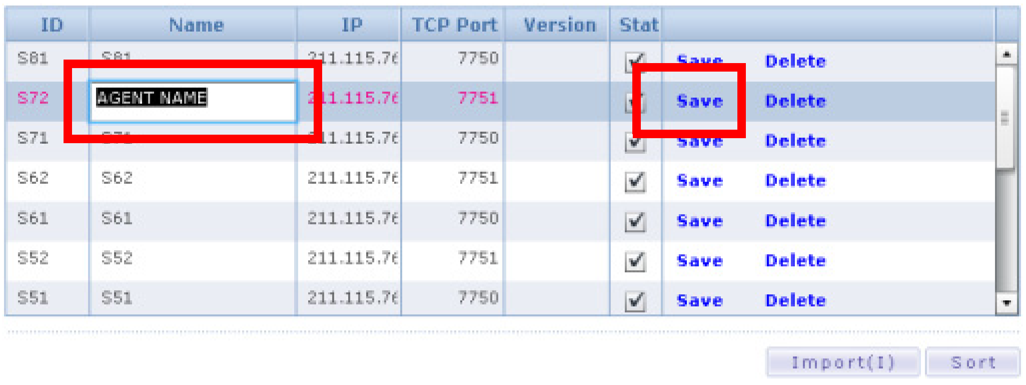
Import(I) Sort

* is a required field.

Notice: If you click the [Import] button, the currently operating JENNIFER agents will appear. After you add a JENNIFER agent, click the [Import] button again, and the newly added JENNIFER agent will appear in the list

By default, the JENNIFER agent name is identical to the JENNIFER agent ID. After changing the name in the JENNIFER agent name column, click the [Save] button. The JENNIFER agent name will then be changed.

Figure 3-4:JENNIFER Agent List



ID	Name	IP	TCP Port	Version	Stat		
S81	S81	211.115.7	7750		✓	Save	Delete
S72	AGENT NAME	211.115.7	7751		✓	Save	Delete
S71	S71	211.115.7	7750		✓	Save	Delete
S62	S62	211.115.7	7751		✓	Save	Delete
S61	S61	211.115.7	7750		✓	Save	Delete
S52	S52	211.115.7	7751		✓	Save	Delete
S51	S51	211.115.7	7750		✓	Save	Delete

Import(I) Sort

Notice: You must login again if you want to see the changed JENNIFER agent name in the JENNIFER client.

3.4. Byte Code Instrumentation

In order to monitor your application effectively, you should inject tracking code for performance data collection and profile information extraction code in the classes that constitute the Java application without modifying the source code of the monitored Java application. This is called Byte Code Instrumentation (hereinafter, BCI). In JENNIFER, the module responsible for BCI is called the Instrumentation or LWST module.

3.4.1. LWST Build and Installation

The BCI tasks performed by the LWST modules are divided into runtime processing, which is performed by thelwst.boot.jar and lwst.javaagent.jar files; and processing, which is performed by the lwst.jdk.jar file, which packs the main Java classes such as java.lang.ClassLoader and java.net.Socket offline.

First, the lwst.boot.jar file is the LWST core module, and it is installed in bootclasspath. The lwst.javaagent.jar file used for Java 1.5 or higher is registered as javaagent.

LWST build refers to patching the main Java classes, such as java.lang.ClassLoader and java.net.Socket, and packing them into the lwst.jdk.jar file offline. LWST build generates the AVA_HOME/jre/lib/rt.jar file or the lwst.jdk.jar file based on the JVM module that plays the same role as the JAVA_HOME/jre/lib/rt.jar file through the lwst40.sh utility. The lwst.jdk.jar file created by LWST build is installed in the bootclasspath directory.

Notice: In Java 1.4.x or lower, LWST build patches the java.lang.ClassLoader class and performs BCI. However, in Java 1.5 or higher, javaagent is used to perform BCI. Therefore, Java 1.4 or lower does not use the lwst.javaagent.jar file.

The LWST build method is as follows. This utility is located in the JENNIFER_HOME/agent directory.

```
lwst40.sh [command option] [rt.jar]
```

The first parameter that is used when running the lwst40.sh utility is called the COMMAND option. The COMMAND option is not case-sensitive.

Table 3-2: LWST Build COMMAND Options

Option	Descriptions
JDK15	This generates the lwst.jdk.jar for Java 1.5 or higher. The java.lang.ClassLoader class is not patched. **Warning:** If this option is not set for Java 1.5 or higher, duplicate instrumentation will occur.
AS400	This option is for the AS 400 Websphere system. No class is patched from the rt.jar file, and only the SQL tracking classes are used to make the lwst.jdk.jar file.
JROCKIT	This is the option for BEA JROCKIT Java 1.4.x or lower.

Table 3-2: LWST Build COMMAND Options

Option	Descriptions
ZOS13	This is the option for Java 1.3.x running on the mainframe OS.
SAFE	If an error is generated while starting the Java application after the JENNIFER agent has been installed, this option should be used. LWST build has a different location to change the byte code for the java.lang.ClassLoader class.

The second parameters are used to set the location of the Java library file containing the main Java classes, such as java.lang.ClassLoader and java.net.Socket. If the Sun JVM is used, it usually refers to the JAVA_HOME/lib/rt.jar file.

Using LWST build, you can decide to monitor the JDBC, the Java threads or the Class loader. This is called the LWST build patch option. This option should be set by directly modifying the lwst40.sh file.

The following is related to the build patch option in the lwst40.sh file.

```
OPT="$OPT -Dbuild_jdbc=true"
OPT="$OPT -Dbuild_classloader=true"
OPT="$OPT -Dbuild_thread=false"

OPT="$OPT -Dbuild_collection=false"
#OPT="$OPT -Dtype_collection=TRU64"
#OPT="$OPT -Dbuild_collection_map=false"
#OPT="$OPT -Dbuild_collection_list=false"

OPT="$OPT -Dbuild_file=true"
OPT="$OPT -Dbuild_socket=true"
OPT="$OPT -Dbuild_xml=true"
```

The table below describes each option.

Table 3-3: LWST Build Patch Options

Option	Description
build_jdbc	Determines whether to monitor the JDBC and the SQL. The default configuration is false.
build_classloader	Specifies whether to patch the java.lang.ClassLoader class. If this option is set to false, then most of the LWST functions will not operate in Java 1.4x or lower.
build_thread	Specifies whether to monitor the java.lang.Thread objects. If a Java thread-related error occurs, this option should be set to false.

Table 3-3: LWST Build Patch Options

Option	Description
build_collection	Specifies whether to monitor the Java collection class such as java.util.HashMap and java.util.ArrayList. As this monitoring can degrade performance, this option must only be set to true if it is necessary to determine the cause of Java heap memory leakage.
build_collection_map	When the build_collection option is set to true, this option is used to specify whether to monitor the map type of information structure classes, such as java.util.HashMap and java.util.HashTable.
build_collection_list	When the build_collection option is set to true, this option is used to specify whether to monitor the list type of information structure classes, such as java.util.ArrayList and java.util.Vector.
build_file	Specifies whether to patch the java.io.FileOutputStream and java.io.FileInputStream classes. If performance is degraded while monitoring an application with excessive file IO, this option should be set to false.
build_socket	Specifies whether to patch the java.net.Socket class. If performance is degraded while monitoring the socket, this option should be set to false. Even when this option is set to true, you can still monitor the socket by using the socket_simple_trace option of the JENNIFER agent.
build_xml	Specifies whether to monitor the Java standard XML parser. If this option is set to true, you can track the XML parsing time of the javax.xml.parsers.SAXParser and javax.xml.parsers.DocumentBuilder class. The default configuration is false.

Notice: If you perform LWST build, the LWST build patch option will be displayed on the console. Please study this message carefully.

3.4.2. Cautions in the Application of LWST

You must pay attention to the following items when using the LWST for profiling.

• If you profile a large-sized JSP file, overload can occur.

While in operation, the JSP file is converted to a class file by the WAS. This class can be profiled using the LWST. However, if you profile a JSP file that is larger than about 2MB, CPU or memory usage could increase.

To prevent such a problem, the JENNIFER agent offers the following option. Any class exceeding the size set using the hook_class_max_size option of the JENNIFER agent will not be processed by the BCI. The default configuration is 1048576 (1mb), and it is expressed in bytes.

```
hook_class_max_size = 1048576
```

If a class exceeding the size set by this option is profiled, then the LWST log file will display the message, 'too big class[class name] len nnn bytes'. If you are not concerned about BCI overloading, you can increase this value.

• If you use a collection class like java.util.ArrayList as Data Transfer Objects, collection monitoring will lead to overloading.

According to some reports, if the java.util.HashMap or java.util.ArrayList class is used as a Data Transfer Object, collection monitoring will frequently lead to overloading due to Java garbage collection.

• Unless you use JDK15 as a LWST build command option for Java 1.5 or higher, duplicate instrumentation will occur.

By default, JENNIFER patches the class loader to perform BCI. However, you must use a javaagent function for Java 1.5 or higher.

Therefore, when you perform LWST build for Java 1.5 or higher, you must use JDK15 as the command option and avoid patching the java.lang.ClassLoader class. If you neglect to do this and assign the lwst.javaagent.jar file to the javaagent option, the LWST log file will display an alarm message saying that duplicated instrumentation is being attempted.

```
[WARNING] Duplicated instrumentation!
```

No bug will be generated by duplicated instrumentation, but there will be some functional limitations in the Java 1.5 environment. In this case, you must use the JDK15 option to perform LWST build again.

It is necessary to check the log file to determine how the instrumentation module is initialized.

- Instrumentation module initialization using the lwst.javaagent.jar file

```
[INF] -javaagent used!!
```

- Instrumentation module initialization using the lwst.jdk.jar file.

```
[INF] The instrumentation mode is for java14/java13!!
```

3.5. JENNIFER Agent Utilities

This section describes some useful utilities for JENNIFER agent installation.

3.5.1. Class Finder

Class finder is a utility that searches for class information in the class path. This utility resides in the JENNIFER_HOME/agent directory..

```
finder.sh [mode] [classpath] [target]
```

The descriptions of the options are as follows:

- Using the [mode] option, you can perform various functions such as CALL, CLASS, SUPER, and INTERFACE.
- Using the [classpath] option, you can assign multiple jar files, which must be separated by semicolons [;].
- Using the [target] option, you can set the class or method information to search for.

For example, you can search for the class invoking the java.lang.Thread.sleep method in the rt.jar file as follows:

```
finder.sh CALL /usr/java/jre/lib/rt.jar java.lang.Thread.sleep
```

You can search for the class invoking the sleep method in the rt.jar file as follows.

```
finder.sh CALL /usr/java/jre/lib/rt.jar *.sleep
```

You can search for the class using the java.lang.Thread method in the rt.jar file as follows.

```
finder.sh CALL /bin/java/jre/lib/rt.jar java.lang.Thread.*
```

You can search for the java.lang.Thread class in the rt.jar file as follows.

```
finder.sh CLASS /usr/java/jre/lib/rt.jar java.lang.Thread
```

You can search for the class having the getConnection method in the rt.jar file as follows.

```
finder.sh METHOD /usr/java/jre/lib/rt.jar getConnection
```

You can search for the sub-class of java.lang.Thread method in the rt.jar file as follows.

```
finder.sh SUPER  /usr/java/jre/lib/rt.jar java.lang.Thread
```

You can search for the class implementing the java.lang.Runnable interface in the rt.jar file as follows..

```
finder.sh INTERFACE /usr/java/jre/lib/rt.jar java.lang.Runnable
```

3.5.2. JSP Utilities

The JSP files used to investigate JENNIFER agent installation errors reside in the JENNIFER_HOME/agent/tuning directory. You must copy them to the WAS before executing them.

The following is the descriptions about JSP file.

Table 3-4: JSP Utilities

JSP File	Descriptions
env.jsp	Search for the WAS environment variables. This is useful for investigating the normal installation of JENNIFER.
jarcheck.jsp	Examines which JAR file the class is loaded from. You can utilize it when applications are distributed incorrectly and the class path is abnormal.

4 .NET Agent Management

4.1. JENNIFER .NET Agent Configuration File

All JENNIFER agent configurations, including monitoring standards and methods, and network port number assignments, are set by the JENNIFER agent options. The file which is designated as a file name in the appSettings/add[@key='Jennifer.FileName'] value property of the web.config is a JENNIFER agent configuration file.

```
<appSettings>
  <add key="Jennifer.FileName" value="default_pool.conf" />
</appSettings>
```

4.1.1. Format of the Configuration File

The JENNIFER agent option for the Java version is saved in a text formatted configuration file. The file has the following characteristics:

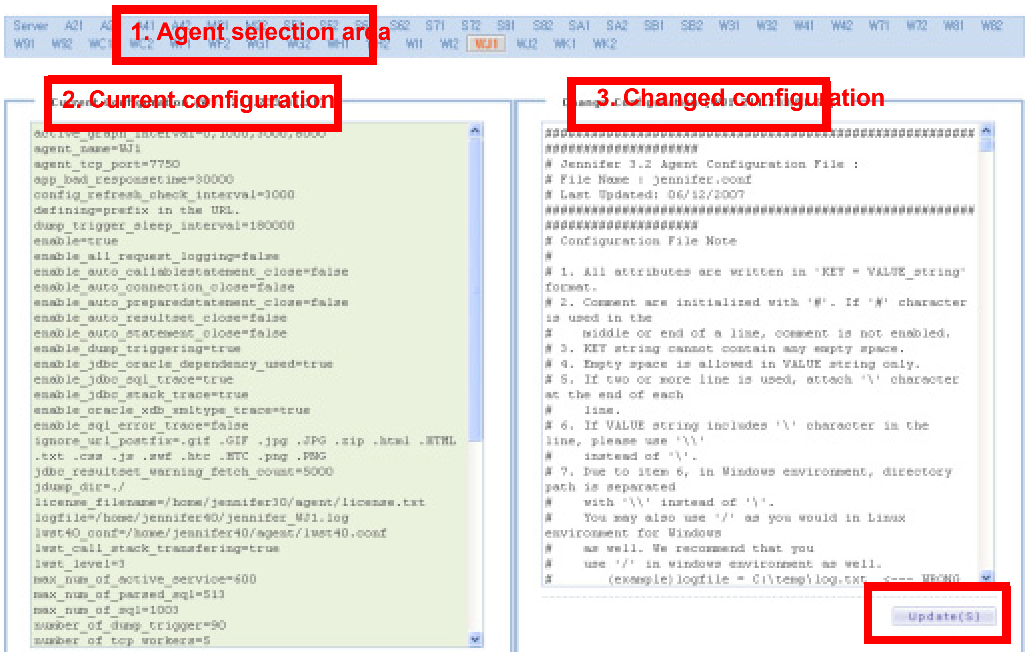
- The format is ‘key = value’. The key represents the JENNIFER agent option.
- Any lines starting with [#] are comments. However, the appearance of a [#] symbol in the middle of a text string will not indicate a comment.

- Instead of [=], you can use ‘SPACE’ to separate the key and the value.
- For the above reason, it is prohibited to use ‘SPACE’ as the key.
- However, you can include ‘SPACE’ in the value.
- If you need to use more than one line, add [\] at the end of the line. If you need to use [\] for text, then use [\\] to finish the line.
- For the reason shown above, you must use [\\], not [\] to distinguish the directory in the Microsoft windows environment. However, you can use [/] in Windows, as in the Unix environment, and you are recommended to do so.
- With a few exceptions, options change is reflected in the system as soon as the change is made. In addition, the option that can be used when you restart Java application installed the JENNIFER agent, is also described.

4.1.2. Changing the Configuration

The configuration file can be directly modified using a text editor, or using the **[Properties | Configuration]** menu in the JENNIFER client. However, only the users in Administrator group are allowed to modify the configuration options through the JENNIFER client.

Figure 4-1:JENNIFER Option Setting Screen



```
active_graph_interval=0,1000,3000,8000
agent_name=WJ1
agent_tcp_port=7750
app_bad_responsetime=30000
config_refresh_check_interval=3000
defining=prefix in the URL.
dump_trigger_sleep_interval=180000
enable=true
enable_all_request_logging=false
enable_auto_callablestatement_close=false
enable_auto_connection_close=false
enable_auto_preparedstatement_close=false
enable_auto_resultset_close=false
enable_auto_statement_close=false
enable_dump_triggering=true
enable_jdbc_oracle_dependency_used=true
enable_jdbc_sql_trace=true
enable_jdbc_stack_trace=true
enable_oracle_xdb_xmltype_trace=true
enable_sql_error_trace=false
ignore_url_postfix=.gif .GIF .jpg .JPG .zip .html .HTML
.txt .css .js .swf .htc .HTC .png .PNG
jdbc_resultset_warning_fetch_count=5000
jdump_dir=./
license_filename=/home/jennifer30/agent/license.txt
logfile=/home/jennifer40/jennifer_WJ1.log
lwst40_conf=/home/jennifer40/agent/lwst40.conf
lwst_call_stack_transfering=true
lwst_level=3
max_num_of_active_service=400
max_num_of_parsed_sql=513
max_num_of_sql=1003
number_of_dump_trigger=90
number of tcp workers=5
```

```
##################################################################
####################
# Jennifer 3.2 Agent Configuration File :
# File Name : jennifer.conf
# Last Updated: 06/12/2007
##################################################################
####################
# Configuration File Note
#
# 1. All attributes are written in 'KEY = VALUE_string'
format.
# 2. Comment are initialized with '#'. If '#' character
is used in the
#     middle or end of a line, comment is not enabled.
# 3. KEY string cannot contain any empty space.
# 4. Empty space is allowed in VALUE string only.
# 5. If two or more line is used, attach '\' character
at the end of each
#     line.
# 6. If VALUE string includes '\' character in the
line, please use '\\'
#     instead of '\'.
# 7. Due to item 6, in Windows environment, directory
path is separated
#     with '\\' instead of '\'.
#     You may also use '/' as you would in Linux
environment for Windows
#     as well. We recommend that you
#     use '/' in windows environment as well.
#        (example)logfile = C:\temp\log.txt  <--- WRONG
```

1. Agent selection area - To check or change the JENNIFER agent option, select the JENNIFER agent here.

2. Current configuration - In the current configuration on the left, review the JENNIFER agent option.
3. Changed configuration - Modify the existing option, or add a new one in the configuration change input form at the right. Click the [change] button at the bottom to apply the change.

4.1.3. Making a New Configuration File

If you want to make a new JENNIFER agent configuration file, copy the existing configuration file (ex. app_pool.conf) in the JENNIFER_HOME/agent.net/conf directory, and change the following option values:

```
agent_pool =[agent1 id]:[agent1 port], [agent2 id]:[agent2 port] ……
```

If you want to monitor more than one Java application on the same hardware, you must assign a new JENNIFER agent ID and use a different agent TCP listening port number.

If you want to monitor multiple applications on the same hardware, you should assign a new JENNIFER agent ID and use a different agent TCP listening port number. For instance, when you make the each “app_A_pool.conf” file for the “A” web site and “app_B_pool.conf” file for the “B” web site, you should assign a each different ID ad port number in the conf file as following.

[app_A_pool.conf File] :

```
agent_pool = N10:9000
```

[app_B_pool.conf File]

```
agent_pool = T10:9100
```

4.2. Default Configuration and Management of the JENNIFER Java Agent

This section describes the exceptions that can occur during the JENNIFER agent installation and the options that need to changed after JENNIFER agent is installed.

4.2.1. JENNIFER Agent ID Pool

A single JENNIFER server can collect performance data from multiple JENNIFER agents. For this reason, each JENNIFER agent should have a unique ID, so that the JENNIFER server can distinguish the performance data from different agents. This can be set using the agent_pool option of the JENNIFER agent.

```
agent_pool=N10:9000
```

Warning: The JENNIFER agent ID should be composed of alphanumeric characters only, and must be three characters in length.

If one application instance(EXE) is executed, set one ID like an above example - "[Unique ID]:[Port Number]". However, it is possible to set the value of "Web Garden" of "AppPool" property to above 2, when the w3wp.exe is multi-processed based on the value of "Web Garden".

Therefore, you have to set the multiple IDs according to the number of Web Garden. The following is an example about to set an ID when the setting value of "Web Garden" is 2.

```
agent_pool=N10:9000,N11:9001
```

Notice: Unlike the Java version, the change in the JENNIFER .NET agent ID is not reflected in running time. Therefore, you should recycle the w3wp.exe after changing the ID.

4.2.2. Log File

The JENNIFER agent log file can be set using the logfile option of the JENNIFER agent.
.

```
logfile = jennifer_{0}.log
```

The "{0}" format is configured by default depending on the characteristics of the application as follows:

```
Web Application: Jennifer_[App Pool name]_[Agent Unique ID]_[Today's
date].log
Application: Jennifer_[Agent Unique ID]_[Today's date].log
```

If today is March 16th, 2010, the log is recorded in JENNIFER_HOME/agent.net/log/jennifer_DefaultAppPool_N10_20100316.log file by default. This log file contains exceptional cases, service dumps and debugging contents that occur in the application.

If you want not to record daily log files, set the enable_logfile_daily_rotation option of the JENNIFER agent to ‘false’ .

```
enable_logfile_daily_rotation = false
```

4.2.3. License Key File

The JENNIFER agent license key is stored in a text file set by the license_filename option of the JENNIFER agent.

```
license_filename = license.txt
```

By default, the license key is saved in the JENNIFER_HOME/agent.net/license.txt.

4.2.4. License Key and JENNIFER Agent Enablement

If you adjust invalid JENNIFR license key due to the wrong IP number or expired one, or set the “enable” option of JENNIFER agent to ‘false’ , the JENNIFER agent runs as follows:

All functions of the JENNIFER agent may stop, even though you restart the application. In other words, JENNIFER does not collect and send the performance data any more.(But, JENNIFER collects the CPU, DB connections and Memory data which is not related to application request/response profiling function. Of course, this does not give any impact on application operation.

However, the profiling related code is activated internally. For this reason, user should unregist the JENNIFER program to inactivate the profiling codes and restart the application. It is because the .NET profiling executes in changing the IL(Intermediate Language) when the module is loading. The already loaded module can not be executed to the past code due to the changed IL code.

4.2.5. Network Configuration

The JENNIFER agent sends the collected performance data to the JENNIFER server. In this section, we will describe the network configuration used for this task.

4.2.5.1. TCP Network Configuration

The JENNIFER server makes the reverse connection to the TCP port of the JENNIFER agent. JENNIFER usually uses the TCP reverse connection when it searches the large-scale of performance data such as active service lists, statistical data of application processing status for 10 minutes, loaded class lists, and socket/file lists that are not stored in the DB. Therefore, set the agent ID and port number in the agent_pool of JENNIFER agent.(The default port number in the app_pool.conf is 7800.)

```
agent_pool=N10:7800
```

Agent TCP Worker is the thread of JENNIFER agent which processes such JENNIFER server' s request. Using number_of_tcp_workers, set the number of Agent TCP Worker. The default value is 5.

```
number_of_tcp_workers = 5
```

If you want to change these two options about TCP network configuration, restart the application where the JENNIFER agent is inatalled.

4.2.5.2. UDP Network Configuration

The JENNIFER agent sends performance data to the UDP port of the JENNIFER server. First, using the udp_server_host option of the JENNIFER server, set the IP address for the JENNIFER server.

```
udp_server_host = 127.0.0.1
```

Use the UDP port number of the JENNIFER server to set a range of options for the JENNIFER agent.

```
server_udp_runtime_port = 6901
server_udp_listen_port = 6902
server_udp_lwst_call_stack_port = 6703
```

If these two options are modified, the changes will be applied without restarting the Java application in which the JENNIFER agent is installed.

4.2.5.3. Network Tests

Sometimes, the restrictions imposed by the network equipment, such as a firewall, can hinder network communication. To resolve this problem, check whether or not the network is normal.

For a TCP connection, use a simple telnet program to determine whether the port is open.

```
telnet 127.0.0.1 7750
```

For a UDP connection, verify that the JENNIFER server is receiving the data correctly. The JenniferSoft provides the JENNIFER_HOME/agent.net/Utility/udptest.exe utility program for this purpose. As this utility is executed on the hardware in which the JENNIFER agent is installed, arbitrary data will be sent to the JENNIFER server to conduct UDP connection tests.

```
udptest.exe [jennifer_server_ip] [port] [datalen]
ex) udptest.exe 127.0.0.1 6901 100
```

If the following message is displayed in the JENNIFER server console, the UDP connection is normal.

```
RECV(6901) from=127.0.0.1 data=100 bytes
```

4.2.6. Exceptional Cases that can Occur during JENNIFER Agent Installation

• Lost application cookie information after JENNIFER installation

The JENNIFER agent uses cookies to determine the number of visitors and concurrent users. However, both the space allotted to cookies and the number of cookies that can be used simultaneously are limited, so if other Java applications have already used too many cookies, the cookie information could be lost. In this case, set the hotfix_remote_address_for_wmonid option of the JENNIFER agent to 'true'.

```
hotfix_remote_address_for_wmonid = true
```

However, once this option is set to'true', you can no longer collect the number of visitors and simultaneous users.

Notice: If you set the hotfix_remote_address_for_wmonid option to 'true', you will see the number of visitors and concurrent users only within intranet system.

4.3. Setting JENNIFER Agent Names

The JENNIFER agent ID is limited to three letters, such as 'W11'. The ambiguity of such a name makes it very difficult for the JENNIFER client to clearly understand the meaning of the tasks. To resolve this problem, assign an arbitrary name to the JENNIFER agent, and allow the JENNIFER client to use this name.

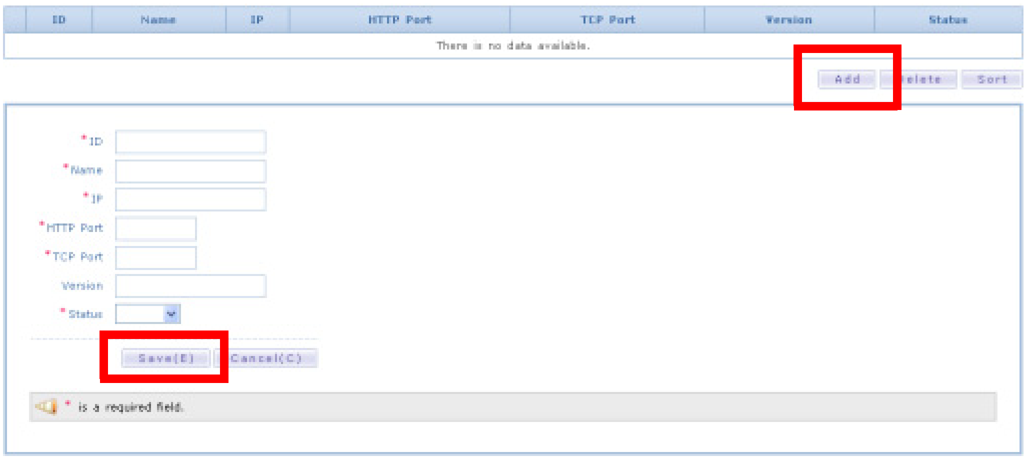
The following method is used to assign a name to the JENNIFER agent.

First, select the **[Properties | Configuration | Domain Setting]** menu, and enter the domain information.

Notice: The domain is a JENNIFER server. For this reason, you must enter the JENNIFER server-related information in the domain information.

When you click the [Add] button at the bottom of the domain list, the domain input form will appear. Enter your data in the form, and click the [Save] button at the bottom.

Figure 4-2:Domain Input Screen



ID	Name	IP	HTTP Port	TCP Port	Version	Status
There is no data available.						

*ID	
*Name	
*IP	
*HTTP Port	
*TCP Port	
Version	
*Status	

* is a required field.

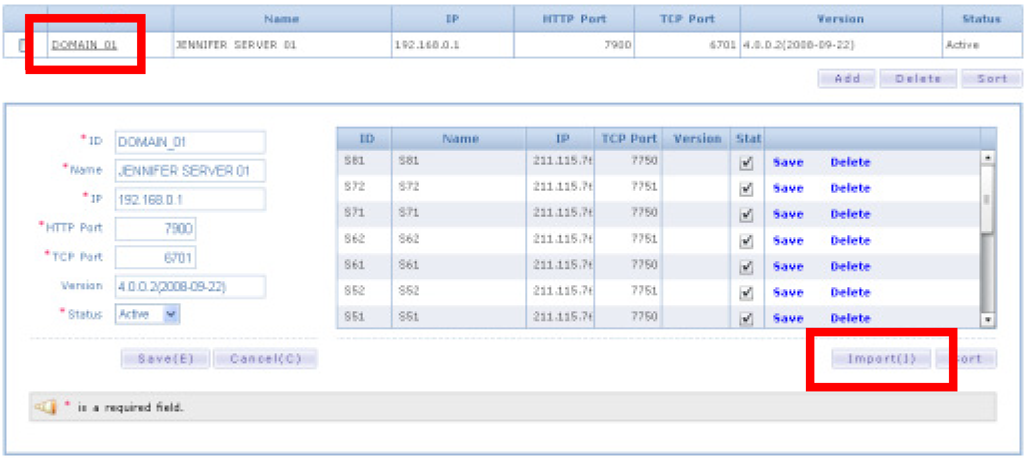
The descriptions of the fields in the domain input form are as follows:

Table4-1: Domain Input Form

Field	Descriptions
ID	The unique ID for the JENNIFER server. Only alphanumeric characters can be used, and it must contain no spaces.
Name	The JENNIFER server's name
IP	The JENNIFER server's IP address
HTTP port	The HTTP port number for the JENNIFER server
TCP port	The JENNIFER server's port number, as set by the server_tcp_port option
Version	Enter 4.0 for the JENNIFER server version. Versions lower than 4.0 cannot be integrated in the domain.
Status	Any JENNIFER server set to Inactive is not used.

The domain list will show the newly added domain information. As you click the domain information ID, the domain information will appear at the bottom. If you click the [Import] button in the right lower corner, the JENNIFER agent list will appear.

Figure 4-3:Importing the JENNIFER Agent



		Name	IP	HTTP Port	TCP Port	Version	Status
	DOMAIN_01	JENNIFER SERVER 01	192.168.0.1	7900	6701	4.0.0.2(2008-09-22)	Active

*ID	DOMAIN_01
*Name	JENNIFER SERVER 01
*IP	192.168.0.1
*HTTP Port	7900
*TCP Port	6701
Version	4.0.0.2(2008-09-22)
*Status	Active

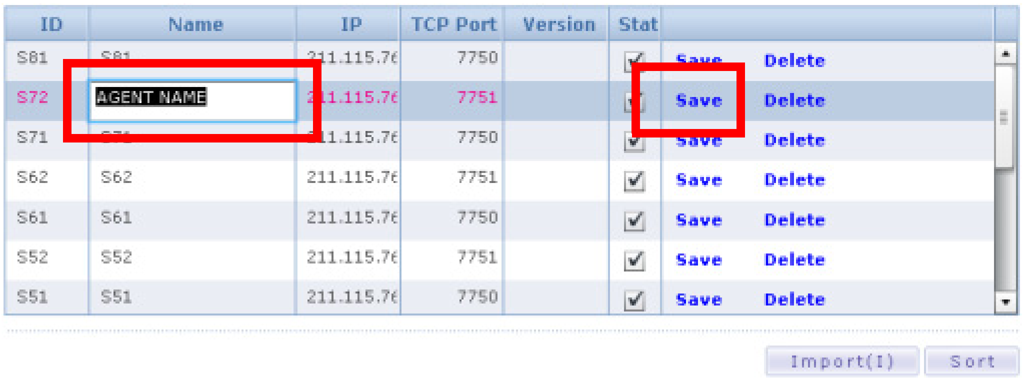
ID	Name	IP	TCP Port	Version	Stat		
S81	S81	211.115.7(	7750		✓	Save	Delete
S72	S72	211.115.7(	7751		✓	Save	Delete
S71	S71	211.115.7(	7750		✓	Save	Delete
S62	S62	211.115.7(	7751		✓	Save	Delete
S61	S61	211.115.7(	7750		✓	Save	Delete
S52	S52	211.115.7(	7751		✓	Save	Delete
S51	S51	211.115.7(	7750		✓	Save	Delete

* is a required field.

Notice: If you click the [Import] button, the currently operating JENNIFER agents will appear. After you add a JENNIFER agent, click the [Import] button again, and the newly added JENNIFER agent will appear in the list

By default, the JENNIFER agent name is identical to the JENNIFER agent ID. After changing the name in the JENNIFER agent name column, click the [Save] button. The JENNIFER agent name will then be changed.

Figure 4-4:JENNIFER Agent List



ID	Name	IP	TCP Port	Version	Stat		
S81	S81	211.115.7(	7750		✓	Save	Delete
S72	AGENT NAME	211.115.7(	7751		✓	Save	Delete
S71	S71	211.115.7(	7750		✓	Save	Delete
S62	S62	211.115.7(	7751		✓	Save	Delete
S61	S61	211.115.7(	7750		✓	Save	Delete
S52	S52	211.115.7(	7751		✓	Save	Delete
S51	S51	211.115.7(	7750		✓	Save	Delete

Notice: You must login again if you want to see the changed JENNIFER agent name in the JENNIFER client.

4.4. JENNIFER Agent Utilities

This section describes some useful utilities for JENNIFER agent installation.

4.4.1. Utilities

The JSP files used to investigate JENNIFER agent installation errors reside in the JENNIFER_HOME/agent/tuning directory. You must copy them to the WAS before executing them.

The following is the descriptions about JSP file.

Table 4-2: JSP Utilities

JSP File	Descriptions
env.jsp	Search for the environment variables of w3wp process running IIS. This is useful for investigating the normal installation of JENNIFER.
comTest.aspx	Investigates that the profilier module of JENNIFER .NET Agent is activated.
ipchecker.exe	Shows the IP list of the currently running computer. Identifies IP which is needed in getting JENNIFER license.
udptest.exe	Identifies that the UDP port is opened to the computer installed the JENNIFER server.

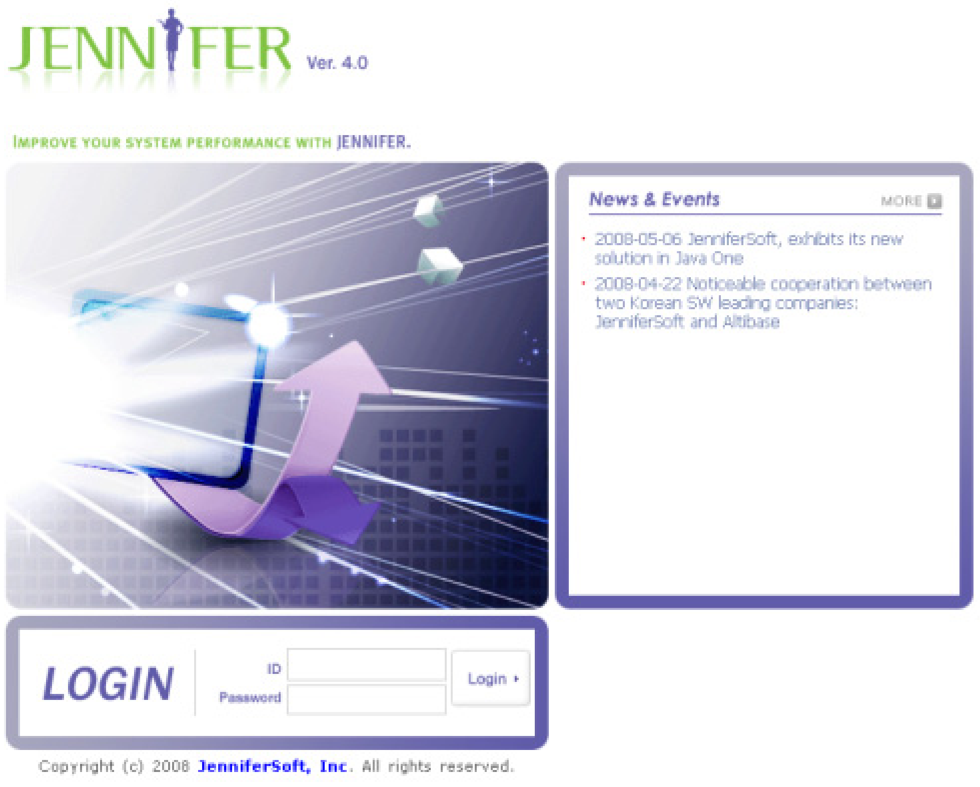
5 User Interface

5.1. Log-in

If you enter the following URL, which consists of the port number and the domain address or the IP address of the server where the JENNIFER server is installed, the log-in screen will appear. All of the JENNIFER functions can be used after you log in. The default port number for the JENNIFER server is 7900.

```
http://jennifer_server_ip:7900
```

Figure 5-1:Log-in Screen



- 2008-05-06 JenniferSoft, exhibits its new solution in Java One
- 2008-04-22 Noticeable cooperation between two Korean SW leading companies: JenniferSoft and Altibase

Copyright (c) 2008 JenniferSoft, Inc. All rights reserved.

If you want to log in, you need to enter your password and click the **[Log-in]** button. The administrator's ID is "admin," and its password is also "admin." After you log in for the first time, you are strongly recommended to change the password.

At the right of the log in screen, the [News & Events] menu can be seen. If you want to hide it, you should set the ui_hide_news option of the JENNIFER server to true.

```
ui_hide_news = true
```

If you log in successfully, then you will move to the **[Dashboard | JENNIFER Dashboard]** menu. The first screen that appears after you log in can be set according to the user and his/her privileges.

5.2. Client Setting

The JENNIFER client is web-based, and therefore the user accesses the JENNIFER user interface through a web browser. Since the charts are implemented via Java applets, the web browser that is used must have the Java plug-in. In addition, since the grid is implemented in flash files, a flash player is required.

5.2.1. Operating System

The JENNIFER client supports Microsoft windows XP and Vista. If you choose Windows classic theme on the windows XP, JENNIFER Black skin staly does not displayed properly. You can check the current theme on the [Properties] menu of the context menu by clicking the right mouse button.

Notice: Officially, the JENNIFER client does not support Linux or Mac OS. However, if your operating system can allow the installation of Firefox 3.0 or higher, with the Java plug-in 1.6.0_10 or higher, you should be able to use the JENNIFER client.

5.2.2. Web Browser

Various web browsers are supported, including Microsoft IE 7.0 and 8.0, and Mozilla firefox 3.0.

Warning: IE 6.0 will not be supported. IE 6.0 does not fully meet the web standard inclusing CSS and not support PNG image file clearly. Therefore, JENNIFER does not support IE 6.0. We recommend you to use Firefox 3.0 unless you upgrade it into IE 7.0 or 8.0.

If you want to user JENNIFER, you should set the web browser in such that it can use cookies and Java scripts. Most of web browsers are basically set to use these functions.

5.2.3. Java Plug-in

Since some aspects of the JENNIFER client are implemented via Java applets, you must install the Sun Java plug-in 6.0 or higher.

Warning: JENNIFER 4.0 or higher does not support Microsoft Java VM.

5.2.3.1. Installing and Uninstalling

To install or uninstall the Java plug-in, you should refer to the Sun Microsystems Java site at (http://java.sun.com/javase/downloads/index.jsp).

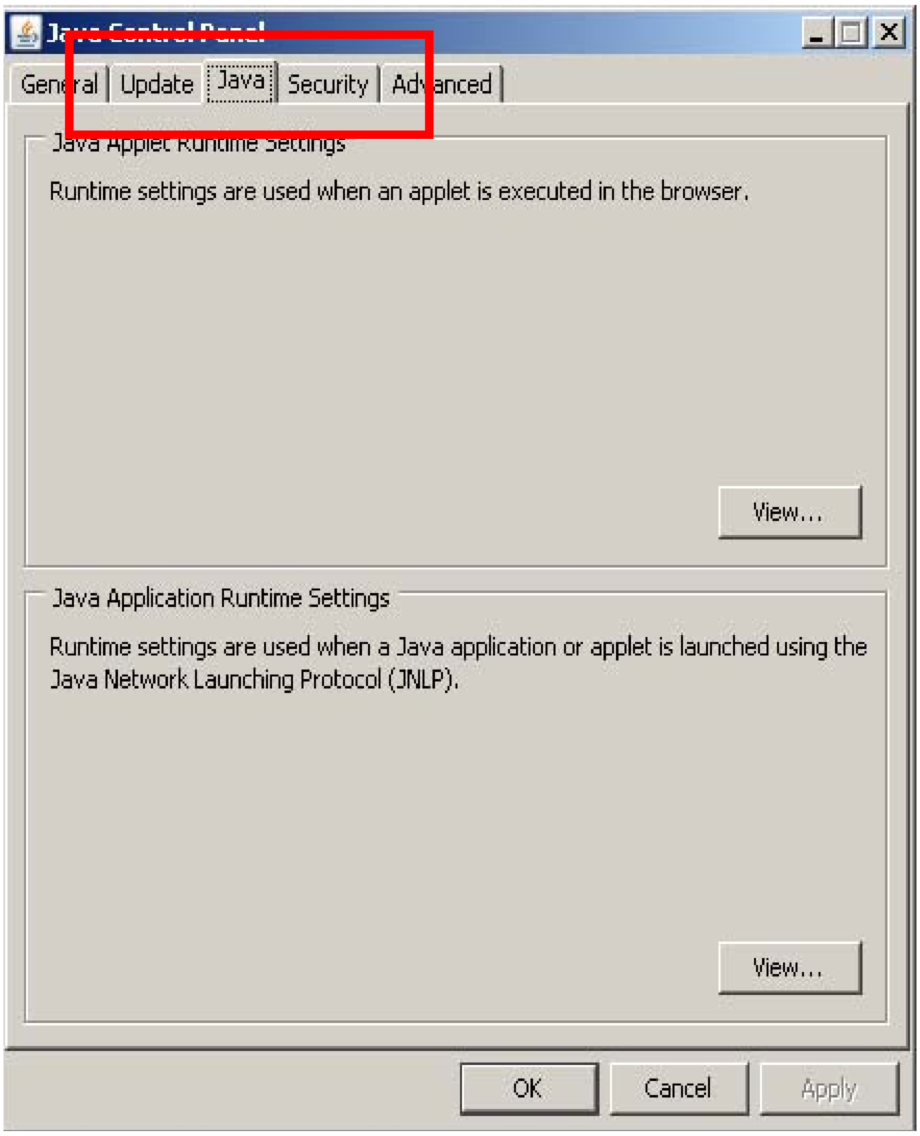
5.2.3.2. Memory Setting

The default amount of maximum Java heap memory that Java applets can use is 96 MB(64 MB up to Java 1.6.0_6). You have to set the max/min value of the Java heap memory in order to use the JENNIFER client more stable. In most cases, the default setting is sufficient, but if there are too many agents or if the amount of processing is too great, then you need to set the maximum Java heap memory to a higher amount.

In Windows OS, you can set the Java heap memory amount of your Java plug-in as follows.

- Move to the control panel.
- Double-click on the Java icon in the control panel, and the Java control panel will appear. Click the Java tab.

Figure 5-2:Java Control Panel



Java Applet Runtime Settings

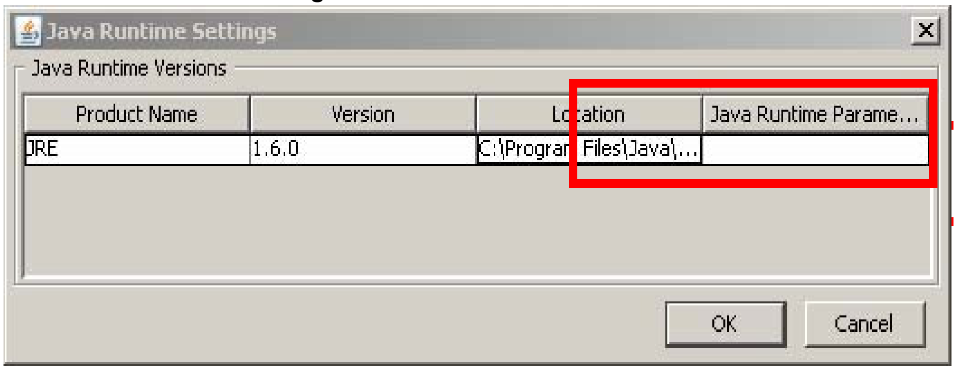
Runtime settings are used when an applet is executed in the browser.

Java Application Runtime Settings

Runtime settings are used when a Java application or applet is launched using the Java Network Launching Protocol (JNLP).

- If you click the **[View]** button in the Java applet runtime setting menu, the Java runtime setting screen will appear. If you set multiple Java, select the one that is to be used.

Figure 5-3:Java Runtime Setting



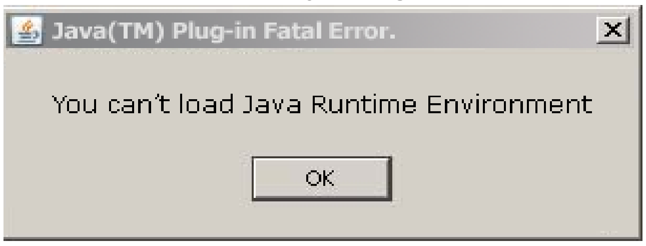
Product Name	Version	Location	Java Runtime Parame...
JRE	1.6.0	C:\Program Files\Java\...	

- For instance, if you want to set the minimum and maximum Java heap memory to 100MB and 150MB, you should enter the following in the Java runtime variable column as follows:

```
-Xms100m -Xmx150m
```

The maximum Java heap memory is affected by the computer specifications and environment. If you exceed the maximum setting, the web browser will be shut down or the following error message will appear:

Figure 5-4:Excessive Max Java Heap Memory Setting Errors



You can't load Java Runtime Environment

Since there is no clear limit for the maximum permissible Java heap memory, a user must find the optimum setting for his or her needs through a process of trial and error.

5.2.3.3. Optimizing CPU Utilization

You can check the java.exe indicating Java applet CPU utilization on the Process tap in Task Manager.

You can see the CPU utilization while the JENNIFER dashboard is opened.

You can optimize the CPU utilization using several options. Set an option among the belowign options and check the CPU utilization. How to set an option is the same way to set the Java Heap Memory.

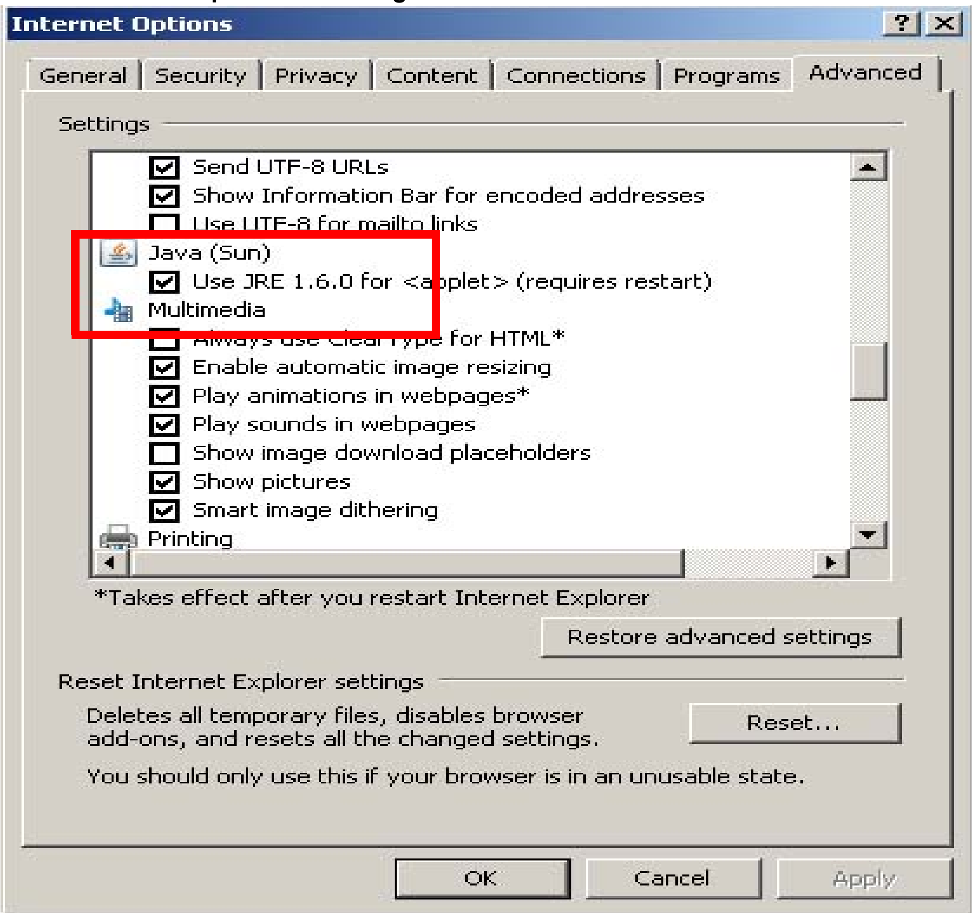
```
-Dsun.java2d.d3d=false
-Dsun.java2d.ddoffscreen=false
-Dsun.java2d.noddraw=true
```

Untimately, only set the option indicationg the least CPU utilization. Depending on your hardware machine, you can use the least CPU utilization as you do not set any option.

5.2.3.4. Web Browser Setting

If you use Microsoft IE 7.0 or IE 8.0, then you should activate the Java plug-in by selecting [Tools | Internet options] from the menu. After clicking the [Advanced] tab in the internet options menu, scroll to the bottom of the list, and you will be able to see the Java plug-in information. Here, you should activate the Java plug-in.

Figure 5-5:Internet Explorer 8.0 Setting



- [✓] Send UTF-8 URLs
- [✓] Show Information Bar for encoded addresses
- [ ] Use UTF-8 for mailto links

Java (Sun)
- [✓] Use JRE 1.6.0 for <applet> (requires restart)

Multimedia
- [ ] Always use ClearType for HTML*
- [✓] Enable automatic image resizing
- [✓] Play animations in webpages*
- [✓] Play sounds in webpages
- [ ] Show image download placeholders
- [✓] Show pictures
- [✓] Smart image dithering

Printing

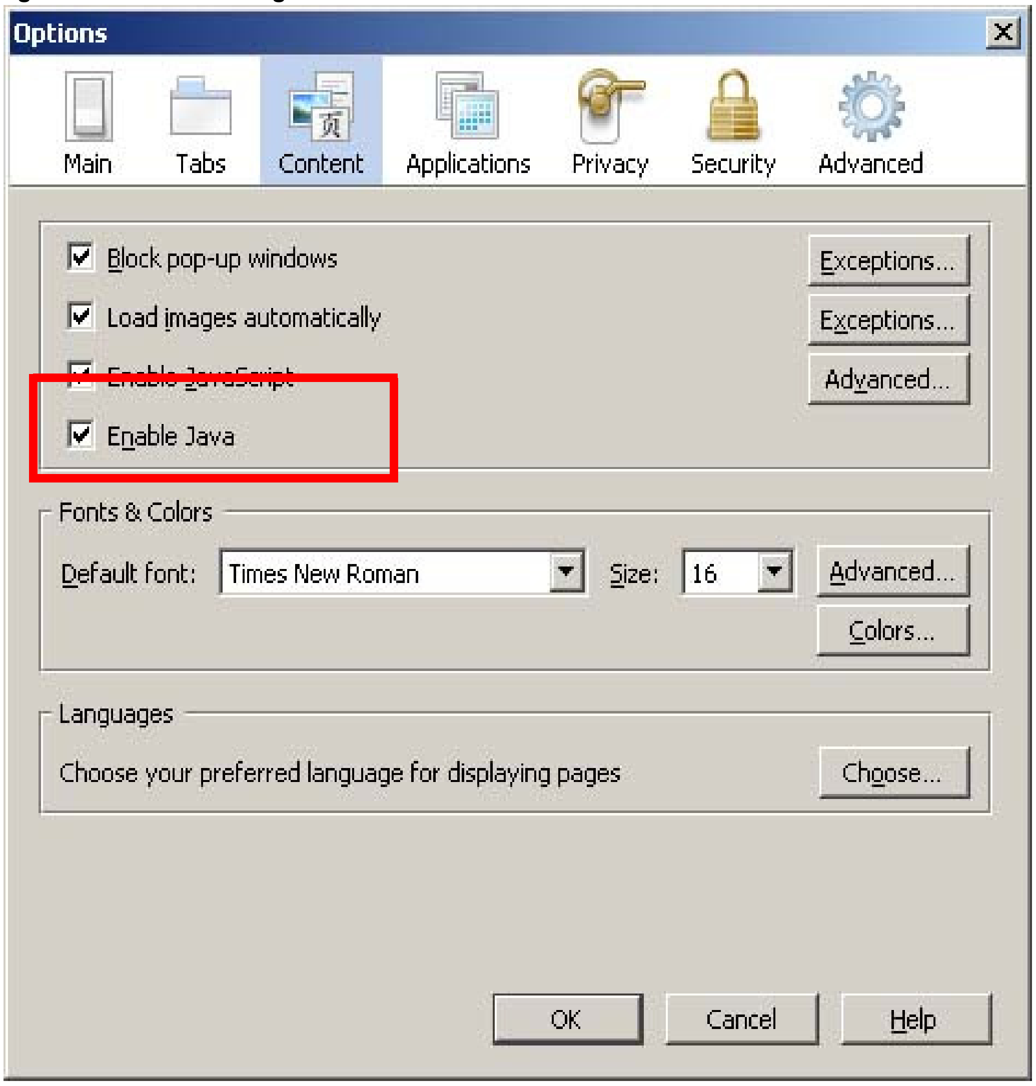
*Takes effect after you restart Internet Explorer

Deletes all temporary files, disables browser add-ons, and resets all the changed settings.

You should only use this if your browser is in an unusable state.

If you use Mozilla Firefox 3.0, you should activate the Java plug-in by selecting [Tools | settings] from the menu. After clicking the [Content] tab in the setting window, select the Java use option. By default, this option should be activated.

Figure 5-6:Firefox Setting



☑ Block pop-up windows
☑ Load images automatically
☑ Enable JavaScript
☑ Enable Java

Fonts & Colors

Default font: Times New Roman Size: 16

Languages

Choose your preferred language for displaying pages

5.2.3.5. Flash Player

Some aspects of the JENNIFER client are implemented through flash files. This requires the installation of Adobe Flash Player 9.0 or higher.

5.3. User Interface Structure

The JENNIFER user interface is essentially comprised of the upper area, the toolbar, the board area and the main area.

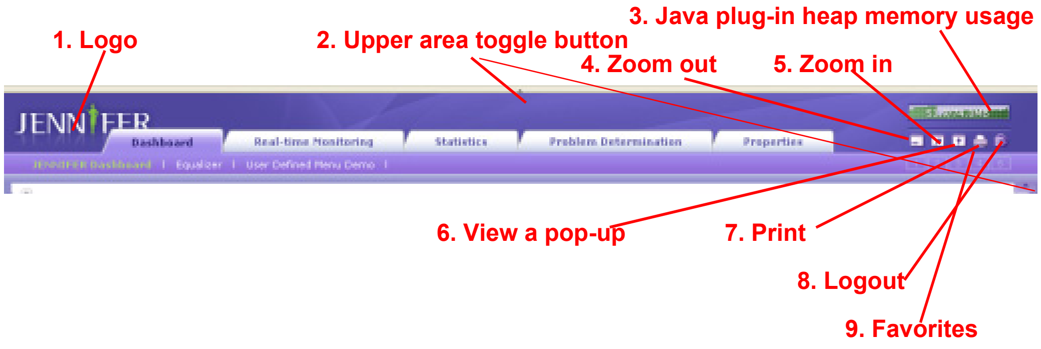
Table 5-1: User Interface Structure

Area	Descriptions
Menu bar	Provides the menu and the common functional buttons.
Toolbar	Provides a variety of tools and news
Board area	Provides the content of real-time alerts and the service status
Main area	Provides a range of information according to selected menus

5.3.1. Menu Bar

The menu bar is comprised of three levels. Most of the JENNIFER functions can be accessed through these menus.

Figure 5-7:Menu Bar



The main functions of the menus are as follows:

1. Logo - The initial setting is a JENNIFER logo image. If desired, you can replace it with another image.
2. Menu bar toggle button - If you click the menu bar toggle button, then the upper area menu will be hidden. If you click it when the menu bar is hidden, then it will reappear.

3. Java plug-in heap memory usage charts and contextual menu - This displays the Java plug-in's heap memory usage. If you click on this chart, a contextual menu will appear. Using this menu, you can perform various functions (Language and style change, GC, file management, BBS option setting for saving screen capture, etc).
4. Zoom-out button - If you click this button, the size of the main area will shrink by 10%. Using this button, you can't zoom out the size of the main area less than 10%.
5. Zoom-in button - If you click this button, the size of the main area will increase by 10%. There is no limit when you increase the size of the main area.
6. Pop-up button - Use this button to open the current screen in a new window. The new window will not include the menu bar. The default is that the pop-up screen is created as a same window. If you click this button by pressing the Shift key, you will see the new different pop-up window.
7. Print button - Prints the current screen. If you want to print out the dashboard screen properly, use the Microsoft IE 7.0 or IE 8.0
8. Log-out button - If you want to log out, click the log-out button. After logging out, you will be directed to the log-in screen.
9. Favorites - Use this button to go to your favorite menu options.

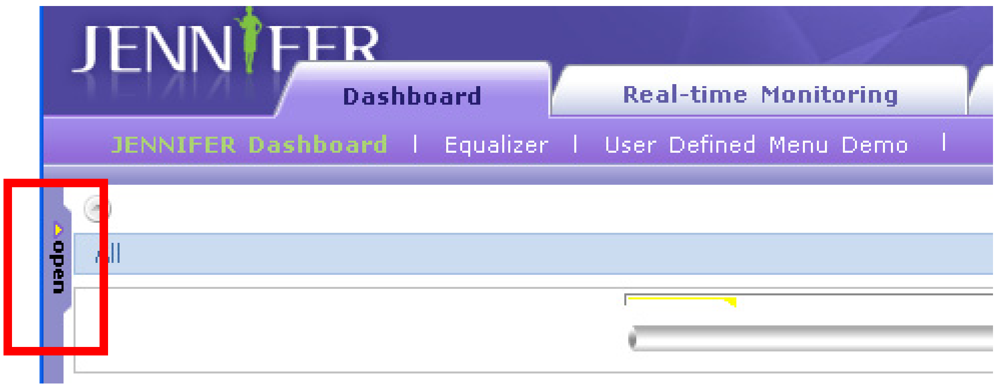
If you set the ui_hide_header option of the JENNIFER server to true, you can have the menu bar hidden in the browser.

```
ui_hide_header = true
```

5.3.2. Toolbar

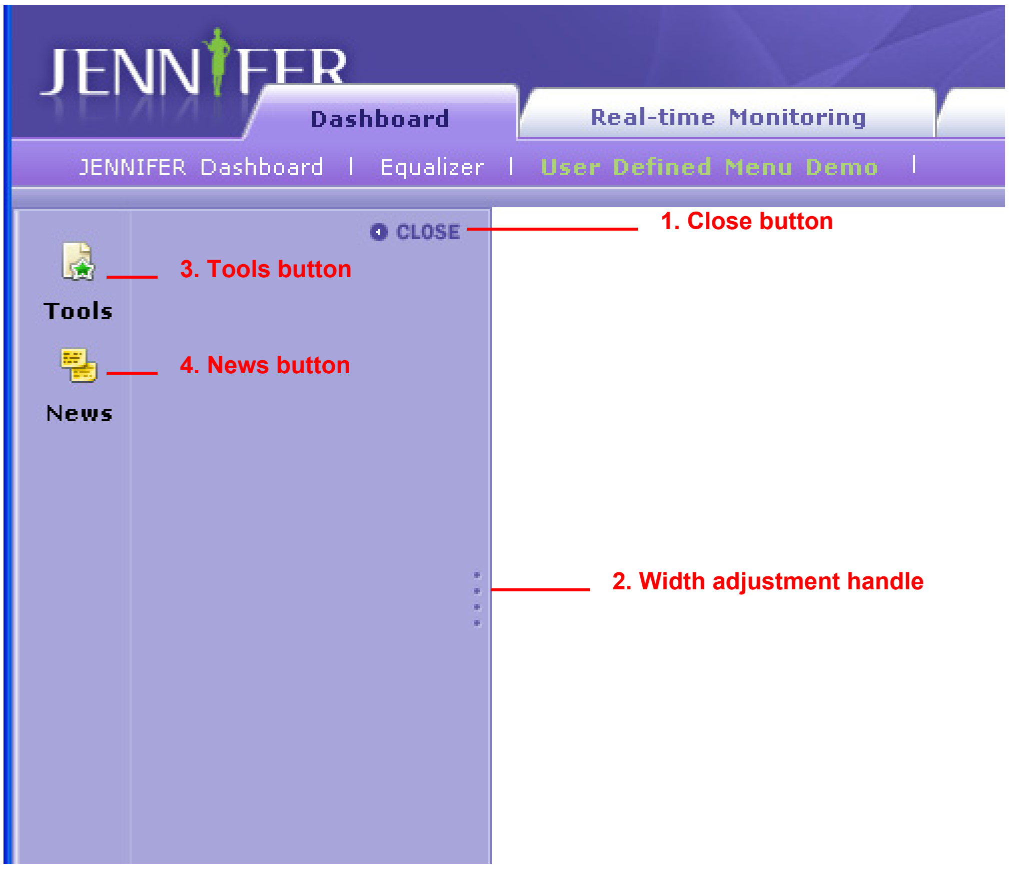
The toolbar provides a range of different tools, as well as the JENNIFER news.

Figure 5-8:Opening the Tool Bar



If you click the [open] button, the toolbar will appear.

Figure 5-9:Toolbar



- Close button - If you click the close button, the toolbar area will disappear.
- Width adjustment handle - This is used to adjust the width of the toolbar.
- Tools button - If you click this button, the tools list will appear.
- News button - If you click this button, the news from JENNIFER will appear.

The main functions of the tool buttons are as follows:

- Server Control Center - Use this button to see the major data from JENNIFER server such as network setting information, Java heap memory utilization, X-View processing status, think time, etc.
- Alert List - Use this button to see the recent and today's alert lists and to search the past alerts.
- Query Tool- This is used to execute SQL for the database used by JENNIFER server.

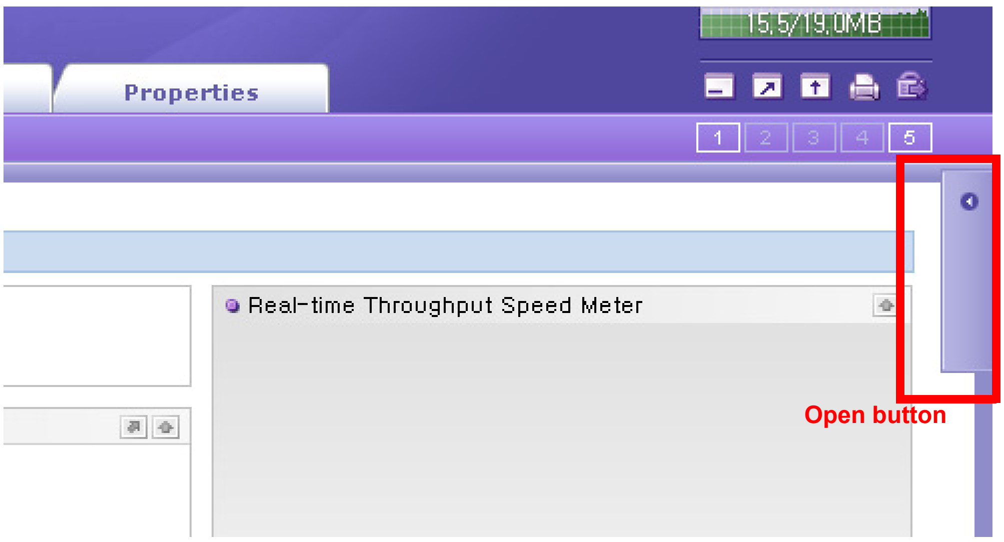
- Build Query - This is used to combine binding parameter and constant parameter for an SQL.
- REMON List - This is used to display REMON data list that is transmitted to JENNIFER server.
- Table Space Reorg - This is used to reorganize the table space. This function is only available when you use the Apache Derby as a JENNIFER server DB.

5.3.3. Board Area

The board area provides a range of information, including real-time alerts, daily service status and real-time service status. The board area is typically hidden in the right of the web browser, and it automatically appears along with an alarm sound when an error occurs. In the board area, you can view the overall status of the monitored Java application.

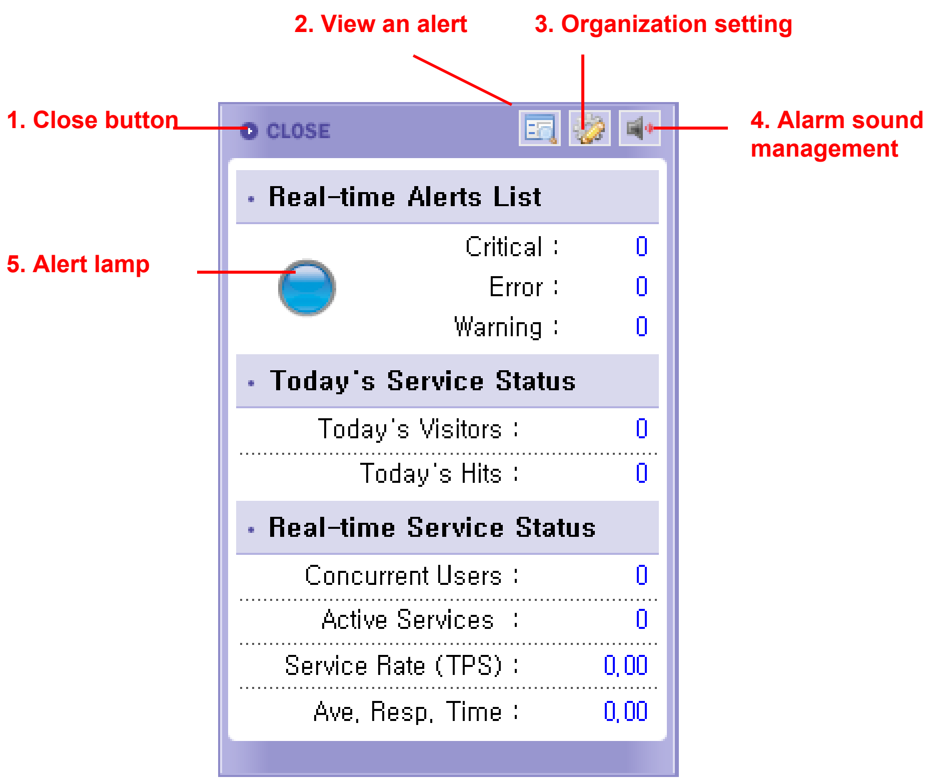
Notice: The board area is only displayed for the user group having an authority about board area.

Figure 5-10:Opening the Board Area



If you click the [Open] button, the board area that was hidden in the right side of the web browser will be launched.

Figure 5-11:Board Area



- Real-time Alerts List

Critical :	0
Error :	0
Warning :	0

- Today's Service Status

Today's Visitors :	0
Today's Hits :	0

- Real-time Service Status

Concurrent Users :	0
Active Services :	0
Service Rate (TPS) :	0.00
Ave. Resp. Time :	0.00

1. Close button - If you click the close button, the board area will disappear.
2. View an alert - To check the details of an alert, press this button.
3. Organization setting - An alert has three different levels - critical, error and warning. When an alert has been generated, the board area will appear, regardless of the alert type. If you want to see the board area only for alerts of a specific type, then click this button or right-click in the board area. A contextual menu will appear that allows you to change the option.
4. Alarm sound management - If you click this button, then you will not hear the alarm sound. If you want to hear it, click this button, again.
5. Alert lamp - Under normal conditions, the alert lamp is in blue in color. In the event of an error, it will change to a red color. If you click on the lamp when it is red, it will change to blue, reflecting the fact that you have acknowledged the alert.

The following is a list of real-time alerts defined by JENNIFER.

Table 5-2: Real-time Alerts

Alerts	Descriptions
Critical	# of today’s occurred critical alerts
Error	# of today’s occurred error alerts

Alerts	Descriptions
Warning	# of today’s occurred warning alerts

The following is a list of daily service status.

Table 5-3: Daily Service Status

Daily Service Status	Descriptions
# of today’s visitors	Today’s visitors
# of today’s hits	Today’s hits

The following is a list of real-time service status.

Table 5-4: Real-time Service Status

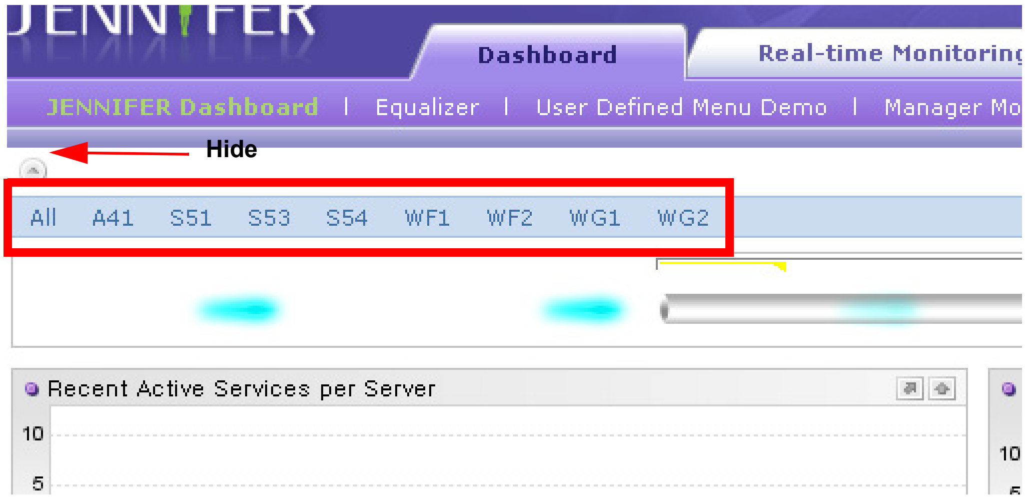
Real-time Service Status	Descriptions
# of concurrent users	Real-time concurrent users.
# of active services	Real-time active services
Rate of throughputs	Real-time throughputs
Average response time	Real-time average response time

5.3.4. Main Area

This area displays various monitoring information. If you click on the menu bar, the content of the main area will be changed.

There is the area to select JENNIFER agent in the most upper area of JENNIFER dashboard.

Figure 5-12:JENNIFER Agent Area


JENNIFER
Dashboard
Real-time Monitoring
JENNIFER Dashboard | Equalizer | User Defined Menu Demo | Manager Mo
Hide
All A41 S51 S53 S54 WF1 WF2 WG1 WG2
Recent Active Services per Server
10
5

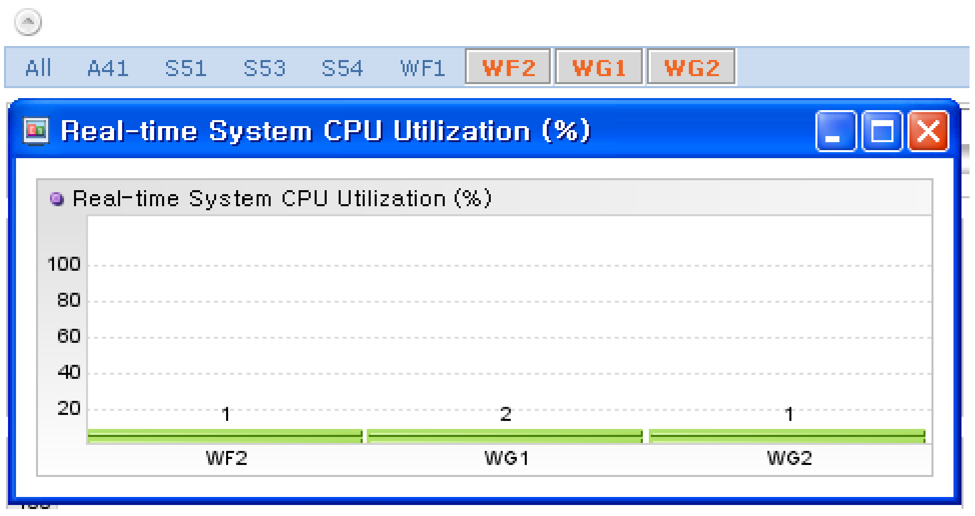
According to the characteristics of the JENNIFER dashboard, items on the JENNIFER agent selecting area are different.

- All- This button is used to display every JENNIFER agent on the dashboard. You can also select all JENNIFER agent at once using this button when you need to search the data about the all JENNIFER agents.
- Server- You can see this only in the **[Properties| Configuration]** menu. Use this button to change the JENNIFER server option.
- JENNIFER Agent- This displays JENNIFER agent. You may select several agents at the same time or choose just one based on the characteristics of the JENNIFER dashboard. If the JENNIFER agent stops, you will see the [Stop] beside the agent name. If there is the JENNIFER agent that you monitored in the past, not current, the agent appears in the JENNIFER agent selected area through the **[Statistics]** menu. And you will see the [Unconfigured] beside the agent name.

Notice: You can toggle the JENNIFER agent selecting area using the **[Hide]** button.

If you select a certain agent, the dashboard displays the charts that is only related to the agent. If you click the [All] button in this situation, all agents are displayed in the dash-board. If you do not select any agent, all JENNIFER agents are displayed. (Default value is [All])

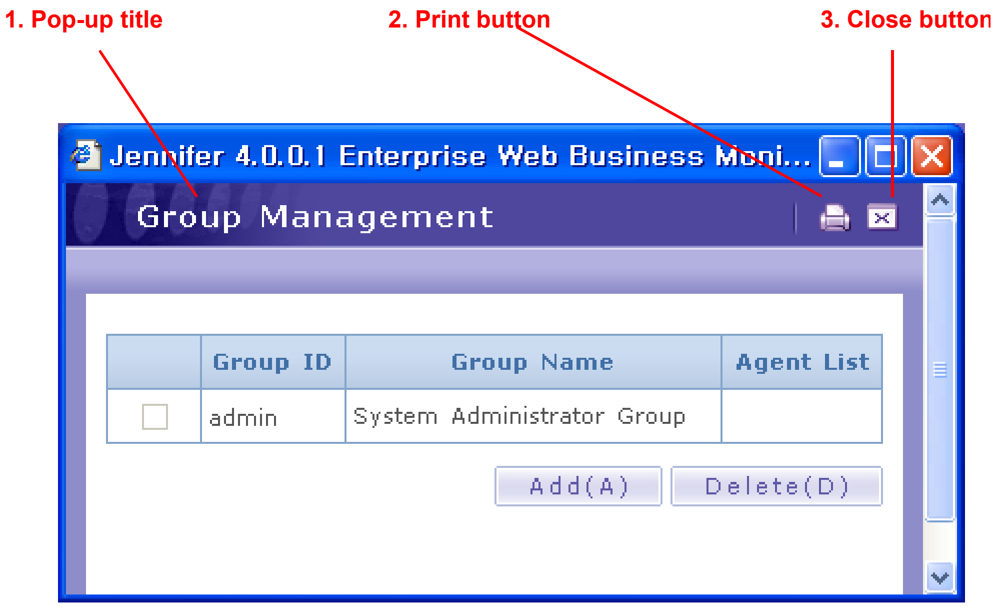
Figure 5-13:Selecting JENNIFER Agent


All
A41
S51
S53
S54
WF1
WF2
WG1
WG2
Real-time System CPU Utilization (%)
Real-time System CPU Utilization (%)
100
80
60
40
20
1
2
1
WF2
WG1
WG2

5.3.5. Pop-up Window

The following is the description about a pop-up window.

Figure 5-14:Pop-up Window



	Group ID	Group Name	Agent List
☐	admin	System Administrator Group	

1. Pop-up title- This displays the title of pop-up window.
2. Print button- If you click this button, a pop-up window will be printed.

3. Close button- If you click this button, a pop-up window will be closed.

5.4. Main Functions of the User Interface

5.4.1. Changing the Style

Styles define the theme of the JENNIFER user interface. The user can select between BLACK and WHITE. The default style is BLACK. The style can be changed in the following manner:

1. If you click the Java plug-in heap memory chart in the upper area, the contextual menu will appear.
2. From the contextual menu, select the style change option, and then select a new style to use.

5.4.2. Multi-language Service

The JENNIFER user interface supports various languages such as the English, Japanese, Chinese, French and Korean languages.

Notice: UTF-8 encoding is used between the JENNIFER server and the web browser. You may not change it.

5.4.2.1. Option Setting

The multi language list can be set using the supported_language_list option.

```
supported_language_list = en,ko,ja,zh,fr,es,pt
```

Each language code is separated by a comma[,]. If you do not wish to use a certain language, you should remove the language code from the option list.

Table 5-5: Language and Language Code

Language	Language Code
English	en
Korean	ko
Japanese	ja

Language	Language Code
Chinese	zh
French	fr
Spanish	es
portuguese	pt

5.4.2.2. Changing the Language

The basic language is set according to the language that has been set for your web browser. If you want to switch to a different language, proceed as follows:

1. Click the Java plug-in heap memory chart in the upper area menu, and the contextual menu will appear.
2. From the contextual menu, select the language change option, and select the language that you wish to use.

5.4.2.3. Changing the Message

You can change an individual message on your own. For instance, if you want to replace the first menu, **[Dashboard],** with **[Main screen]**, do the following:

1. Click the Java plug-in heap memory chart in the upper area menu. The contextual menu will appear.
2. After selecting the language change option from the contextual menu, select a key.
3. Check the key of the message to be changed. The key for **[Dashboard]** is [menu.dashboard].
4. Move to the **[Properties | Configuration | Message Setting]** menu.
5. Click the [Add] button at the bottom.
6. After entering the key, the value, and the language, click the [Save] button. For instance, enter [menu.dashboard] for the key, **[Main screen]** for the value and [en] for the language.
7. Click the [Refresh] button at the bottom.
8. After logging out, log in again to view the changed message.

5.4.3. Saving the Screen Capture on the BBS

You can save the current screen on the BBS. If you want, do the following:

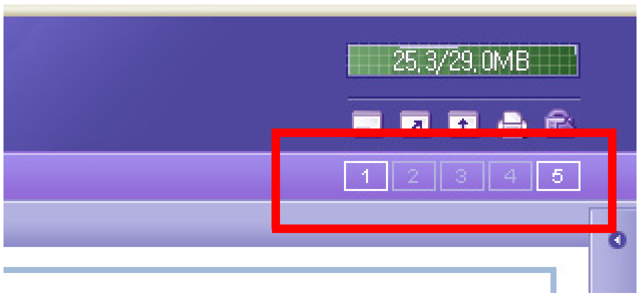
1. Click the Java plug-in heap memory chart in the upper area menu. The contextual menu will appear.

2. Click the **[Save Screen Capture to BBS]** from the context menu.
3. You can confirm the saved image on the **[Properties | BBS]** menu.

5.4.4. Favorites

You can save up to five menu options that you use frequently. The numbers 1, 2, 3, 4 and 5 in the lower right corner of the upper area menu represent your favorites menu. If the number is white, then it means that a shortcut has been assigned to it. If it is grey, then no shortcut has been assigned to it. If you place your mouse cursor on the white number, you can see the favorite menu option that it is linked to. If you click on the number, you will be moved to the favorite menu option.

Figure 5-15:Favorites



For instance, if you want to select the **[Real-time Monitoring | Profile]** menu as your favorite menu, proceed as follows.

1. Move to your favorite menu. In other words, move to the **[Real-time Monitoring | Profile]** menu.
2. Double click on an arbitrary grey number in the favorite menu. The **[Real-time Monitoring | Profile]** menu will then be linked to the number, and the number will turn white.

If you want to remove a favorite, then double-click on the grey number.

5.4.5. Changing the Logo and the Title

In the **[Properties | Configuration | UI Setting]** menu, you can change the logo and the title.

Table 5-6: Changing Logo and Title

Menu	Descriptions
Logo	Logo image in the menu bar
Login Logo	Logo image in the login window
Title	Title message in the web browser
Copyright	Copyright message in the login window

If you want to switch back to the default logo after changing it, first select the initialization check box and click the [Save] button.

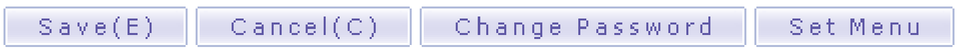
Notice: If you upload the log image, JENNIFER only supports png and gif file type.

5.4.6. Using the Hotkeys

For the convenience of the user, JENNIFER supports the use of hotkeys. For example, if you press the left-cursor key and the Ctrl key simultaneously, then the screen will be reduced in size. On the other hand, if you press the right-cursor key and the Ctrl key simultaneously, it will be magnified. These two hotkeys correspond to the zoom in and out buttons in the menu bar. In addition, if you press the Ctrl key and the up-cursor key simultaneously, the current screen will appear in a new window. This hotkey corresponds to the pop-up button in the menu bar. If you press the Ctrl key and the down-cursor key simultaneously while the new window is open, then the new window will be closed. This hotkey corresponds to the close button in the pop-up menu.

The hotkeys for the buttons in the main area appear next to their names. The hotkey for the button called **(E)** is the Enter key. For other hotkeys, you need to press a combination of keys simultaneously, usually consisting of the Ctrl or the Alt key and a letter. If no hot-key is displayed next to the function, there is no hotkey assigned.

Figure 5-16:Using the Hotkeys



Warning: If there is a cursor on the HTML text area, the Enter key is not a hotkey, but it will change the line, instead. So, if you want to use the Enter key as your hotkey, you need to move the cursor away from the HTML text area.

5.4.7. Java Plug-in Garbage Collection

If you want to perform garbage collection on the Java plug-in heap memory, proceed as follows.

1. If you click on the Java plug-in heap memory chart in the upper area menu, the contextual menu will appear.
2. From the contextual menu, select ‘garbage collection’.

5.4.8. Deleting Temporary Java Plug-in Files

For performance improvement, the application names, the SQL, and the previous general performance data should be stored in the user’s computer. If you want to delete these files, proceed as follows.

1. If you click on the Java plug-in heap memory chart in the upper area menu, the contextual menu will appear.
2. From the contextual menu, select ‘Delete temporary files’.

5.5. Menu Structure

The JENNIFER menu structure is comprised of the major 5 upper menu and its child menu.

5.5.1. Dashboard

By selecting **[Dashboard]** from the menu, you can comprehensively monitor a range of performance data. The JENNIFER dashboard focuses on the effective monitoring of the Java application.

5.5.2. Real-time Monitoring

This menu provides the data displaying the real-time service status through the **[X-View]**, **[Profile]**, **[Application]** and **[Real-time Status]** menu.

5.5.3. Statistics

On this menu, user can search the past X-View data, the statistical data for the past application throughput and other variety statistics status data.

In addition, JENNIFER provides a daily, weekly, monthly **[Report]**, **[CRUD Matrix]** and **[Query Tool]** menu.

5.5.4. Problem Determination

User can detect and analyze the performance problem and error on this menu. For instance, you can use the menu to analyze collection object, file and socket and HTTP session object.

In addition, you can search the Java system properties, loaded class list, class or JAR file, modified file and unused class, etc.

5.5.5. Properties

User can configure the JENNIFER server/JENNIFER agent option and domain/node on this menu. It also provides license/user authority/business group/ BBS management setting menu.

6 Service & User Monitoring

6.1. Java Application Service & Transaction

The Java application is comprised of multiple application services. An application service is not a fixed concept, but is a statistical unit that is used as a reference for collecting the performance data, such as hits and response time.

Notice: The application service is sometimes called the application. However, this has a different connotation than the use of the term application to imply an entire program. you must understand the difference in context.

In general, the application service is based on tasks. For example, the application services in a shopping mall application could be viewing the product list, viewing the details, putting products in carts, ordering and payment. However, as one task could be done at many detailed levels, and it is desirable to collect the performance data from each of the detailed levels, the application services can be classified into more detailed levels than the tasks.

First of all, in the web application, the URI is a typical way to refer to the application service.

```
/view.jsp
/order.jsp
/pay.jsp
```

However, sometimes the same URI is used for every request, and the request parameters are used to distinguish the application service.

```
/service.do?action=view
/service.do?action=order
/service.do?action=pay
```

In this case, the parameters can be delivered by the HTTP POST method, not the HTTP GET method. For this reason, it is necessary to distinguish the application services using other a means other than the URI.

In addition, for non-web applications, distinguish the application service by using a specific method of a specific class.

```
example.ViewManager View method of the class
example.OrderManager Order method of the class
example.PayManager Pay method of the class
```

A transaction implies that the Java application processes the application after the user requests the application service. Hits and average response time are performance data based on the results of transaction processing, while the application service plays the role of metadata.

6.1.1. Application Start Point

The application service start point is the point at which the transaction begins.Strictly speaking, the transaction start point is the network module in the front that receives the client’s request, but since most requests are received at the same port number, it is impossible to distinguish the application services at this point. Since transaction-related performance data is collected from the application service, the application start point should be the first point at which you can distinguish the application services.

The method for setting the application start point varies depending on whether the monitored Java application is WAS-based or not.

6.1.1.1. Web Application

The WAS-based application start point is the service (ServletRequest, ServletResponse) method of the javax.servlet.http.HttpServlet class. In this case, no additional configuration is required.

However, to monitor the overriding sublet that redefines this method, you have to set the sublet using the http_service_class option of the JENNIFER agent. Multiple entities are separated by semicolons [;]. For example, if the example.AServlet and example.BServlet classes inherit the javax.servlet.http.HttpServlet class and redefine the service method, change the configuration as follows:

```
http_service_class = example.AServlet;example.BServlet
```

To set the modified http_service_class option, you have to restart Java application that installed the JENNIFER agent.

Although not registered in the http_service_class options, the next sublet provided by the WAS automatically becomes the application start point. These classes are mostly the parent sublet classes of the JSP for each WAS. Therefore, most WAS, it is not necessary to perform additional configuration for the JSP.

```
javax.servlet.http.HttpServlet
com.evermind.server.http.JSPServlet
com.caucho.jsp.JavaPage
jrun.jsp.runtime.HttpJSPServlet
allaire.jrun.jsp.HttpJSPServlet
weblogic.servlet.jsp.JspBase
```

If you want to use the method having the non-service type of ServletRequest and ServletResponse classes as the application start point, use the http_service_method option of the JENNIFER agent.

```
http_service_method = doFilter
```

In the http_service_method option, the first two parameters can use the ServletRequest and ServletResponse method, regardless of how many parameters exist. However, it is not possible to assign more than two methods. As the service method is automatically registered, you can easily monitor the sublet.

If you want to set the filter class implementing the javax.servlet.Filter interface as the application service start point, proceed as follows. For example, let’s assume that the exampl.FrontFilter class implementing the javax.servlet.Filter interface is the application service start point.

```
http_service_class = example.FrontFilter
http_service_method = doFilter
```

You can avoid monitoring request URLs that you do not need to collect the performance data from.

First, use the ignore_url option of the JENNIFER agent to define the non-monitored URLs.

```
ignore_url = /check.jsp,/test.jsp
```

Next, you can use the ignore_url_postfix option of the JENNIFER agent to skip URLs that end in a certain pattern..

```
ignore_url_postfix = .gif .GIF .jpg .JPG .zip .html .HTML .txt .css .js
.swf .htc .HTC .png .PNG
```

Using the ignore_url_prefix option of the JENNIFER agent, you can skip the URLs that begin with a certain pattern.

```
ignore_url_prefix = /admin,/console
```

6.1.1.2. Non-web Java

For a non-WAS-based general Java application, you can set the application service start point using the tx_server option of the JENNIFER agent. To, set the modified option starting with tx_server, you have to restart Java application that installed the JENNIFER agent.

For example, let's assume that the example.OrderManager class processes the order. If you want to set this as the application service start point, use the tx_server_class option of the JENNIFER agent.

```
tx_server_class = example.OrderManager
```

Let's assume that the example.PayManager class processes the payment. If you also want to set this class as the application service start point, enter it in the tx_server_class option after a [;]. If you want to have two or more entries in the option related to the application service start point, use a semicolon [;] as a separator.

```
tx_server_class = example.OrderManager;example.PayManager
```

The application service start point implies a certain method of a certain class. Therefore, if you set the configuration as shown earlier, every method of the example.OrderManager class is recognized as the application service start point.

Thus, if you want to set only a certain method as the application service start point, use the tx_server_target_method option of the JENNIFER agent.

```
tx_server_target_method = order
```

However, if some of the classes set as the application service start points share the same method, and you want to set only certain methods as the application service start point, set them as follows:

```
tx_server_target_method = example.OrderManager.order
```

On the other hand, if you want to exclude certain methods from the application service start point, use the tx_server_ignore_method option of the JENNIFER agent.

```
tx_server_ignore_method = example.OrderManager.someMethod
```

You can also set the application service start point using the accessor of the method. For example, if you want to set the public accessor and the method without an accessor as the application service start point, set it as follows:

```
tx_server_access_method = public;none
```

The following parameters can be set using the tx_server_access_method option.

- public - public accessor
- protected - protected accessor
- private - private accessor
- none - no accessor

If all classes that inherit certain classes are the application service start points, use the tx_server_super option of the JENNIFER agent. For example, if you want to set all the classes inheriting example.BaseAction as the application service start points, set them as follows:

```
tx_server_super = example.BaseAction
```

In this case, only the classes that directly inherit from example.BaseAction become the application service start points. For example, if class A inherits from the example.BaseAction class and class B inherits from class A, then class A is recognized as the application service start point, but class B is not.

If all the classes implementing a certain interface are the application service start points, use the tx_server_interface option of the JENNIFER agent. For example, if you want to set all classes implementing the example.IAction interface as the application service start points, set them as follows:

```
tx_server_interface = example.IAction
```

However, in this case only the classes directly implementing the interface become the application service start points. For example, if class A implements the example.IAction interface and class B inherits from class A, then class A is recognized as the application service start point, but class B is not.

Using the class name, you can set the application service start point. In this case, use the tx_server_prefix, tx_server_postfix, and tx_server_ignore_prefix options of the JENNIFER agent. For example, if you want to set all classes whose names begin with example.action as the application service start points, set them as follows:

```
tx_server_prefix = example.action
```

In addition, if you want to set all classes whose names ends with Action as the application service start points, set them as follows:

```
tx_server_postfix = Action
```

If you want to exclude classes with certain names from the application service start points, use the tx_server_ignore_prefix option of the JENNIFER agent. This option can override all other options.

```
tx_server_ignore_prefix =
```

With the tx_server option of the JENNIFER agent, you can perform the same configurations in various ways. Using the tx_server_target_method option, you can set only the method that is the actual application service start point.

For example, let's assume that we have the pkg.ClassA and pkg.ClassB class, and that the run method of ClassA and the process method of ClassB are the application service start points. If the run method also exists in ClassB and it is irrelevant to the application service start point, then you can set it as follows:

```
tx_server_class = pkg.ClassA;pkg.ClassB
tx_server_target_method = run;process
tx_server_ignore_method = pkg.ClassB.run
```

Alternately, you can set it as follows here:

```
tx_server_class = pkg.ClassA;pkg.ClassB
tx_server_target_method = pkg.ClassA.run;pkg.ClassB.process
```

6.1.2. Java Application Service Naming

The application service names are saved in the APPLS table. Essentially, the web application uses the URI as the application service name. If set using the tx_server option, then the class or the method name will be used as the application service name. However, you can also set the application service name using the parameter value or the parameter or the returned value of the method.

Application service naming is setting the application service names in various ways, as follows:.

6.1.2.1. Web Application

The web application uses the request URI as the application service name. However, due to the development of the web application framework, it is sometimes difficult to distinguish the application service with the URI only. In this case, you can use the request parameter to specifically set the application service name. To achieve this, set the request parameter name using the url_additional_request_keys option of the JENNIFER agent.

```
url_additional_request_keys = key1,key2
```

In this case, use the parameter value returned by the following code to determine the application service name.

```
request.getParameter(”key1”);
request.getParameter(”key2”);
```

For example, the application service name for a certain request can be given as follows:.

```
http://127.0.0.1:8080/app.jsp?key1=a&key2=b
   → /app.jsp+(key1=a&key2=b)
http://127.0.0.1:8080/app.jsp?key1=a
   → /app.jsp+(key1=a)
http://127.0.0.1:8080/app.jsp?key1=a&key2=
   → /app.jsp+(key1=a)
http://127.0.0.1:8080/app.jsp
   → /app.jsp
```

You can also use some portions of the parameter value:

```
url_additional_request_keys = key1:5
```

According to the configuration shown above, the value returned by the following code is added to the application service name.

```
request.getParameter("key1").substring(0, 5)
```

In general, the separator used between the URI and the parameter is [?]. However, in the WAS for mobile terminals, other characters are used. You can set it using the uri_separator option of the JENNIFER agent.

```
uri_separator = ?
```

The url_additional_request_keys option is not applicable to the file upload request(multipart/form-data).

Using the ignore_url_post_request_parsing_prefixes option of the JENNIFER agent, you can disable the url_additional_request_keys option for the URI that begins with a certain name.

```
ignore_url_post_request_parsing_prefixes = /kesa.web,/insa/x_servlet
```

In addition, if the parameter value is multiple languages, and if multiple languages are processed by invoking the setCharacterEncoding method of the javax.servlet.http.HttpServletRequest object with sublet 3.2 or higher, then the multiple languages might not be correctly displayed. This is because the JENNIFER agent obtains the parameter value before the setCharacterEncoding method is called from the source code.

```
doGet(...) {
    request.setCharacterEncoding("KSC5601);
    ...
}
```

Therefore, in this case you must set the encoding type using the request_set_character_encoding option of the JENNIFER agent.

```
request_set_character_encoding = KSC5601
```

In addition, some solutions like AquaLogic have user information within URL. Next is an example about AquaLogic URL.

```
/x/gate/user_123_01_21/http;//ap1/xx
```

If you use this URI as an application service name, the application service name will increase in proportion to the number of user. For instance, the number of application is 1,000 with 1000 users, the application service name would be one million. In this case, the java.lang.OutOfMemoryError may occur in the JENNIFER server.

To solve this error, you can create a application service name with a certain URL pattern by setting the JENNIFER agent uri_starter option.

```
uri_start = /http
```

If you set the uri_start option of the JENNIFER agent into /http, the application service name will be the following.

```
/http;//ap1/xx
```

6.1.2.2. General Java Application

Essentially, the application service name is comprised of the class name and the method name. By using tx_server_ntype option of the JENNIFER agent, you can set it in various ways. The default is FULL, and other values include: SIMPLE, CLASS and METHOD. To set the modified tx_server_ntype option, you have to restart Java application that installed the JENNIFER agent.

```
tx_server_ntype = FULL
```

For example, if the process method of the pkg.ClassB class is the application service start point, then the application service name can be determined by the tx_server_ntype option.

- FULL - public void pkg.ClassB.process()
- SIMPLE - ClassB.process
- CLASS - ClassB
- METHOD - process

You can use the parameter value of the method that is the application service start point as the name. To do this, you must set the tx_server_using_param option of the JENNIFER agent to true. To set the modified tx_server_using_param option, you have to restart Java application that installed the JENNIFER agent

```
tx_server_using_param = true
```

If the tx_server_using_param option is set to true, the first java.lang.String type of parameter in the method that is the application service start point becomes the application service name. If there are no such parameters, then the application service name is determined by the tx_server_ntype option.

Also, as the application service name, you can use the parameter of the returned value of the method of the class that is called within the method that is the application service start point.

If you want to use the parameter of the method of the class that is called within the method that is the application service start point, set it as follows: To set the modified lwst_txserver_method_using_param option, you have to restart Java application that installed the JENNIFER agent

```
lwst_txserver_method_using_param = pkg.ClassC.f1(String)
```

In this case, the first java.lang.String type of parameter becomes the application service name.

If you want to use the returned value of the method of the class that is called within the method that is the application service start point, set it as follows: To set the modified lwst_txserver_method_using_return option, you have to restart Java application that installed the JENNIFER agent

```
lwst_txserver_method_using_return = pkg.ClassC.f2(String)
```

However, in this case the returned object should be a java.lang.String type.

In the lwst_txserver_method_using_param and lwst_txserver_method_using_return options, enter multiple entries separated by semicolons [;].

If multiple options are set, then the following priority should apply to determine the application service name.

- lwst_txserver_method_using_param or lwst_txserver_method_using_return
- tx_server_using_param
- tx_server_ntype

In the lwst_txserver_method_using_param or lwst_txserver_method_using_return option, the parameter or the returned value of the method that is executed last will become the application service name.

6.1.2.3. Logic Based Naming

Sometimes, it is impossible to determine the application service name at the start point. For example, if the SOAP protocol is being used, it is difficult to determine the name without parsing the XML at the application service start point. In this case, you can add the class and method names that begin with tx_naming in the logic flow where the transaction is executed to the application service names. To set the modified tx_naming option, you have to restart Java application that installed the JENNIFER agent

For example, let's assume that the critical class to determine the application service name is example.BizAction. If you want to add it to the application service names, use the tx_naming_class option of the JENNIFER agent..

```
tx_naming_class = example.BizAction
```

Next, let's assume that the example.LogAction class is also critical to determine the application service name. If you want to add this class to the application service names, enter it in the tx_naming_class option, using a semicolon [;] to separate it from the previous class. In addition to the tx_naming_class options, if you want to enter multiple entries in the option related to the application service name, use a semicolon as a separator.

```
tx_naming_class = example.BizAction;example.LogAction
```

When a certain method name of a certain class is added to the application service names, it is affected by the way you set the tx_naming_ntype option of the JENNIFER agent. Thus, if you set it as shown above, then all methods of the example.BizAction class that are called while processing the transactions will be added to the application service names.

Therefore, if you want to add only certain methods to the application service names, use the tx_naming_target_method option of the JENNIFER agent.

```
tx_naming_target_method = process
```

However, if some of the classes set as the application service start points share the same method and you want to add only certain methods to the application service names, set them as follows:

```
tx_naming_target_method = example.BizAction.process
```

On the other hand, if you want to exclude certain methods from the application service names, use the tx_naming_ignore_method option of the JENNIFER agent..

```
tx_naming_ignore_method = example.BizAction.someMethod
```

Using an accessor of the method, you can set the method to be added to the application service names. For example, if you want to add the public accessor and the method without an accessor to the application service name, proceed as follows:

```
tx_naming_access_method = public;none
```

The following parameters can be set in the tx_naming_access_method option.

- public - public accessor
- protected - protected accessor
- private - private accessor
- none - no accessors

If you want to add all classes inheriting from a certain class to the application service name, use the tx_naming_super option of the JENNIFER agent. For example, if you want to add all classes inheriting from example.BaseAction to the application service names, set it as follows:

```
tx_naming_super = example.BaseAction
```

However, in this case only the classes directly inheriting from it are added to the application service names. For example, if Class A inherits from the example.BaseAction class and class B inherits from class A, then class A is added to the application service name, but class B is not.

If you want to add all the classes implementing a certain interface to the application service name, use the tx_naming_interface option of the JENNIFER agent. For example, if you want to add all classes implementing the example.IAction interface to the application service names, set it as follows:

```
tx_naming_interface = example.IAction
```

However, in this case only the classes directly implementing the example.|Auction interface will be added to the application service names. For example, if class A implements the example.IAction interfaces and class B inherits from class A, then class A is added to the application service names, but class B is not.

Using the class names, you can set the class to be added to the application service names. In this case, use the tx_naming_prefix, tx_naming_postfix and tx_naming_ignore_prefix options of the JENNIFER agents. For example, if you want to add all classes whose names begin with example.action to the application service names, set it as follows:

```
tx_naming_prefix = example.action
```

In addition, if you want to add all the classes whose names end with Action to the application service names, set it as follows:

```
tx_naming_postfix = Action
```

If you want to exclude classes with certain names from the application service names, use the tx_naming_ignore_prefix option of the JENNIFER agent. This option overrides all other options.

```
tx_naming_ignore_prefix =
```

Essentially, the class name and method are added to the application service name using the tx_naming option configuration. You can perform various configurations using the tx_naming_ntype option of the JENNIFER agent. The default is CLASS and other values include FULL, SIMPLE and METHOD.

```
tx_naming_ntype = CLASS
```

For example, if you want to add the process method of the pkg.ClassB class to the application service names, the following messages will be added to the application service names, depending on how you set the tx_naming_ntype option.

- FULL - public void pkg.ClassB.process()
- SIMPLE - ClassB.process
- CLASS - ClassB
- METHOD - process

You can also add a parameter value that is not a class or a method name to the application service names. To achieve this, set the tx_naming_using_param option of the JENNIFER agent to true.

```
tx_naming_using_param = true
```

If the tx_naming_using_param option is set to true, then the first java.lang.String type of parameter is added to the application service names. If there is no such parameter, then the application service name is determined by the tx_server_ntype option.

In addition, you can add a returned value that is not a class or a method name to the application service names. To achieve this, set the tx_naming_using_return option of the JENNIFER agent to true.

```
tx_naming_using_return = true
```

When the tx_naming_using_return option is set to true, if a returned value is a java.lang.String type, it will be added to the application service name. It it is not, the tx_naming_ntype option should apply.

For example, let's assume that the following code is executed by txn.jsp.

```
<%
    example.BizAction bizAction = new example.BizAction();
    bizAction.process(request, response);

    example.LogAction logAction = new example.LogAction();
    logAction.process(request, response);
%>
```

and tx_naming is set as follows:

```
tx_naming_postfix = Action
tx_naming_ntype = CLASS
```

Consequently, the application service name is given as follows.

```
/txn.jsp+BizAction+LogAction
```

6.2. .NET Application Service & Transaction

The application is comprised of multiple application services. An application service is not a fixed concept, but is a statistical unit that is used as a reference for collecting the performance data, such as hits and response time.

Notice: The application service is sometimes called the application. However, this has a different connotation than the use of the term application to imply an entire program. you must understand the difference in context.

In general, the application service is based on tasks. For example, the application services in a shopping mall application could be viewing the product list, viewing the details, putting products in carts, ordering and payment. However, as one task could be done at many detailed levels, and it is desirable to collect the performance data from each of the detailed levels, the application services can be classified into more detailed levels than the tasks.

The URI is a typical way to refer to the application service in the web application.

```
/view.aspx
/order.aspx
/pay.asmx
/submitOrder.svc
```

However, sometimes the same URI is used for every request, and the request parameters are used to distinguish the application service.

```
/service.aspx?action=view
/service.aspx?action=order
/service.aspx?action=pay
```

In this case, the parameters can be delivered by the HTTP POST method, not the HTTP GET method. For this reason, it is necessary to distinguish the application services using other a means other than the URI.

In addition, for non-web applications, distinguish the application service by using a specific method of a specific class.

```
example.ViewManager View method of the class
example.OrderManager Order method of the class
example.PayManager Pay method of the class
```

A transaction implies that the application processes the application after the user requests the application service. Hits and average response time are performance data based on the results of transaction processing, while the application service plays the role of metadata.

6.2.1. Application Service Naming

The application service names are saved in the APPLS table. Essentially, the web application uses the URI as the application service name. However, you can also set the application service name using the parameter value or the parameter or the returned value of the method.

Application service naming is setting the application service names in various ways, as follows:

6.2.1.1. Web Application

The web application uses the request URI as the application service name. However, due to the development of the web application framework, it is sometimes difficult to distinguish the application service with the URI only. In this case, you can use the request parameter to specifically set the application service name. To achieve this, set the request parameter name using the url_additional_request_keys option of the JENNIFER agent.

```
url_additional_request_keys = key1,key2
```

In this case, use the parameter value returned by the following code to determine the application service name.

```
HttpContext.Current.Request.Params.Get(“key1”);
HttpContext.Current.Request.Params.Get(“key2”);
```

For example, the application service name for a certain request can be given as follows:

```
http://127.0.0.1:8080/app.aspx?key1=a&key2=b
→ /app.aspx+(key1=a&key2=b)
http://127.0.0.1:8080/app.aspx?key1=a
→ /app.aspx +(key1=a)
http://127.0.0.1:8080/app.aspx?key1=a&key2=
→ /app.aspx +(key1=a)
http://127.0.0.1:8080/app.aspx
→ /app.aspx
```

You can also use some portions of the parameter value:

```
url_additional_request_keys = key1:5
```

According to the configuration shown above, the value returned by the following code is added to the application service name.

```
HttpContext.Current.Request.Params.Get("key1").Substring(0, 5)
```

In general, the separator used between the URI and the parameter is [?]. However, in the WAS for mobile terminals, other characters are used. You can set it using the uri_separator option of the JENNIFER agent.

```
uri_separator = ?
```

The url_additional_request_keys option is not applicable to the file upload request(multipart/form-data).

Using the ignore_url_post_request_parsing_prefixes option of the JENNIFER agent, you can disable the url_additional_request_keys option for the URI that begins with a certain name.

```
ignore_url_post_request_parsing_prefixes = /kesa.web,/insa/controller
```

In addition, ASP.NET supports the session value which does not use the Cookie and in this case, session key value is included in request URI. The following is an example of URL including session key.

```
http://server/(session_key)/SessionState.aspx

ex) http://localhost/(lit3py55t21z5v55vlm25s55)/Application/
SessionState.aspx
```

If you use this URL as an application service name itself, the name increases comparing to the number of users. For example, if there is 1,000 of applications and 1,000 of users, application service name would be 1,000,000, in which java.lang.OutOfMemoryError problem might occur on the Jennifer Server. To solve this problem, you can extract the application service name based on the URL pattern using the uri_starter option of the JENNIFER agent. Set this option, then you can see the example.

```
uri_starter = )/
```

Application service name is as follows:

```
)/Application/SessionState.aspx
```

6.3. Application Service Monitoring

While monitoring the Java application, JENNIFER collects the performance data using two approaches. The first is transaction performance monitoring, and the second is service performance monitoring.

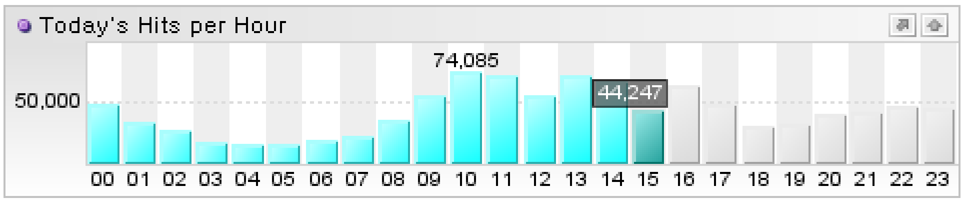
Transaction performance monitoring monitors the processing status of each transaction requested by the client, and it can be done in the X-view. For more details, please refer to [X-view Screen Analysis].

Service performance monitoring monitors the transactions of each application service that are executed in every 10-minute interval. For more details, please refer to [Real-Time Monitoring- Application].

6.3.1. Hits

The number of hits refers to the total number of transactions processed by the Java application. Hits are general performance data, and the number of hits in each 10-minute interval are stored in the HIT column of the PERF_X_01~31 table, and the trend in hits on a certain day can be viewed in the form of a bar chart.

Figure 6-1:Today's Hits per Hour



Notice: The number of hits per transaction and the average response time are referred to as the application processing statistical data. For further details, please refer to [Performance Status and Report]. Detailed analysis of individual transactions is done in the X-view. For further details, please refer to [X-view and Profiling]..

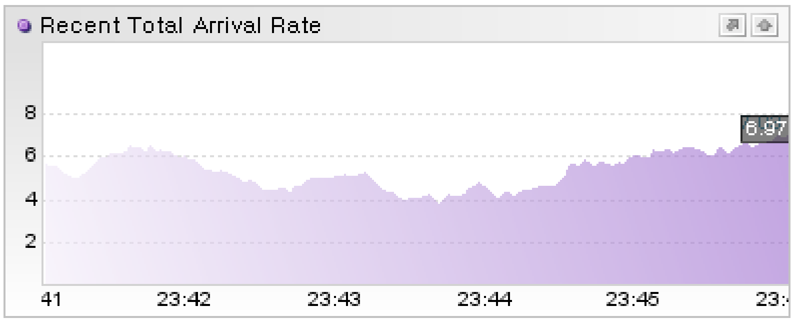
6.3.2. Arrival Rates and Service Rates

An arrival rate refers to the number of transactions initiated by the Java application per minute, while service rate refers to the number of transactions terminated by the Java application per minute.

In most cases, the arrival rate and the service rate are approximately the same. However, if the number of hits is large or if the performance is poor, they can differ from each other significantly.

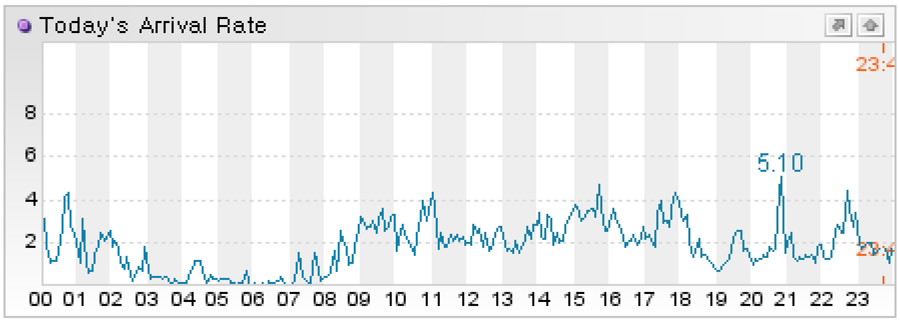
The arrival rate and the service rate are general performance data collected by the JENNIFER agent. Their average values over each 30-second segment can be viewed on the speed bar or speedmeter, or in the form of a runtime line chart.

Figure 6-2:Recent Arrival Rate



In addition, the five-minute averages of arrival rates and service rates are stored in the ARRIVAL_RATE and SERVICE_RATE columns of the PERF_X_01~31 tables. The service or arrival rate trend on a certain day can be viewed in the form of a line chart.

Figure 6-3:Today's Arrival Rate

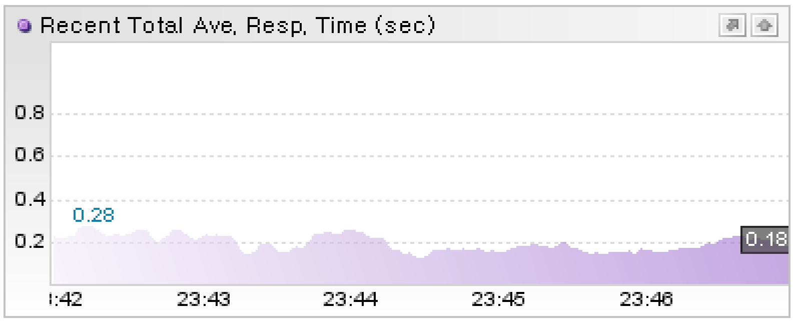


6.3.3. Average Response Time

The average response time is obtained by dividing the total transaction response time by the number of hits. It is expressed in seconds.

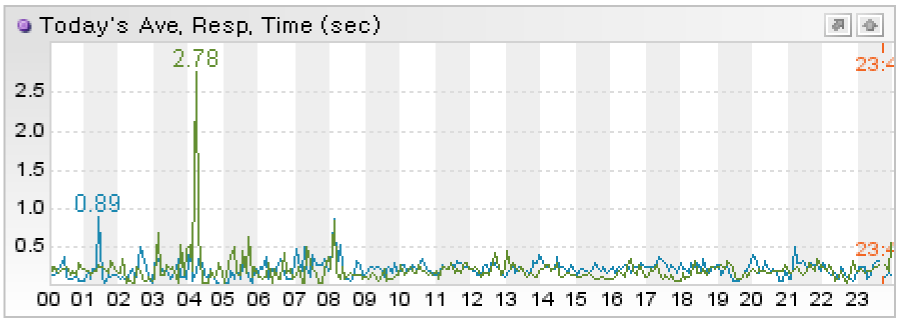
The average response time is general performance data collected by the JENNIFER agent. The 30-second average can be viewed in the form of a runtime line chart.

Figure 6-4:Recent Total Average Time



In addition, the five-minute average response time is saved in the RESPONSE_TIME column of the PERF_X_01~31 table, and the average response time trend for a specific day can be viewed on a line chart.

Figure 6-5:Today's Average Response Time



6.3.4. Transaction and Exception Handling

Exception monitoring and collection in the middle of the processing of the transaction is very critical to service monitoring. If all of the exceptions that occur while a transaction is being executed are monitored and tracked, it can negatively affect the application performance. Therefore, the JENNIFER agent only detects and processes essential exceptions.

The JENNIFER agent monitors and collects exceptions in the following cases.

- End of transactions. In other words, when an exception occurs in the method that is the application service start point; or, when an exception occurring in the middle of transaction processing is delivered to this method.
- An exception occurs in the method that is the external transaction start point; or, an exception occurring in the middle of external transaction processing is delivered to this method.

- An exception occurs while processing the JDBC.

If an exception occurring in the middle of transaction processing (excluding the three cases mentioned above) is handled by the catch text, JENNIFER does not detect it.

6.3.4.1. Transaction Success and Failure

If one of the following exceptions is detected while terminating the transaction, the transaction is deemed to be a failure.

- ERROR_RECURRSIVE_CALL
- ERROR_UNCAUGHT_EXCEPTION
- ERROR_HTTP_IO_EXCEPTION
- ERROR_OUTOFMEMORY
- ERROR_UNKNOWN_ERROR
- ERROR_PLC_REJECTED
- ERROR_JDBC_CONNECTION_FAIL

Notice: Even if an exception occurs due to a simple delay in the response time, the transaction is deemed to have been successful. In addition, even if an exception occurs while handling an external transaction or the JDBC, it is deemed to have been successful, as long as the catch text in the code processes it.

6.3.4.2. External Transactions and Exceptions

The following external transaction related exceptions are detected while processing a transaction.

- WARNING_TX_CALL_EXCEPTION
- WARNING_TX_BAD_RESPONSE

6.3.4.3. DB Handling and Exceptions

The following DB related exceptions are detected while processing a transaction.

- ERROR_DB_CONNECTION_FAIL
- WARNING_DB_TOOMANY_FETCH
- WARNING_JDBC_STMT_EXCEPTION
- WARNING_JDBC_PSTMT_EXCEPTION
- WARNING_JDBC_BAD_RESPONSE

- WARNING_DB_UN_COMMIT_ROLLBACK
- WARNING_DB_CONN_ILLEGAL_ACCESS

6.3.4.4. Arbitrary Exception Detection

Even if an exception occurring while processing the transaction is processed by the catch text, you can use the CustomTrace adaptor to detect it. For more details, refer to [CustomTraceAdaptor].

Even if you detect an exception with a CustomTrace adaptor, the transaction is deemed to have been successful.

6.3.5. Service Monitoring and Exceptions

6.3.5.1. Delayed Response Time

If the transaction response time exceeds the threshold, then the WARNING_APP_BAD_RESPONSE exception will occur. The threshold is set by the app_bad_responsetime of the JENNIFER agent. The default is 30000, and it is expressed in milliseconds.

```
app_bad_responsetime = 30000
```

6.3.5.2. Detecting Infinite Recursive Calls to the Service

If the Java application's sublet or JSP exceeds the threshold (default of 50,000) and it continues to call other sublets or JSP, then the ERROR_RECURRSIVE_CALL exception will occur. You can set the threshold using the recurrsive_call_max_count option of the JENNIFER agent.

```
recurrsive_call_max_count = 50000
```

If a sublet or JSP is calling other sublets or JSP, this means that the forward or include method of the javax.servelt.RequestDispatcher object in the Java code has been called.

To understand the cause of this exception and record the stacktrace in the JENNIFER agent log file, set the recurrsive_call_trace option of the JENNIFER agent to true. The default is false.

```
recurrsive_call_trace = true
```

Essentially, the number of characters in the Stacktrace recorded in the JENNIFER agent log file is limited to 10,000. You can set the number of letters using the recurrsive_call_trace_size option of the JENNIFER agent.

```
recurrsive_call_trace_size = 10000
```

Warning: When this exception occurs, the transaction is terminated. Therefore, if you assign values that are too small to the recurrsive_call_max_count option, the service may not be normal.

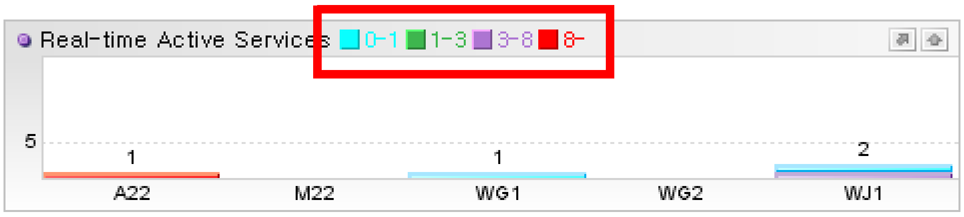
6.4. Active Service Monitoring

The active service refers to the transaction being currently processed by the Java application, which is also the transaction that the method that is the application start point is currently executing.

6.4.1. Number of Active Services

The number of active services is the number of transactions that the Java application is currently processing.The number of active services is general performance data collected by the JENNIFER agent. The current value can be displayed in the form of an equalizer and the thirty-second average value can be viewed in the form of an runtime line chart.

Figure 6-6:Real-time Active Services



However, the number of active services is partitioned into segments of elapsed time. The equalizer chart displays the value for each segment, differentiating the segments by color. The following table describes the segments of elapsed time.

Table 6-1: Elapsed Time Segments

Elapsed time	Description
0-1	For elapsed time of less than one second
1-3	For elapsed time between one and three seconds
3-8	For elapsed time between three and eight seconds
8-	For elapsed time of above eight seconds

These segments can be set using the active_graph_interval option of the JENNIFER server and agent. It is expressed in milliseconds.

```
active_graph_interval = 0,1000,3000,8000
```

Notice: If you want to change this option, change the option in both the JENNIFER server and the JENNIFER agent.

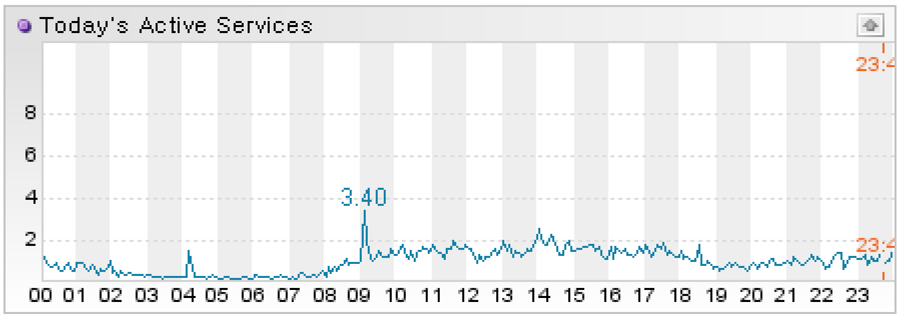
If an active service is delayed for more than 8 sec, it is displayed in the lower part of an equalizer as a default value. If you want to display it on the upper part of an equalizer, set the ui_active_service_reverse option as true.

```
ui_active_service_reverse = true
```

If you have modified this option, you have to login again.

In addition, the five-minute averages of active services are saved in the ACTIVE_SEVICE column of the PERF_X_01~31 table, and the active services trend on a certain day can be viewed in the form of a line chart.

Figure 6-7:Today's Active Services



6.4.2. Active Service Monitoring and Alerts

When the number of active services for all Java applications monitored by the JENNIFER server exceeds the threshold (default: 70) within five seconds, then the ERROR_SERVICE_QUEUING alert is issued. The threshold can be set using the active_service_alert_limit option of the JENNIFER server.

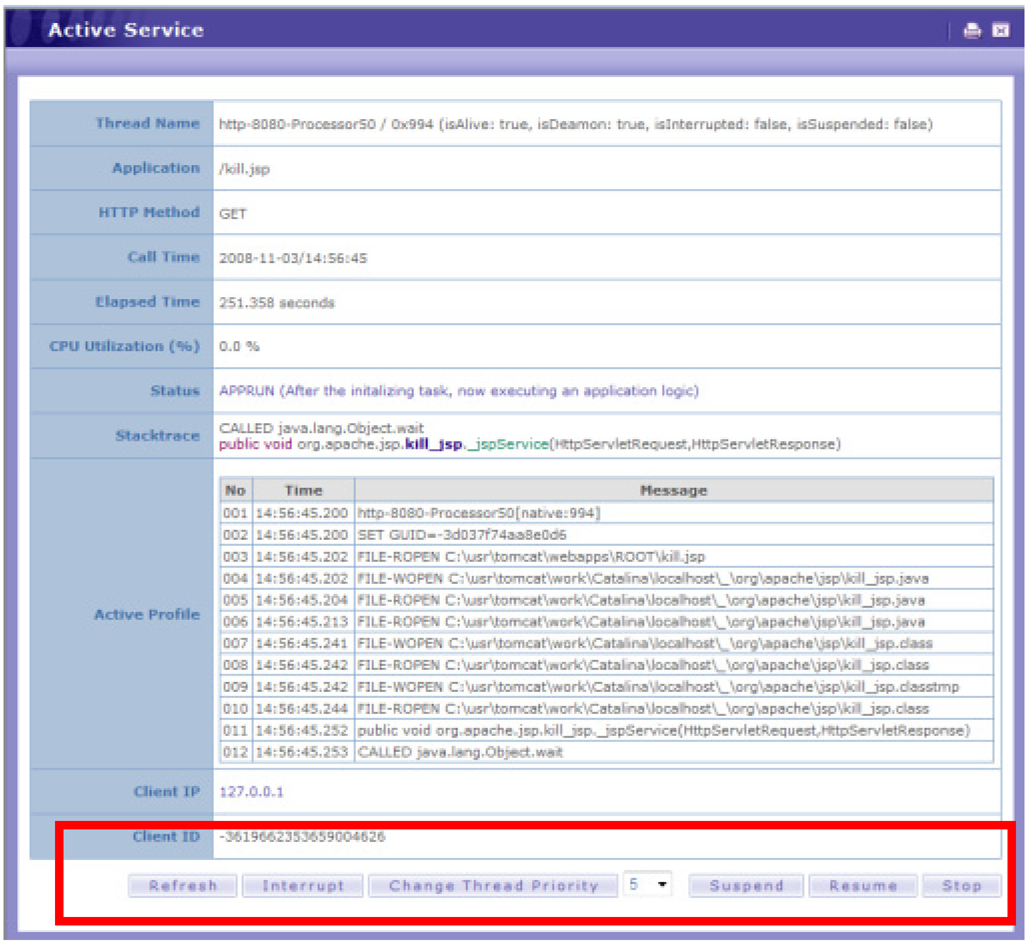
```
active_service_alert_limit = 70
```

6.4.3. Suspending Active Service

You can suspend the certain thread when the Java thread processing transaction is at a unstable status (wait or sleep, etc). The reason to suspend the idle transaction is to release the allocated resource that is occupied by the thread. That means you do not need to terminate the thread if there is enough resource.

If you click the application name having a long processing time on the active service list, you will see the following pop-up window.

Figure 6-8:Details of an Active Service



Thread Name	http-8080-Processor50 / 0x994 (isAlive: true, isDeamon: true, isInterrupted: false, isSuspended: false)
Application	/kill.jsp
HTTP Method	GET
Call Time	2008-11-03/14:56:45
Elapsed Time	251.358 seconds
CPU Utilization (%)	0.0 %
Status	APPRUN (After the initalizing task, now executing an application logic)
Stacktrace	CALLED java.lang.Object.wait public void org.apache.jsp.**kill_jsp**._jspService(HttpServletRequest,HttpServletResponse)
Active Profile	
Client IP	127.0.0.1
Client ID	-3619662353659004626

Active Profile:

No	Time	Message
001	14:56:45.200	http-8080-Processor50[native:994]
002	14:56:45.200	SET GUID=-3d037f74aa8e0d6
003	14:56:45.202	FILE-ROPEN C:\usr\tomcat\webapps\ROOT\kill.jsp
004	14:56:45.202	FILE-WOPEN C:\usr\tomcat\work\Catalina\localhost_\org\apache\jsp\kill_jsp.java
005	14:56:45.204	FILE-ROPEN C:\usr\tomcat\work\Catalina\localhost_\org\apache\jsp\kill_jsp.java
006	14:56:45.213	FILE-ROPEN C:\usr\tomcat\work\Catalina\localhost_\org\apache\jsp\kill_jsp.java
007	14:56:45.241	FILE-WOPEN C:\usr\tomcat\work\Catalina\localhost_\org\apache\jsp\kill_jsp.class
008	14:56:45.242	FILE-ROPEN C:\usr\tomcat\work\Catalina\localhost_\org\apache\jsp\kill_jsp.class
009	14:56:45.242	FILE-WOPEN C:\usr\tomcat\work\Catalina\localhost_\org\apache\jsp\kill_jsp.classtmp
010	14:56:45.244	FILE-ROPEN C:\usr\tomcat\work\Catalina\localhost_\org\apache\jsp\kill_jsp.class
011	14:56:45.252	public void org.apache.jsp.kill_jsp._jspService(HttpServletRequest,HttpServletResponse)
012	14:56:45.253	CALLED java.lang.Object.wait

On this active service window, you can control the certain thread processing transaction. Simply say, you can suspend or stop a thread. The buttons appears only when a user group has killthread authority.

Notice: If the thread is an idle status on Socket Read, you can not suspend the thread before releasing the Socket Read.

Warning: This function can not work properly under the certain WAS. It is tested under WebLogic 8.x/9.x, WebSphere 4.x/5.x/6.x, Tomcat 5.x, etc.

6.4.3.1. Autokill for the Active Service Exceeding Elapsed Time

You can automatically kill the thread that exceeds threshold of elapsed time for a long time. To activate this fuction, set the enable_long_running_thread_auto_kill option of the JENNIFER agent into true.

```
enable_long_running_thread_auto_kill = true
```

Set the threshold by using the long_running_thread_auto_kill_timeout option of the JENNIFER agent. It is expressed in milliseconds.

```
long_running_thread_auto_kill_timeout = 300000
```

6.4.4. Active Parameter

When checking the active service list in the **[Real-time Monitoring | Application]** menu, you can add an HTTP parameter value to the application service name.

Notice: You are only adding an HTTP value to the application service name that is displayed on the screen. You are not modifying the name itself.

To achieve this, set the parameter name to be added to the application service name in the http_post_request_tracking_list option of the JENNIFER agent.

```
http_post_request_tracking_list = txid
```

You can add the first java.lang.String type of parameter of the method of the class called while processing the transaction to the application service name.To set the modified active_param option, you have to restart Java application that installed the JENNIFER agent.

Essentially, add the first java.lang.String type of parameter to the application service name. To do this, set the active_param_type option of the JENNIFER agent to byte[]. The default is java.lang.String.

```
active_param_type = byte[]
```

For example, if you want to add parameter values of all methods of the example.BizAction class to the application service names, set the active_param_class option of the JENNIFER agent.

```
active_param_class = example.BizAction
```

Iif you want to add the parameter values of all methods of the example.LogAction class to the application service names, enter multiple entities separated by semicolons [;] in the active_param_class option. In addition to the active_param_class option, if you want to enter multiple entities in the option related to the application service names, use semicolons [;] as separators.

```
active_param_class = example.BizAction;example.LogAction
```

If you want to add the parameter of a certain method to the application service name, set the active_param_target_method option of the JENNIFER agent.

```
active_param_target_method = process
```

However, if some of the classes that are to be added to the application service names share the same methods, and you want to add the parameter value of a certain method of the class to the application service name, set it as follows:

```
active_param_target_method = example.BizAction.process
```

On the other hand, if you want to exclude the parameter values of certain methods from the application service name, set the active_param_ignore_method option of the JENNIFER agent.

```
active_param_ignore_method = example.BizAction.someMethod
```

Using an accessor of the method, you can set the parameter value of the method to be added to the application service name. For example, if you want to add the parameter value of the public accessor and the method without an accessor to the application service names, set it as follows:

```
active_param_access_method = public;none
```

The following parameters can be set using the active_param_access_method option.

- public - public accessor
- protected - protected accessor
- private - private accessor
- none - no accessor

If you want to add the parameter values of all classes inheriting from a certain class to the application service name, use the active_param_super option of the JENNIFER agent. For example, if you want to add the parameter values of all classes inheriting from example.BaseAction to the application service names, set it as follows:

```
active_param_super = example.BaseAction
```

However, in this case, only the parameter values of the classes directly inheriting from example.BaseAction are added to the application service names. For example, if Class A inherits from the example.BaseAction class and class B inherits from class A, then the parameter value of class A is added to the application service name, but the parameter value of class B is not.

If you want to add the parameter values of all classes implementing a certain interface to the application service name, use the tx_naming_interface option of the JENNIFER agent. For example, if you want to add the parameter values of all the classes implementing the example.IAction interface to the application service names, set it as follows:

```
active_param_interface = example.IAction
```

However, in this case only the parameter values of the classes directly implementing the example.IAction interface will be added to the application service names. For example, if class A implements the example.IAction interfaces and class B inherits from class A, then the parameter value of class A is added to the application service names, but the parameter value of class B is not.

Using the class names, you can set the parameter values of the class to be added to the application service names. In this case, use the tx_naming_prefix, tx_naming_postfix and tx_naming_ignore_prefix options of the JENNIFER agents. For example, if you want to add the parameter values of all classes whose names begin with example.action to the application service names, set it as follows:

```
active_param_prefix = example.action
```

In addition, if you want to add the parameter values of all classes whose names end with Action to the application service names, set it as follows

```
active_param_postfix = Action
```

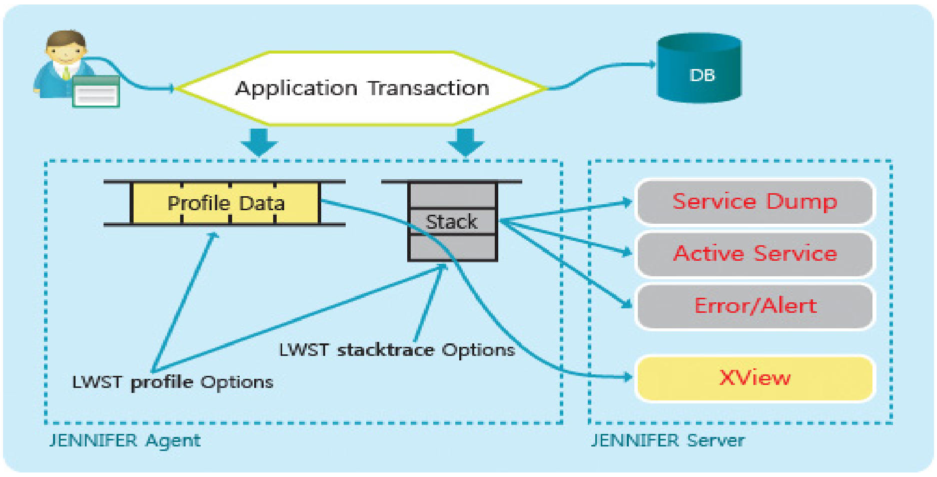
If you want to exclude the parameter values of the classes with certain names from the application service names, use the tx_naming_ignore_prefix option of the JENNIFER agent. This option overrides all other options.

```
active_param_ignore_prefix =
```

6.4.5. Active Stacktrace

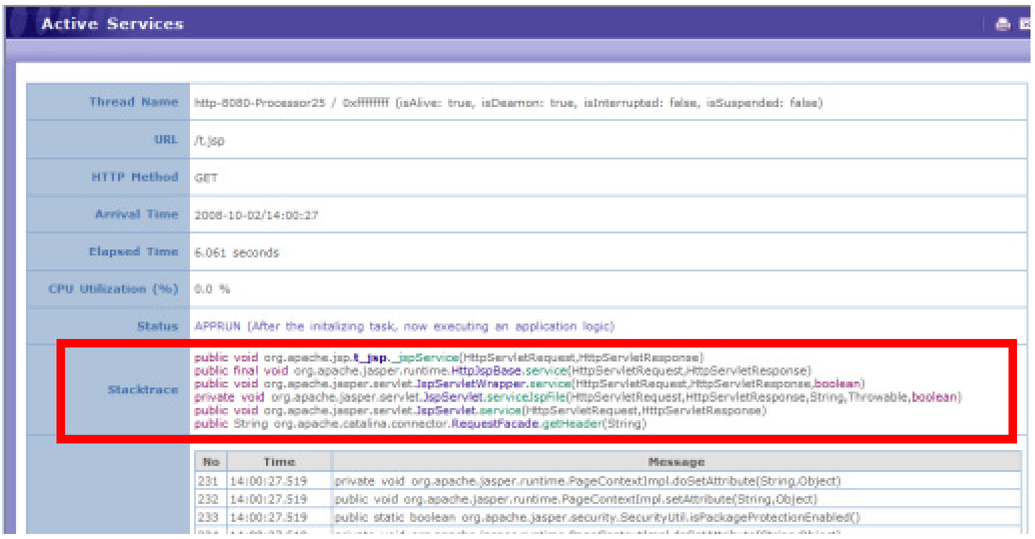
Active Stacktrace extracts the Stacktrace from the active service. This is similar to the X-View profile data. Active stacktrace is volatile data, and it disappears when the transaction is terminated, whereas the X-View profile data is saved permanently and is searchable after the transaction has been terminated.

Figure 6-9:Active Stacktrace and X-View Profile Data



As shown in the figure below, active stacktrace can be viewed in a pop-up window that displays the detailed information of the active service.

Figure 6-10:Active Stacktrace



Thread Name	http-8080-Processor25 / 0xffffffff (isAlive: true, isDeamon: true, isInterrupted: false, isSuspended: false)
URL	/t.jsp
HTTP Method	GET
Arrival Time	2008-10-02/14:00:27
Elapsed Time	6.061 seconds
CPU Utilization (%)	0.0 %
Status	APPRUN (After the initalizing task, now executing an application logic)
Stacktrace	public void org.apache.jsp.t_jsp._jspService(HttpServletRequest,HttpServletResponse) public final void org.apache.jasper.runtime.HttpJspBase.service(HttpServletRequest,HttpServletResponse) public void org.apache.jasper.servlet.JspServletWrapper.service(HttpServletRequest,HttpServletResponse,boolean) private void org.apache.jasper.servlet.JspServlet.serviceJspFile(HttpServletRequest,HttpServletResponse,String,Throwable,boolean) public void org.apache.jasper.servlet.JspServlet.service(HttpServletRequest,HttpServletResponse) public String org.apache.catalina.connector.RequestFacade.getHeader(String)

No	Time	Message
231	14:00:27.519	private void org.apache.jasper.runtime.PageContextImpl.doSetAttribute(String,Object)
232	14:00:27.519	public void org.apache.jasper.runtime.PageContextImpl.setAttribute(String,Object)
233	14:00:27.519	public static boolean org.apache.jasper.security.SecurityUtil.isPackageProtectionEnabled()

However, not all methods called while processing the transaction are displayed in stack-trace. Only the methods of the class set using the stacktrace option of the JENNIFER agent are included in stacktrace. To set the modified stacktrace option, you have to restart Java application that installed the JENNIFER agent.

For example, if you want to include all methods of the example.BizAction class in stack-trace, set the stacktrace_class option of the JENNIFER agent.

```
stacktrace_class = example.BizAction
```

If you want to include all methods of the example.LogAction class in stacktrace, set them in the stacktrace_class options, using semicolons as separators. In addition to the stacktrace_class option, if you want to set multiple entities in the options related to stack-trace, use semicolons [;] as separators.

```
stacktrace_class = example.BizAction;example.LogAction
```

Using an accessor of the method, you can set the method to be added to the stacktrace. For example, if you want to include the public accessor and the method without an accessor to stacktrace, set it as follows.

```
stacktrace_access_method = public;none
```

The following parameters can be set using the stacktrace_access_method option.

- public - public accessor
- protected - protected accessor
- private - private accessor
- none - no accessor

If you want to add the methods of all classes inheriting from a certain class to stacktrace, use the stacktrace_super option of the JENNIFER agent. For example, if you want to add the methods of all the classes inheriting from example.BaseAction to stacktrace, set it as follows:

```
stacktrace_super = example.BaseAction
```

However, in this case only the methods of the classes directly inheriting from example.BaseAction are added to stacktrace. For example, if Class A inherits from the example.BaseAction class and class B inherits from class A, then the method of class A is added to the application service name, but the method of class B is not.

If you want to add the methods of all classes implementing a certain interface to the stacktrace, use the stacktrace_interface option of the JENNIFER agent. For example, if you want to add the methods of all classes implementing the example.IAction interface to stacktrace, set it as follows:

```
stacktrace_interface = example.IAction
```

However, in this case, only the methods of the classes directly implementing the example.IAction interface will be added to stacktrace. For example, if class A implements the example.IAction interface and class B inherits from class A, then the method of class A is added to stacktrace, but the method of class B is not.

Using the class names, you can set the class to be added to stacktrace. In this case, use the stacktrace_prefix, and the stacktrace_postfix options of the JENNIFER agents. For example, if you want to add all the methods of classes whose names begin with stacktrace_interface = example.IAction to stacktrace, set it as follows:.

```
stacktrace_interface = example.IAction
```

In addition, if you want to add the methods of all classes whose names end with Action to stacktrace, set it as follows:

```
stacktrace_postfix = Action
```

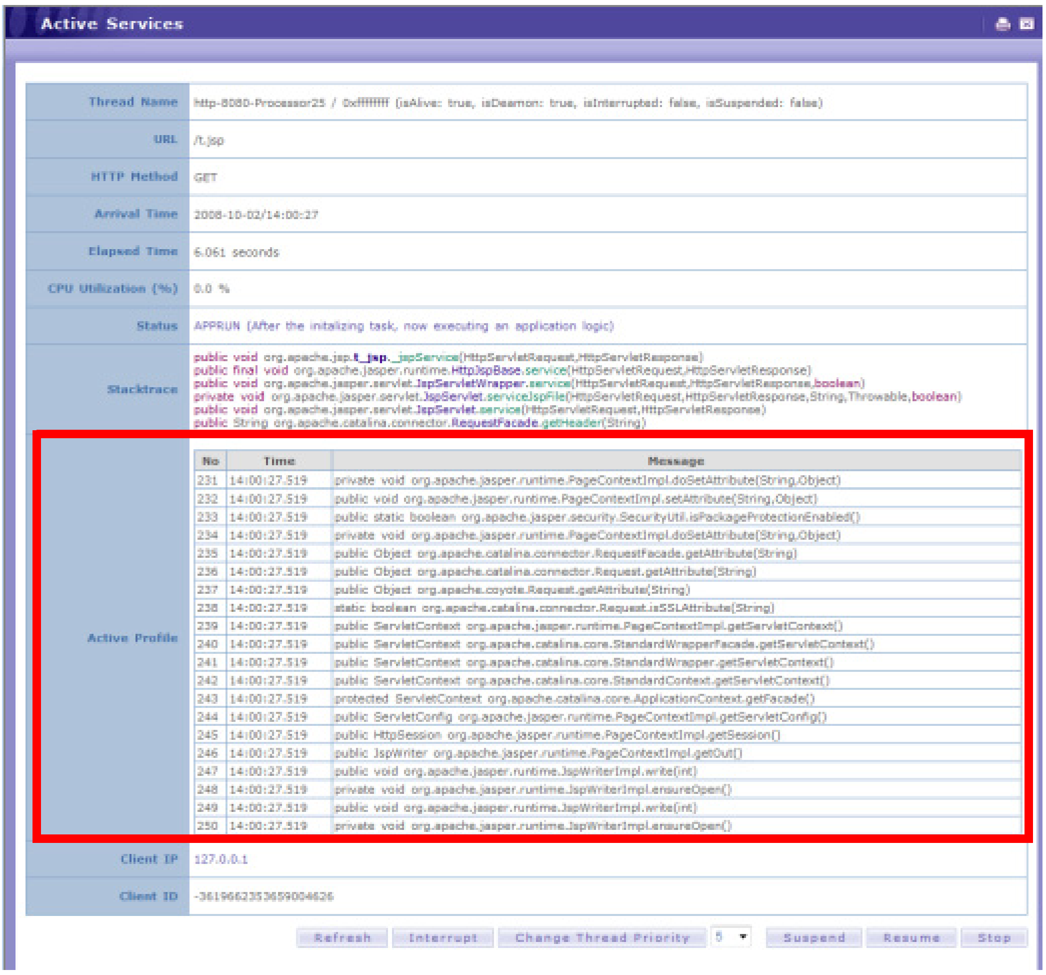
Finally, using the stacktrace_stacksize option of the JENNIFER agent, you can set the number of methods to be added to stacktrace. The default is 300.

```
stacktrace_stacksize = 300
```

6.4.6. Active Profile

As shown below, the active X-view profile data of the active service can be viewed in a pop-up window that shows the detailed information of the active service.

Figure 6-11:Active Profile



Thread Name	http-8080-Processor25 / 0xffffffff (isAlive: true, isDeamon: true, isInterrupted: false, isSuspended: false)
URL	/t.jsp
HTTP Method	GET
Arrival Time	2008-10-02/14:00:27
Elapsed Time	6.061 seconds
CPU Utilization (%)	0.0 %
Status	APPRUN (After the intializing task, now executing an application logic)
Stacktrace	public void org.apache.jsp.t_jsp._jspService(HttpServletRequest,HttpServletResponse) public final void org.apache.jasper.runtime.HttpJspBase.service(HttpServletRequest,HttpServletResponse) public void org.apache.jasper.servlet.JspServletWrapper.service(HttpServletRequest,HttpServletResponse,boolean) private void org.apache.jasper.servlet.JspServlet.serviceJspFile(HttpServletRequest,HttpServletResponse,String,Throwable,boolean) public void org.apache.jasper.servlet.JspServlet.service(HttpServletRequest,HttpServletResponse) public String org.apache.catalina.connector.RequestFacade.getHeader(String)
Active Profile	(see below)
Client IP	127.0.0.1
Client ID	-3619662353659004626

Active Profile:

No	Time	Message
231	14:00:27.519	private void org.apache.jasper.runtime.PageContextImpl.doSetAttribute(String,Object)
232	14:00:27.519	public void org.apache.jasper.runtime.PageContextImpl.setAttribute(String,Object)
233	14:00:27.519	public static boolean org.apache.jasper.security.SecurityUtil.isPackageProtectionEnabled()
234	14:00:27.519	private void org.apache.jasper.runtime.PageContextImpl.doSetAttribute(String,Object)
235	14:00:27.519	public Object org.apache.catalina.connector.RequestFacade.getAttribute(String)
236	14:00:27.519	public Object org.apache.catalina.connector.Request.getAttribute(String)
237	14:00:27.519	public Object org.apache.coyote.Request.getAttribute(String)
238	14:00:27.519	static boolean org.apache.catalina.connector.Request.isSSLAttribute(String)
239	14:00:27.519	public ServletContext org.apache.jasper.runtime.PageContextImpl.getServletContext()
240	14:00:27.519	public ServletContext org.apache.catalina.core.StandardWrapperFacade.getServletContext()
241	14:00:27.519	public ServletContext org.apache.catalina.core.StandardWrapper.getServletContext()
242	14:00:27.519	public ServletContext org.apache.catalina.core.StandardContext.getServletContext()
243	14:00:27.519	protected ServletContext org.apache.catalina.core.ApplicationContext.getFacade()
244	14:00:27.519	public ServletConfig org.apache.jasper.runtime.PageContextImpl.getServletConfig()
245	14:00:27.519	public HttpSession org.apache.jasper.runtime.PageContextImpl.getSession()
246	14:00:27.519	public JspWriter org.apache.jasper.runtime.PageContextImpl.getOut()
247	14:00:27.519	public void org.apache.jasper.runtime.JspWriterImpl.write(int)
248	14:00:27.519	private void org.apache.jasper.runtime.JspWriterImpl.ensureOpen()
249	14:00:27.519	public void org.apache.jasper.runtime.JspWriterImpl.write(int)
250	14:00:27.519	private void org.apache.jasper.runtime.JspWriterImpl.ensureOpen()

The number of profile items to be displayed on the screen for the X-View profile data can be set using the active_profile_max_line option of the JENNIFER server.

```
active_profile_max_line = 50
```

6.5. Peak Load Control(PLC)

PLC (Peak Load Control) is a function used to control the traffic congestion by controlling the number of active services, the major business groups and the performance degradation group.

6.5.1. Basic PLC Configuration

First of all, if you want to use PLC, set the set_limit_active_service option of the JENNIFER agent to true.

```
set_limit_active_service = true
```

When PLC is used, if the number of active services exceeds the threshold set in the max_num_of_active_service option of the JENNIFER agent, additional requests will be rejected to prevent service queuing.

```
max_num_of_active_service = 100
```

When a service request is rejected by PLC, the Java application users must be notified that additional requests cannot be processed because the load has exceeded the threshold. There are two ways in which the users can be notified, and these can be set using the request_reject_type option of the JENNIFER agent.

```
request_reject_type = [message|redirect]
```

If the request_reject_type option is set to message, then the text set in the request_reject_message option of the JENNIFER agent will be displayed to the user.

```
request_reject_message = Workload is too high! Please try again later.
```

If the request_reject_type option is set to redirect, the webpage designated using the request_reject_redirect_url option of the JENNIFER agent will be displayed to the user.

```
request_reject_redirect_url = /error.html
```

Warning: To prevent PLC from recursively rejecting the webpage designated using the request_reject_redirect_url option, you are recommended to use an HTML page in the request_reject_redirect_url option.

6.5.2. PLC Group Configuration

You can apply different PLC configurations to application groups that are classified for different levels of performance degradation and business priority.

6.5.2.1. Setting the Major Business Groups

When the load increases, you can redistribute the capabilities of the Java applications according to their order of importance. By setting the major business groups, you can increase the probability that the most important business will be processed when the load increases.

The major business groups are set in the following manner. First, using the biz_group.num option of the JENNIFER agent, you can set the number of active services belonging to all major business groups.

```
biz_group.num = 20
```

Notice: The max_num_of_active_service option defines the limit for the total number of active services, while the biz_group.num option defines the threshold for the number of active services belonging to the major business groups.

The major business groups are set using the biz_group.#.num and biz_group.#.list option of the JENNIFER agent. You must replace the “#” with the major business group number.

For example, if /biz1.jsp and /biz.2.jsp are grouped together as a major business group with 10 active services, while /biz3.jsp and /biz4.jsp are grouped together as a major business group with 15 active services, set it as follows:

```
biz_group.1.num = 10
biz_group.1.list = /biz1.jsp, /biz2.jsp
biz_group.2.num = 15
biz_group.2.list = /biz3.jsp, /biz4.jsp
```

Notice: You can set up to 20 major business groups.

The following PLC processing policy should apply to the major business groups.

- If the total number of active services does not exceed the threshold set in the max_num_of_active_service option, all requests are processed.
- Even if the total number of active services exceeds the threshold set in the max_num_of_active_service option, all requests pertaining to the major business group are processed, as long as the number of active services for the major business group does not exceed the threshold set in the biz_group.num option.
- Even if the total number of active services exceeds the threshold set by the max_num_of_active_service option and the number of entire active services for the major business group exceeds the threshold set by the biz_group.num option, all requests pertaining to the major business group are processed, as long as it does not exceed the threshold for the number of active services.

6.5.2.2. Setting the Performance Degradation Group

To limit the number of performance-degrading applications that can be executed, you can designate applications that degrade the performance in a performance degradation group, and establish a threshold for the number of active services belonging to the performance degradation group.

A performance degradation group can be set as follows. First, using the limit_group.num option of the JENNIFER agent, you can set the threshold for the number of active services belonging to performance degradation groups.

```
limit_group.num = 20
```

Notice: The max_num_of_active_service option defines the threshold for the total number of active services, while the limit_group.num option defines the threshold for the number of active services belonging to all performance degradation groups.

The major business groups are set using the limit_group.#.num and limit_group.#.list option of the JENNIFER agent. You must replace "#" with the performance degradation group number.

For example, if /bad1.jsp and /bad.2.jsp are grouped together as a performance degradation group for which 10 active services are allowed, while /bad3.jsp and /bad4.jsp are grouped together as a performance degradation group for which 15 active services are allowed, set it as follows:

```
limit_group.1.num = 10
limit_group.1.list = /bad1.jsp, /bad.jsp
limit_group.2.num = 15
limit_group.2.list = /bad.jsp, /bad.jsp
```

Notice: You can set up to 20 performance degradation groups.

The following PLC processing policy should apply to performance degradation groups.

- If the total number of active services exceeds the threshold set by the max_num_of_active_service option, all requests pertaining to the performance degradation groups are rejected.
- If the total number of active services for a performance degradation groups exceeds the threshold set by the limit_group.num option, all requests pertaining to the performance degradation group are rejected.
- If the number of active services for a certain performance degradation group set by the limit_group.#.list option exceeds the threshold for the number of active services for the performance group set in the limit_group.#.num option, the request is rejected.

6.5.2.3. Setting the Application List

The list of applications belonging to a major business group and/or a performance degradation group is set by entering the application names separated by commas [,] as shown below.

```
biz_group.1.list = /biz1.jsp, /biz2.jsp
```

And you can use [*] in application names.

Table 6-2: Setting the Application Names

Application name	Description
/biz*	All applications whose names begin with /biz belong to the group
*biz.jsp	All applications whose names end with biz.jsp belong to the group
biz	All applications containing biz in their names belong to the group.

Warning: You may not use an application name that has been set with the tx_naming option of the JENNIFER agent in the PLC application list.

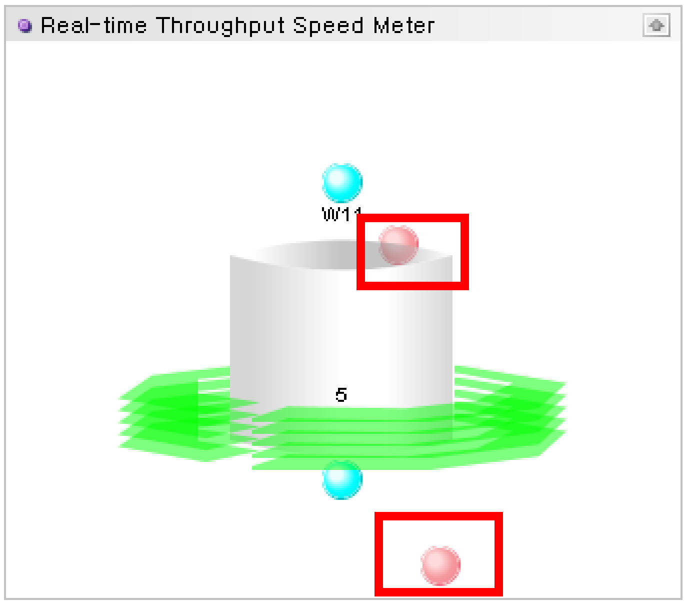
6.5.3. Checking PLC Operation

A user can check PLC operation through the JENNIFER server.

First, when a request is rejected by PLC, an ERROR_PLC_REJECTED exception occurs.

Second, you can check PLC operation using the speedmeter. In normal circumstances, a transaction must pass through the center of the JENNIFER agent column. However, when a request is rejected by PLC it will flow along the right side of the JENNIFER agent column.

Figure 6-12:Checking PLC with a Speedmeter



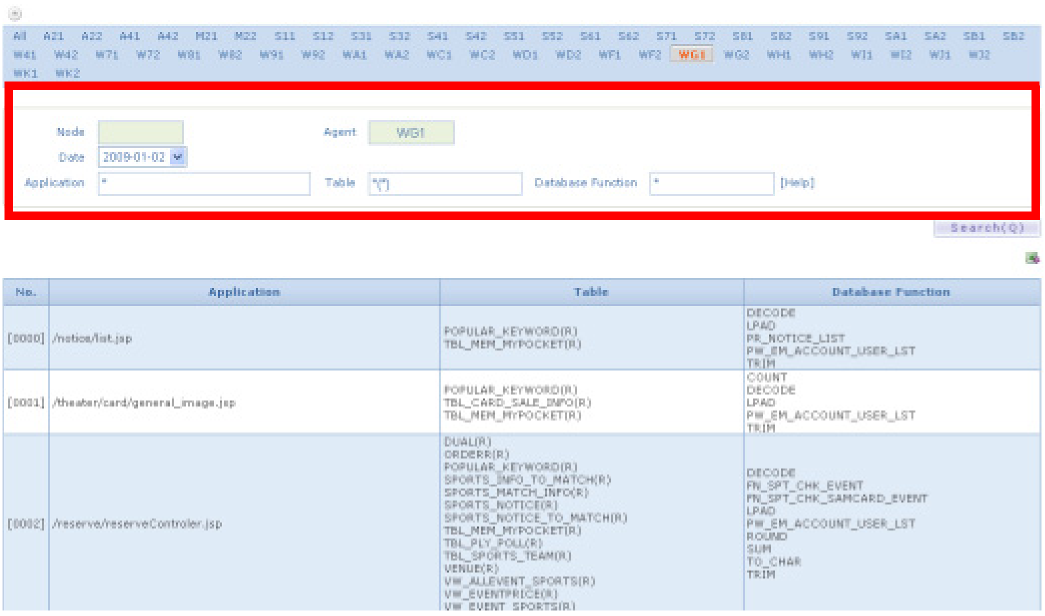
6.5.4. CRUD Matrix

In the **[Statistics | CRUD Matrix]** menu, you can check the relationship between the application services and the database tables or functions that have been created, read, deleted and updated.

This function can be used for comparative analysis between the CRUD matrix in actual operation and the CRUD matrix defined in the project analysis and design phase. Therefore, this function is very useful for managing application quality..

Notice: Searches in the CRUD matrix may take a very long period of time, as they involve a large amount of data.

Figure 6-13:CRUD Search Screen



All A21 A22 A41 A42 M21 M22 S11 S12 S31 S32 S41 S42 S51 S52 S61 S62 S71 S72 S81 S82 S91 S92 SA1 SA2 SB1 SB2 W41 W42 W71 W72 W81 W82 W91 W92 WA1 WA2 WC1 WC2 WD1 WD2 WF1 WF2 WG1 WG2 WH1 WH2 WI1 WI2 WJ1 WJ2 WK1 WK2

Node		Agent	WG1		
Date	2009-01-02				
Application	*	Table	*(*)	Database Function	* [Help]

Search(Q)

No.	Application	Table	Database Function

[0000]	/notice/list.jsp	POPULAR_KEYWORD(R)
		TBL_MEM_MYPOCKET(R)
			DECODE
			LPAD
			PR_NOTICE_LIST
			PW_EM_ACCOUNT_USER_LST
			TRIM

[0001]	/theater/card/general_image.jsp	POPULAR_KEYWORD(R)
		TBL_CARD_SALE_INFO(R)
		TBL_MEM_MYPOCKET(R)
			COUNT
			DECODE
			LPAD
			PW_EM_ACCOUNT_USER_LST
			TRIM

[0002]	/reserve/reserveControler.jsp	DUAL(R)
		ORDERR(R)
		POPULAR_KEYWORD(R)
		SPORTS_INFO_TO_MATCH(R)
		SPORTS_MATCH_INFO(R)
		SPORTS_NOTICE(R)
		SPORTS_NOTICE_TO_MATCH(R)
		TBL_MEM_MYPOCKET(R)
		TBL_PLY_POLL(R)
		TBL_SPORTS_TEAM(R)
		VENUE(R)
		VW_ALLEVENT_SPORTS(R)
		VW_EVENTPRICE(R)
		VW_EVENT_SPORTS(R)
			DECODE
			FN_SPT_CHK_EVENT
			FN_SPT_CHK_SAMCARD_EVENT
			LPAD
			PW_EM_ACCOUNT_USER_LST
			ROUND
			SUM
			TO_CHAR
			TRIM

- Node- If you use node, select the certain node in the JENNIFER agent list and search the CRUD matrix in the node.
- Agent- Select the JENNIFER agent on the list to read CRUD matrix. If you want to see the every report on each JENNIFER agent, click [All].
- Date - You can select the date for searching in the CRUD matrix.
- Application - You can only search for specific application service names in the CRUD matrix.
- Table - You can search for specific tables in the CRUD matrix. The acronym CRUD displayed next to the table name, clearly illustrates the nature of database tasks. C stands for newly created, R for read, U for changed and D for delete. For example, if you want to search for read and changed elements in the element table, enter ELEMENT(RU).
- Database functions - You can search for specific database functions in the CRUD matrix.

The following search conditions can be entered in the table input field.

- * - Searches for all tables.
- *BIZ - Searches for tables whose names end with BIZ.
- *BIZ* - Searches for tables whose names contain BIZ.
- BIZ* - Searches for tables whose names begin with BIZ.

In addition to the table names, you can perform searches based on the CRUD types.

- (*) - All CRUD types are searched.
- *(R) - The R types are searched. This searching condition includes CR, CRUD, RU and RD.
- *(C|R) - The C or R types are searched. You may omit | and enter *(CR) instead.
- *(C&U) - The types including both C and U are searched.
- *(C&!R&!U&!D) - Only the C types are searched.

You can use both the table names and the CRUD type for your search.

- BIZ*(CUD) - Searches for C or R types among the tables whose names begin with BIZ.
- *BIZ(D) - Searches for D types among the tables whose names end with BIZ.
- *BIZ*(R&!C&!U&!D) - Only searches for R types among the tables whose names contain BIZ.

However, if the disable_app_mapping_db option of the JENNIFER server is set to true, you cannot search for the CRUD matrix.

```
disable_app_mapping_db = true
```

6.6. Service Dump

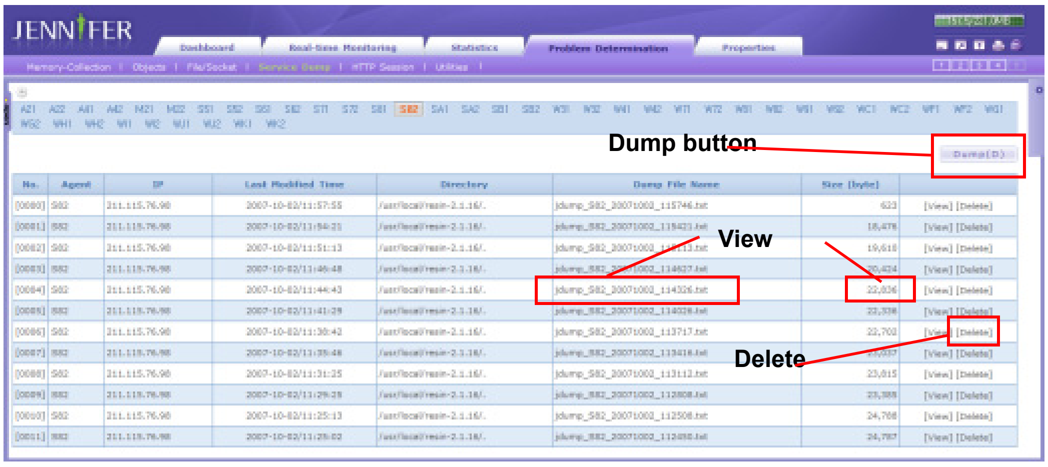
You can record the real-time performance data and the active service list for the Java application in a file. This is called a service dump.

By selecting the **[Problem Determination | Service Dump]** menu, you can record the service dump for a certain JENNIFER agent.

The directory to which the service dump file is saved can be set by the jdump_dir option of the JENNIFER agent. The default directory is the working directory.

```
jdump_dir = ./
```

Figure 6-14:Service Dump



No.	Agent	IP	Last Modified Time	Directory	Dump File Name	Size (byte)	
[0000]	S82	211.115.76.98	2007-10-02/11:57:55	/usr/local/resin-2.1.16/.	jdump_S82_20071002_115746.txt	623	[View] [Delete]
[0001]	S82	211.115.76.98	2007-10-02/11:54:21	/usr/local/resin-2.1.16/.	jdump_S82_20071002_115421.txt	18,478	[View] [Delete]
[0002]	S82	211.115.76.98	2007-10-02/11:51:13	/usr/local/resin-2.1.16/.	jdump_S82_20071002_115113.txt	19,618	[View] [Delete]
[0003]	S82	211.115.76.98	2007-10-02/11:46:48	/usr/local/resin-2.1.16/.	jdump_S82_20071002_114627.txt	20,424	[View] [Delete]
[0004]	S82	211.115.76.98	2007-10-02/11:44:43	/usr/local/resin-2.1.16/.	jdump_S82_20071002_114326.txt	22,836	[View] [Delete]
[0005]	S82	211.115.76.98	2007-10-02/11:41:29	/usr/local/resin-2.1.16/.	jdump_S82_20071002_114028.txt	22,338	[View] [Delete]
[0006]	S82	211.115.76.98	2007-10-02/11:38:42	/usr/local/resin-2.1.16/.	jdump_S82_20071002_113717.txt	22,702	[View] [Delete]
[0007]	S82	211.115.76.98	2007-10-02/11:35:48	/usr/local/resin-2.1.16/.	jdump_S82_20071002_113418.txt	[illegible]	[View] [Delete]
[0008]	S82	211.115.76.98	2007-10-02/11:31:25	/usr/local/resin-2.1.16/.	jdump_S82_20071002_113112.txt	23,815	[View] [Delete]
[0009]	S82	211.115.76.98	2007-10-02/11:29:25	/usr/local/resin-2.1.16/.	jdump_S82_20071002_112808.txt	23,388	[View] [Delete]
[0010]	S82	211.115.76.98	2007-10-02/11:25:13	/usr/local/resin-2.1.16/.	jdump_S82_20071002_112508.txt	24,768	[View] [Delete]
[0011]	S82	211.115.76.98	2007-10-02/11:25:02	/usr/local/resin-2.1.16/.	jdump_S82_20071002_112450.txt	24,787	[View] [Delete]

- Dump button - If you click on the button, you can save the main data of the JENNIFER agent currently selected in a service dump file.

Notice: Even if you click the [Dump] button, the new dump file may not appear in the service dump file list. This is because the JENNIFER agent records the service dump files asynchronously. Therefore, you need to wait several seconds before you will be able to see the new dump file. Do not press the [Dump] button repeatedly if the file does not appear immediately.

- Delete - Delete the service dump file.
- View - If you click on a service dump file or the **[View]** link, you can view the content of the dump file in a pop-up window.

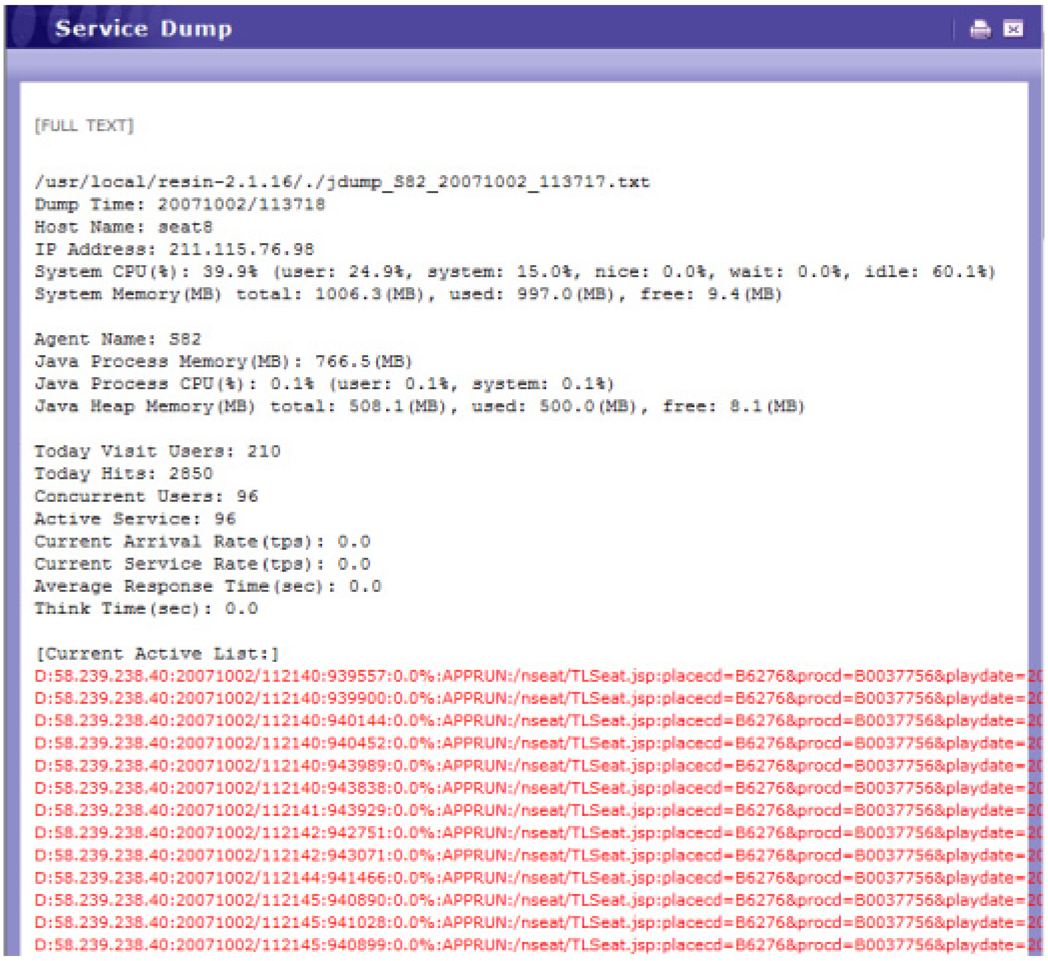
Notice: The service dump file is saved in the working directory of the JENNIFER agent, not the JENNIFER server.

6.6.1. Viewing the Content of the Service Dump File

A service dump file records the following:

- Time at which the service dump file is recorded
- Resource performance data at the dumping point: System CPU utilization, system memory utilization and Java heap memory utilization
- Service performance data at the dumping point: Active services, concurrent users and arrival rates
- Java environment variables

Figure 6-15:Contents of a Service Dump File



```
[FULL TEXT]

/usr/local/resin-2.1.16/./jdump_S82_20071002_113717.txt
Dump Time: 20071002/113718
Host Name: seat8
IP Address: 211.115.76.98
System CPU(%): 39.9% (user: 24.9%, system: 15.0%, nice: 0.0%, wait: 0.0%, idle: 60.1%)
System Memory(MB) total: 1006.3(MB), used: 997.0(MB), free: 9.4(MB)

Agent Name: S82
Java Process Memory(MB): 766.5(MB)
Java Process CPU(%): 0.1% (user: 0.1%, system: 0.1%)
Java Heap Memory(MB) total: 508.1(MB), used: 500.0(MB), free: 8.1(MB)

Today Visit Users: 210
Today Hits: 2850
Concurrent Users: 96
Active Service: 96
Current Arrival Rate(tps): 0.0
Current Service Rate(tps): 0.0
Average Response Time(sec): 0.0
Think Time(sec): 0.0

[Current Active List:]
D:58.239.238.40:20071002/112140:939557:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112140:939900:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112140:940144:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112140:940452:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112140:943989:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112140:943838:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112141:943929:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112142:942751:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112142:943071:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112144:941466:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112145:940890:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112145:941028:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
D:58.239.238.40:20071002/112145:940899:0.0%:APPRUN:/nseat/TLSeat.jsp:placecd=B6276&procd=B0037756&playdate=2
```

6.6.2. Automatic Service Dump Record

By adjusting the configuration, you can allow automatic service dump to begin when the number of active services exceeds the threshold. To accomplish this, set the enable_dump_triggering option of the JENNIFER agent to true.

```
enable_dump_triggering = true
```

The threshold for the number of active services can be set using the number_of_dump_trigger of the JENNIFER agent.

```
number_of_dump_trigger = 90
```

To prevent frequent service dumping, set the automatic service dump interval by using the dump_trigger_sleep_interval option of the JENNIFER agent. It is expressed in milliseconds.

```
dump_trigger_sleep_interval = 180000
```

6.7. Advanced Service Monitoring

Now, the service monitoring method in an SOA(Service Oriented Architecture) environment will be described.

6.7.1. Global Transaction Tracking

Essentially, JENNIFER tracks the transaction that is processed by a single Java thread. In other words, when a request is delivered to the Java application, a working thread is assigned to process the request, and this thread processes the transaction from the beginning to the end. However, in an SOA environment such as EAI (Enterprise Application Integration) or ESB (Enterprise Service Bus), one or more Java threads or multiple Java applications may process a single request by using the XML web service.

In other words, a single request can be processed by multiple transactions in one or more applications. For this reason, it is necessary to connect the transactions that process the same request for the purpose of service monitoring. To do this, JENNIFER uses GUID (Globally Unique Identifier).

To distinguish transactions, JENNIFER assigns a UUID (Universally Unique Identifier) to every transaction that is processed. However, to track the relationships between multiple transactions that process the same request, it assigns them the same GUID. This is referred to as global transaction tracking or GUID tracking.

For GUID tracking, set the trace_related_transaction option of the JENNIFER agent to true.

```
trace_related_transaction = true
```

The UUID is created by JENNIFER and assigned to a transaction, but the GUID is created and managed by an application, not JENNIFER. Therefore, JENNIFER should determine the GUID value and assign it to a transaction. JENNIFER provides various means of achieving this.

First, extract the GUID from the HTTP header information and assign it to the transaction. To do this, you must set the HTTP header key name that represents the GUID using the relatx_guid_keyname option of the JENNIFER agent..

```
relatx_guid_keyname = _J_GUID_
```

In the configuration shown above, the GUID and the UUID are assigned to the transaction and the preceding transaction while executing the following code:

```
String guid = request.getHeader("_J_GUID_");
```

In addition, you can use the parameter of an arbitrary method of the class called while processing the transaction as the GUID assigned to the transaction.

Using the guid_param option of the JENNIFER agent, you can set the method of the class from which the GUID is to be extracted. In this case, you need to describe the parameter of the method as well. The parameter can be set with a class name that does not include the package name. For example, if you want to use the parameter of the process method of the example.Action class as the GUID assigned to the transaction, set it as follows: To set the modified guid_param option, you have to restart Java application that installed the JENNIFER agent.:

```
guid_param = example.Action.process(String, String)
```

The parameter type should be java.lang.String or byte[].

Essentially, the first parameter is used as the GUID assigned to the transaction. Using the guid_param_index option of the JENNIFER agent, you can set the parameter to be used as the GUID. If it is set to 0, the first parameter will be used as the GUID, and if set to 1, the second parameter will be used.

```
guid_param_index = 1
```

Moreover, using the guid_return option of the JENNIFER agent, you can add the method return value to the transaction as a GUID. In this case, you need to describe the parameter of the method as well. he parameter can be set with a class name that does not include the package name. For example, if you want to add return value the parameter of the process method of the example.Action class as the GUID assigned to the transaction, set it as follows: To set the modified guid_return option, you have to restart Java application that installed the JENNIFER agent.

```
guid_return = example.Action.execute(String, String)
```

In addition, if no GUID exists, you can adjust the configuration to assign the UUID of the current transaction to the GUID. To do this, set the enable_guid_from_tuid option of the JENNIFER agent to true.

```
enable_guid_from_tuid = true
```

GUID-tracking related transactions can be monitored in the GUID view of the X-View. For more details, refer to [GUID], and for a more detailed analysis, refer to [X-view Data Transmission].

Notice: If you are unable to assign the GUID using the methods shown above, then you need to modify the application source code. For further details, refer to [ActiveTraceUtil].

6.7.2. HTTP Header and Parameter Logging

If you want to record the HTTP header and parameter value in the JENNIFER agent log file, set the dump_http_hide_all option of the JENNIFER agent into ‘ false’. Default is ‘true’.

```
dump_http_hide_all = true
```

You can also set the HTTP header and parameter name which is not to be recorded in the log filebecause of the security problem by setting the dump_http_hide_key option of the JENNIFER agent . Separate multiple values with a comma [,] as a delimiter. For instance, if you do not want to record the HTTP header value with the name of cookie, set the option as follows:

```
dump_http_hide_key = cookie
```

Notice: It is applied for the HTTP header and parameter simultaneously.

If you want to record the HTTP header for the certain URL in the JENNIFER agent log file, set the URL using dump_http_header_url_prefix option of the JENNIFER agent. If you set as below, JENNIFER records every log data for the URL. Separate multiple values with a comma [,] as a delimiter..

```
dump_http_header_url_prefix = /
```

For the parameter, use the dump_http_parameter_url_prefix option of the JENNIFER agent. Setting way is the same with the dump_http_header_url_prefix option.

6.8. User Monitoring

When monitoring a WAS-based web application, you can collect a variety of data, such as active users, concurrent users, visitors and thinktime. This activity is called user monitoring.

6.8.1. User Monitoring Using Cookies

In the client/server environments of the past, it was an acceptable practice to consider the number of TCP/IP connections as representative of the number of users. However, since the HTTP protocol in a web environment is connectionless, the number of TCP/IP connections is not necessarily the same as the number of users. To overcome this problem, JENNIFER performs user monitoring using cookies.

When a transaction begins, the JENNIFER agent checks to determine if the wmonid cookie exists. If it does not exist, then it assumes that the transaction is a request from a new user, increases the number of visitors accordingly, and creates an arbitrary value to be designated as a wmonid cookie.

This cookie is a permanent one. This means that even if the user restarts the web browser, the wmonid cookie is maintained, and the number of visitors is not increased. Of course, when calculating the number of visitors, we consider the time as well as the presence of the wmonid cookie. For more details, refer to [Visitors].

However, the cookie size is limited. This means that if the application is using too many cookies, then when the JENNIFER agent adds new wmonid cookies, the maximum permissible cookie size could be exceeded. In this case, set the hotfix_remote_address_for_wmonid option of the JENNIFER agent to true, so that user monitoring with cookies is deactivated.

```
hotfix_remote_address_for_wmonid = true
```

If the hotfix_remote_address_for_wmonid option is set to true, then the agent does not uses cookies to distinguish users, but uses client IP addresses instead. As a result, the accuracy of the count of the number of concurrent users and visitors is reduced.

In addition, there is an important aspect that should be considered when monitoring multiple web applications that provide services for omdividual domain names. For example, let us assume that we are monitoring two web applications, and each domain name is given as follows:

```
http://www.jennifersoft.com
http://sales.jennifersoft.com
```

When a user uses the web application provided by the domain name http://www.jennifer-soft.com for the first time, a new wmonid cookie is assigned. However, when this same user access the web application provided by the domain name http://sales.jennifer-soft.com, a new wmonid cookie is assigned, as the wmonid cookie assigned to the previous web application cannot be read. As a result, the same user may be counted twice.

In some situations, it is desirable to differentiate the users for each domain when calculating the total number of users. However, if you want to regard them as the same user, use the default_cookie_domain option of the JENNIFER agent.

```
default_cookie_domain = .jennifersoft.com
```

The default_cookie_domain option must start with [.]. When this option is used, the number of users accessing other IP addresses is not taken into account when calculating the number of concurrent users and visitors.

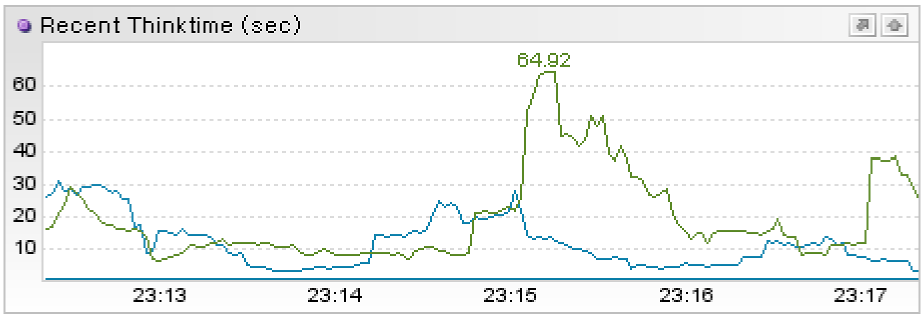
Notice: The content shown above is not related to concurrent users or visitors to a certain JENNIFER agent. It describes the total number of visitors and concurrent users for all JENNIFER agents.

6.8.2. Thinktime

Thinktime is the average time interval for calling the transactions with the same wmonid cookie.

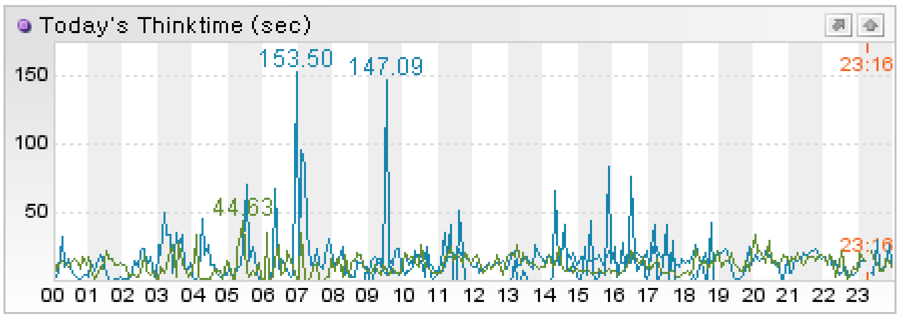
Thinktime is general performance data collected by the JENNIFER agent. The 30-second averages can be viewed on a runtime line chart.

Figure 6-16:Recent Thinktime



In addition, the five-minute averages for thinktime are stored in the THINKTIME column of the PERF_X_01~31 table. You can see the thinktime trend for a certain day in the form of a line chart.

Figure 6-17:Today's Thinktime



6.8.3. Active Users

Theoretically, as a result of the development of HTML frames and AJAX and the change in web browser interfaces, one or more active services can be connected to the same user.

In other words, transactions with the same wmonid cookie can exist among all active services. When calculating the number of active users, it is assumed that the transactions with the same wmonid cookie are only counted once.

The five-minute average of the number of active users is stored in the ACTIVE_USER column of the PERF_X_01~31 table.

Notice: The number of active users is not displayed in any chart.

6.8.4. Visitors

The number of visitors is collected from each time segment, and is dependent on the presence of the wmonid cookie. A time segment is further classified into date and time. The number of visitors on a certain date is referred to as daily visitors, while the number of visitors within a certain time period is referred to as hourly visitors.

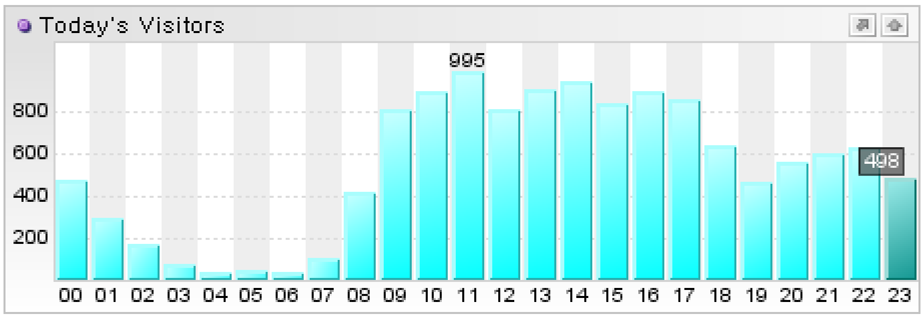
For example, if a transaction without a wmonid cookie is received at 01:25 on Oct 10th, the number of daily visitors is increased by one, while the number of hourly visitors for the hour-long segment starting at 01:00 is increased by one as well.

If a transaction with a wmonid cookie is received at 02:30 on Oct 10th, then the number of daily visitors is not increased, but the number of hourly visitors for the hour-long segment starting at 02:00 is increased by one. In other words, when the time is changed, the number of hourly visitors is increased, regardless of whether or not the wmonid cookie exists.

However, if a user who already visited on Oct 9th requests a transaction at 01:01 on Oct 10th (meaning a transaction that involves a wmonid cookie), the number of daily visitors on Oct 10th is increased by one. In other words, when the date changes, the number of daily visitors also changes, regardless of the existence of the same wmonid cookie.

The number of visitors is general performance data, and the change in the number of hourly visitors in five-minute segments is saved in the VISIT_HOUR column of the PERF_X_01~31 table, while the change in the number of daily visitors in five-minute intervals is saved in the VISIT_DAY column of the PERF_X_01~31 table. You can see the trend in the number of visitors on a certain date in a bar chart.

Figure 6-18:Today's Visitors



When calculating the number of visitors in consideration of time, consider whether or not the JENNIFER server has been restarted. The JENNIFER server stores the wmonid cookie value in the Java heap memory to calculate the number of visitors. However, when the JENNIFER server is restarted, all cookie values stored in the Java heap memory will disappear. Consequently, the accuracy of the number of visitors is reduced.

To address this issue, when restarting the JENNIFER server, you can specify whether or not to include the number of visitors collected before it was turned off in the calculation of the number of visitors after it is restarted. If you set the enable_visit_user_added_while_reloading option of the JENNIFER agent to true, then the previous value will be added, but if set to false, your calculation will start again.

```
enable_visit_user_added_while_reloading = true
```

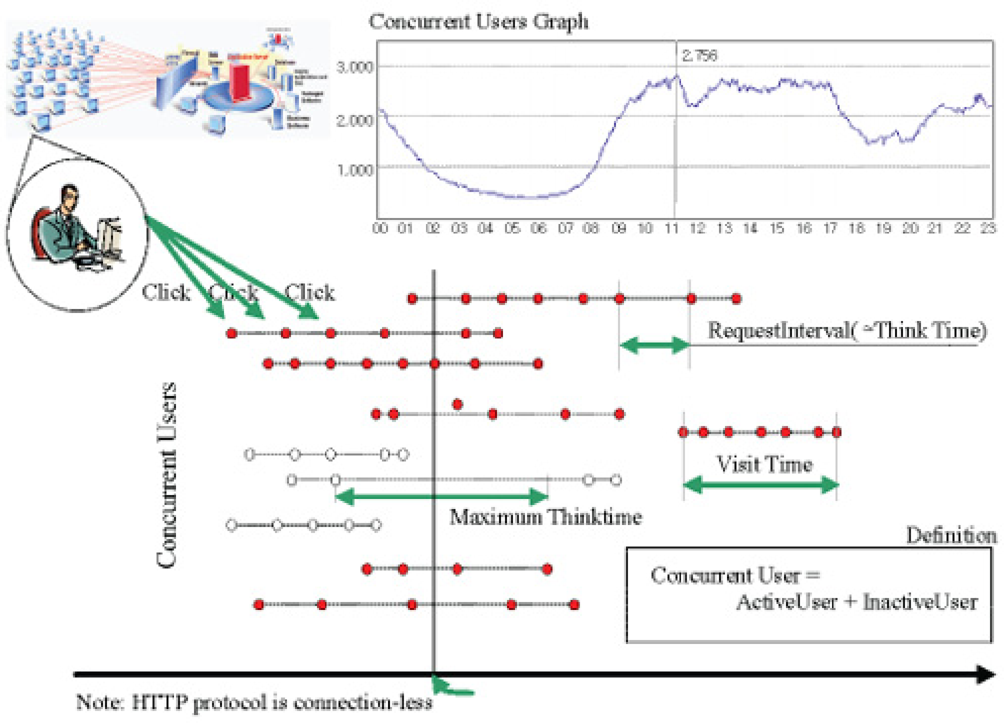
Notice: In general, for a system like an internet shopping mall, where users are changed frequently, set it to true. For a system with a fixed number of users, like an internal company system, set it to false. The default is false.

6.8.5. Concurrent Users

6.8.5.1. Understanding Concurrent Users

The following figure represents a user who accesses the web application and clicks on it for the first time, navigates to the next web page to click on it, and then finally leaves the web application. Other users also visit the web application at different times to use the service in this manner.

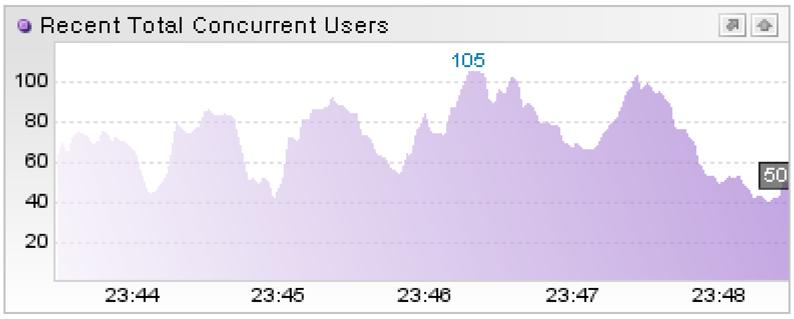
Figure 6-19:Understanding Concurrent Users



The time interval between each click is defined as thinktime and the time interval between the initial visit and the final visit is defined as visit time. On this plot, you can draw a line across a certain time period to check the number of accessors or concurrent users. There are a total of six concurrent users. In other words, the number of concurrent users refers to the number of users who access and use the web application at a certain time.

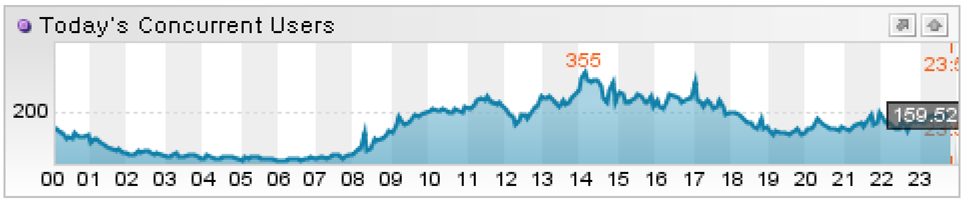
The number of concurrent users is general performance data collected by the JENNIFER agent. You can see the real-time 30-second average on a runtime line chart.

Figure 6-20:Recent Total Concurrent Users



In addition, the average number of concurrent users is saved in the CONCURRENT_USER in the PERF_X_01~31 table, and the trend in the number of concurrent users on a certain date can be viewed in the form of a line chart.

Figure 6-21:Today's Concurrent Users



6.8.5.2. Measuring the Number of Concurrent Users

The algorithm that JENNIFER uses to measure the number of concurrent users is based on performance theory, and it is a revolutionary method for accurately measuring the number of users of a web application. In client/server systems of the past, you could consider the number of TCP/IP connections to represent the number of concurrent users. However, as the HTTP protocol does not maintain the connection between the computer and the server at all times, but only makes a connection when it directly processes the user's request, the number of TCP/IP connections is not identical to the number of concurrent users. Therefore, we are obliged to use the following algorithm based on performance theory to accurately measure the number of concurrent users.

Figure 6-22:Algorithm for Measuring the Number of Concurrent Users

$$\text{Throughput(tps)} = \frac{\text{ActiveUser}}{\text{Resp.Time(sec)}} \quad \text{Little's Law}$$

$$\text{ActiveUser} = \text{ConcurrentUser} \times \frac{\text{Resp.Time(sec)}}{\text{Request Interval(=Resp.Time+ThinkTime)}}$$

$$\text{Throughput(tps)} = \frac{\text{ConcurrentUser}}{\text{Request Interval(=Resp.Time+ThinkTime)}}$$

$$\Downarrow$$

$$\text{ConcurrentUser} = \text{ActiveUser} + \text{Throughput(tps)} \times \text{ThinkTime(sec)}$$

$$\text{ConcurrentUser} = \text{ActiveUser} \times \{ 1 + \frac{\text{ThinkTime(sec)}}{\text{Resp.Time(sec)}} \}$$

$$\text{ConcurrentUser} = \text{Throughput(tps)} \times \{ \text{Resp.Time(sec)} + \text{ThinkTime(sec)} \}$$

It has been mathematically verified that the higher the throughput is, the more accurate the result is. After several dozen of performance test simulations and the monitoring of actual operating systems, the accuracy of this algorithm has been fully verified.

For example, if 500 users at a large hall access the same system, then the number of concurrent users measured by JENNIFER, or the number of the javax.servlet.http.HttpSession objects in the web application server is equal to 500. However, if everyone suddenly leaves the hall for lunch, JENNIFER's measurement method yields 0 immediately, but the number of the javax.servlet.http.HttpSession objects in the web application server is maintained at 500 during the session time-out.

Notice: Sometimes, the number of concurrent users will change dramatically. This can occur due to a technical difficulty in calculating the response time, thinktime and TPS. In this case, the results should be ignored, and the values derived during normal situations should be used.

6.8.5.3. Utilizing the Number of Concurrent Users

The number of concurrent users is very important data in terms of performance. However, it is sometime difficult to analyze whether a problem has occurred due to a technical difficulty or an increase in the number of concurrent users. Therefore, using the number of concurrent users as a means of analyzing the cause of problems is not recommended.

However, in the performance testing phase of web applications, the number of concurrent users plays a highly critical role in our understanding of the relationship between the performance indices of the performance testing environment and the actual operating environment.

In other words, a load test involves virtual users and designs of workload for each task, and the virtual user is directly related to the number of concurrent users measured by JENNIFER. You can examine whether or not the workload in the load test is similar to that in the actual operating environment.

6.8.6. Entire Performance Data and JENNIFER Agent Groups

Now, the method for calculating the total number of visitors or concurrent users when the JENNIFER server monitors two or more JENNIFER agents will be described.

For example, let's assume that the number of hourly visitors to the W11, and W12 and W13 JENNIFER agent between 9:00 and 10:00 are 120, 140 and 70, respectively.

The easiest way to calculate the total number of visitors is to sum up the number of visitors for all the JENNIFER agents. In this case, the result is 330.

However, if the same user visited all three JENNIFER agents between 9:00 and 10:00, then the result would then be greater than the actual number, because the same user is being counted more than once. To solve this problem, the JENNIFER server uses the wmonid cookies to prevent the same users from being counted repetitively.

However, it is sometimes necessary to measure the total number of visitors or concurrent users of a specific JENNIFER agent, rather than all JENNIFER agents.

For example, if you want to calculate the total number of visitors to the W11 and W12 JENNIFER agents only, use the agent_group option of the JENNIFER server.

```
agent_group = Group ID: agent list
```

The JENNIFER agent group ID must start with '@' and end with a two-digit number. Therefore, JENNIFER agent group ID's range from @00 to @99.

The list of JENNIFER agents belonging to the JENNIFER agent group is comprised of multiple JENNIFER agent IDs separated by commas [,]. Therefore, the above example should be set as follows.

```
agent_group = @01:W11,W12
```

Two or more JENNIFER agent groups are separated by semicolons [;].

```
agent_group = @01:W11,W12; @02:W13,W14
```

The JENNIFER agent group ID performs the same roles as the JENNIFER agent ID. Thus, all performance data for each JENNIFER agent group ID is saved in the PERF_X_01~31 table.

Warning: You can not set the same JENNIFER agent in more than two JENNIFER agent groups.

6.9. Batch JOB Monitoring

6.9.1. Batch JOB Characteristics

A batch job is a program that is terminated after executing a designated logic. A Batch JOB program has the following characteristics.

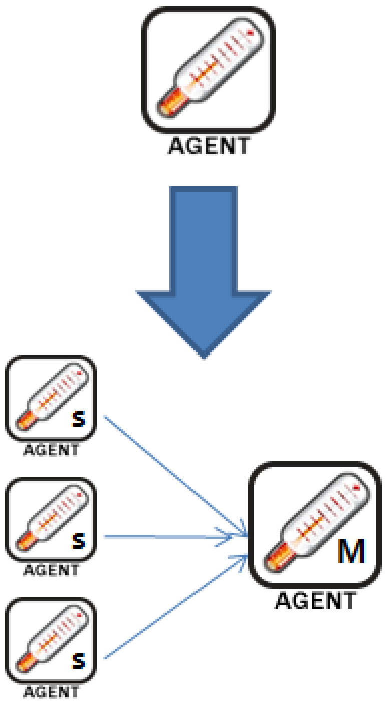
- During execution, no request is received from the user.
- It starts at the designated time-point and it is terminated after executing all the logics.
- Within one system, multiple batch jobs can be executed in a pre/post relationship.
- There can be parallel processing logics within a process.

Batch Job monitoring refers to program performance monitoring that has these characteristics.

6.9.2. Basic Concepts of Jennifer Batch JOB Monitoring

A batch job is executed and terminated within a short period of time. Therefore, one batch job is monitored as service online.

Figure 6-23:Basic Concepts of Jennifer Batch JOB Monitoring



Agent Separation

From the perspectives of Jennifer, service is executed by a process having an agent ID. Therefore, there needs to be an additional virtual master and a sub agent used to directly collect the information from each batch job.

UDP Centered Data Collection

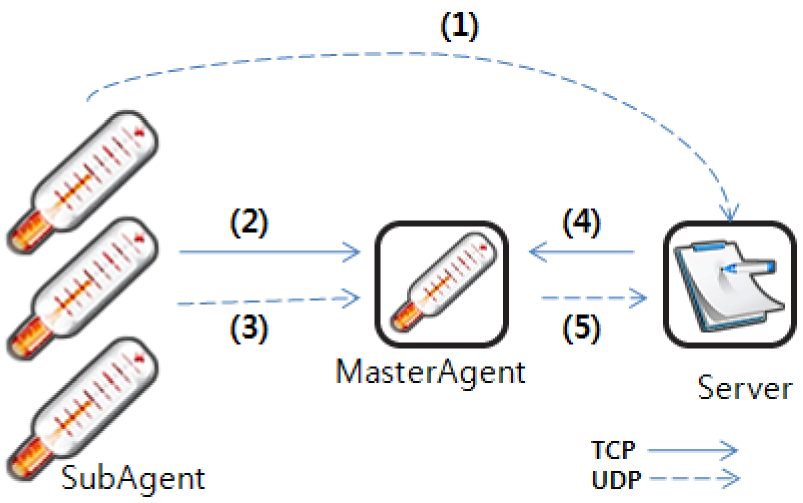
A batch job is terminated within a short period of time. So, there is no significant advantage in maintaining a TCP session. Therefore, most of information is collected through the UDP.

Using TCP Sessions for Active Stacks

Periodic statistical information is collected from the UDP, but to inquire about the active stack or the active profile of an ongoing batch job, you still need a TCP session.

6.9.3. JENNIFER Architecture for Batch JOB Monitoring

Figure 6-24:JENNIFER Architecture for Batch JOB Monitoring



3. The Runtime Data(Transaction Start/End) and the Profile Data are transmitted. From the standpoints of Jennifer, it is not important which SubAgent it is or whether the data is from a general Jennifer agent or not.
4. SubAgent is a TCP session connected to MasterAgent. Only the response to the requests sent from MasterAgent is sent here. In addition, when this session is terminated, it is automatically connected. An active service list or each thread's active stack/active profile are sent to this.
5. The summary data sent every minute is transmitted to MasterAgent. And, the text data(SQL, Application Name, Method Name) collected at SubAgent is transmitted here and SubAgent sends the application and the SQl execution statistics through (3) periodically or when the process is terminated.
6. The Jennifer Server assumes that MasterAgent is a general Jennifer Agent for communication. Especially, it collects the statistics information that SubAgent sends through (3) and every 10 minutes, it sends them through (4) to the Jennifer Server.
7. MasterAgent integrates the summary data sent by SubAgent through (3) and sends them to the Jennifer Server every second. MasterAgent does not directly execute the service, so there is no profile information generated at the start/end time.

Important characteristics are that the UDP is used for data collection and the TCP is used for controls. TCP Listening is done by the master agent and the sub agent connects one TCP session to the master agent at the time of execution.

6.9.4. SubAgent Installation

- The installation method for SubAgent is same as that for a general Jennifer Agent. But the only difference is that the current agent execution mode is SubAgent and the master agent and the Jennifer Server network information should be set.
- SubAgentMode must be added to the Java execution variables.

```
-Djennifer.subagent=true
```

- Set the master agent network information in jennifer.conf(ex. w11.conf).(The below values are the defaults when the setting is omitted..

```
master_host_name = 127.0.0.1
master_udp_listen_port = 6705
```

- You can set the network information of the Jennifer Server set for execution in the basic Jennifer mode in the same way.
- The Jennifer Agent name should be set same as the agent name for the master agent that is connected..

Notice: Therefore, if you want to install SubAgent and MasterAgent in the same machine, it is better to share the configuration file.

6.9.5. MasterAgent Installation

The master agent can be simply executed. The Jennifer Agent directory has the “masteragent.sh/bat” file. Execute this file.

The following is its contents. .

```
java  -Xbootclasspath/p:./lwst.boot.jar   \
      -cp ./jennifer.jar                  \
      -Djava.library.path=.               \
      -Djennifer.config=w11.conf          \
      com.javaservice.jennifer.agent.master.MasterAgent
```

A master agent receives the UDP data from its sub agent. Therefore, you must set the listening port. Add the following configuration in the jennifer.conf file. If not designated, the default is 6705.

```
master_udp_listen_port = 6705
```

Using the Jennifer Agent directory, execute the following file.

```
masteragent.bat
```

6.9.5.1. Profile Configuration

A batch job can lead to multiple SQL executions. If there are too many profile data, then you should start excluding the least important one. For this purpose, the several options are added. .

```
profile_ignore_connection=false
profile_ignore_fetch=false
profile_sql_time=0
```

- The above setting uses the default.
- When profile_ignore_connection is set to true, DB connection open/close are excluded from the profile information.
- When profile_ignore_fetch is set to true, the profile item called FETCH is excluded.
- When profile_sql_time is set to 10, then an SQL text executed at the speed of 10ms or below is excluded from the profile information.

6.9.6. Text Cache Controls between Master and Sub Agents

A sub agent transmits through the UDP. There is cache that records the list of transmission. You can adjust its size.

```
master_max_num_of_text = 1000
master_max_num_of_sql = 5000
```

APP, TX, ERR caches obey master_max_num_of_text and the SQL count is set by master_max_num_of_sql.

A sub agent does not store the text itself but it stores the fact that it is sent. So you can make this number little bit larger. When the value is sufficiently large, then the text is sent from sub to master only once. However, when a sub agent restarts, it will be retransmitted all over again.

Similarly, a master agent had an additional cache that stores the text received from a sub agent. This value is determined by master_max_num_of_text, master_max_num_of_sql.

6.10. .Net Batch Process Monitoring

This chapter will describe the method for monitoring a batch job by using the Jennifer Agent(.Net version).

Notice: For the concepts of batch monitoring implemented in Jennifer and installation in Java, please refer to the document, ”Batch JOB Monitoring”.

6.10.1.MasterAgent Installation and Execution

1. [conf file configuration]

Copy the ”[agent installation folder]\conf\app_pool.conf” file and create the “batchjob_master.conf” file. Since you need to make a connection to the Jennifer Server, open the batchjob_master.conf file in your memo pad and set the Jennifer Server address.

```
udp_server_host = Jennifer Server Address]
```

2. [Master Agent Execution]

The master agent execution file is installed altogether with JENNIFER, so no additional installation work is necessary. Simply, execute the module provided in the below path.

```
.NET 1.1: [Agent Installation Folder]\bin\MasterAgent.Clr10.exe
.NET 2.0 or higher: [Agent Installation Folder]\bin\MasterAgent.exe
```

6.10.2.SubAgent Installation and Execution

Copy the config file for the master agent where your sub agent will be subordinated to and create the new conf file. To imply that it is a sub agent, add the following.

```
[Ex: youragent.conf]


SUB_AGENT = true
```

3. [Connecting the config file to a batch process]

Assuming that the following execution file path is used for the batch process to be monitored,

```
C:\BatchJobs\DailyWorker.exe
```

Connect the ".config" file to the process name and create a new file in the conf folder.

```
[Agent Installation Folder]\conf\DailyWorker.exe.config
```

The file content is shown below. Designate the conf(ex. Youragent.conf) that was made for the sub agent previously. .

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <appSettings>
        <add key="Jennifer.FileName" value="youragent.conf" />
    </appSettings>
</configuration>
```

4. [User Defined Method Profiling Configuration]

In general, in a batch process, the user defined method is regarded as the unit of transactions. So, according to the method indicated in ".Net method profiling", you need to make an additional "[txserver]" section.

In addition, in case of a batch process, you must define the "[batchjob]" section and define the main function for the application program.

For instance, if the user code is defined as follows,

```
namespace BatchJob1
{
    class Program
    {
        static void Main(string[] args)
        {
        }

        void DoBatchJob()
        {
        }
    }
}
```

If DoBatchJob is the method executing the actual batch job and it is the target of monitoring by txserver, then in the profiler.ini file, you must set the configuration as follows.

```
[batchjob.process name]
BatchJob1.Program.Main=1

[txserver.process name]
BatchJob1.Program.DoBatchJob=1
```

5. [JENNIFER Agent Environment Variable Setting]

For profiling of the batch process, you must set the following environment variables prior to execution of the process.

```
SET COR_ENABLE_PROFILING=1
SET COR_PROFILER={FF8C2B6C-DBB5-4AED-9876-2CED6FFAF4C9}
```

An environment variable can be set in various methods. In case of a .bat file or NT service, you can use the registry as described in ”Environment Variable Setting”.

But if the two methods can’t help you, you can then use a global environment variable. If it is registered as a system variable, all the .net processes executed in the OS are involved with Jennifer Agent profiling works. To resolve this problem, Jennifer .NET recognizes an additional environment variable option called “JENNIFER_PROFILE”.

Therefore, if the user's batch process name is "DailyWorker.exe", then you can register the following environment variables.

```
Environment variable name: JENNIFER_PROFILE
값: dailyworker.exe
```

Notice: The process name should be lower case letters.

If there are multiple batch processes, you can use a semi colon as a separator to designate it as follows. .

```
Environment variable name: JENNIFER_PROFILE
값: dailyworker.exe;weeklyworker.exe;monthlyworker.exe
```

After setting, execute the batch process to start normal monitoring.

6.10.3.Agent Monitoring Cancel / Removal

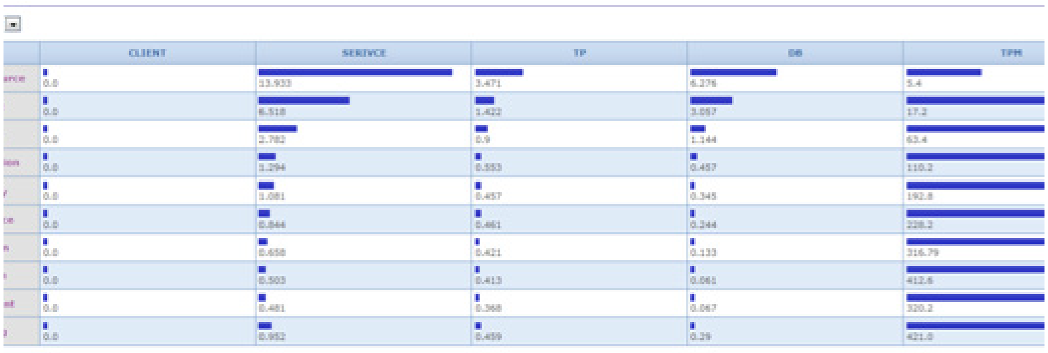
Once the Jennifer Agent is installed, if you want to cancel batch process monitoring only, then you must set , in your environment variable, "COR_ENABLE_PROFILING" to "0". (Or if it is set to는 "JENNIFER_PROFILE", omit the process name or delete the JENNIFER_PROFILE environment variable itself.)

If you want to remove the Jennifer Agent as well, then terminate all the batch processes being monitoring and execute the "[agent installation folder]/uninstall_Jennifer.bat" with the privilege of an administrator and manually delete the installation folder.

6.11. Business Monitoring

Business Monitoring is one of the new functionalities in JENNIFER® 4.5, with this functionality you are able to monitor the performance data collected from business groups.

Figure 6-25:Business Monitoring

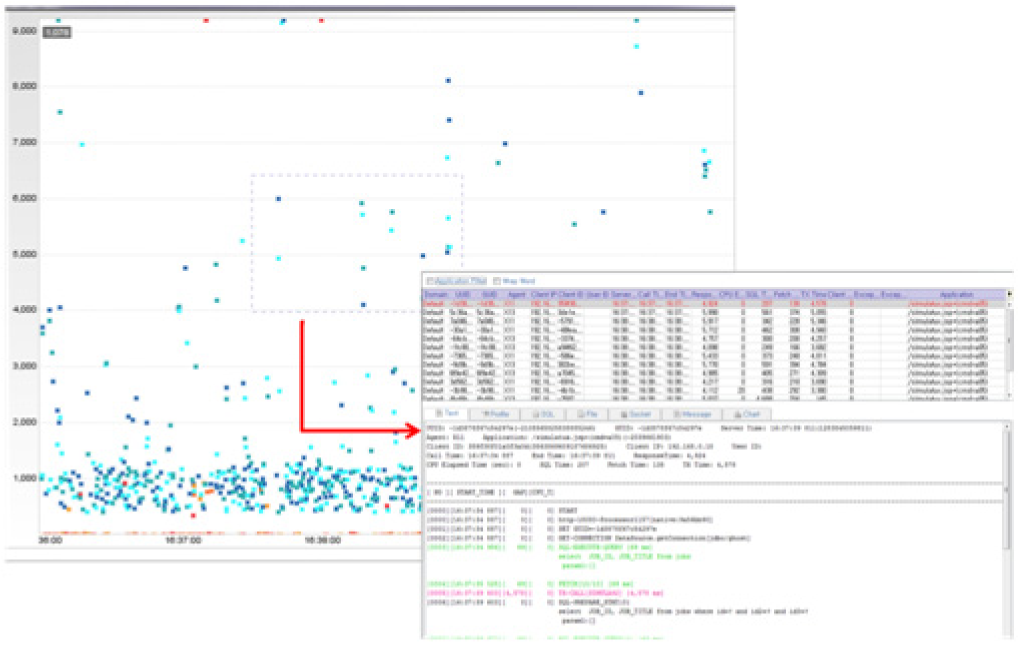


	CLIENT	SERIVCE	TP	DB	TPM
arce	0.0	13.933	3.471	6.276	5.4
	0.0	6.518	1.422	3.057	17.2
	0.0	2.782	0.9	1.144	63.4
ion	0.0	1.294	0.553	0.457	110.2
y	0.0	1.081	0.457	0.345	192.8
ce	0.0	0.844	0.461	0.244	228.2
n	0.0	0.658	0.421	0.133	316.79
	0.0	0.503	0.413	0.061	412.6
nt	0.0	0.481	0.368	0.067	320.2
	0.0	0.952	0.459	0.29	421.0

The main screen of the business monitoring function shows various performance data for 5 minutes intervals about the business groups, this performance data such as the Client Elapsed Time, Average Response Time, External Transactions Time, SQL Time and Fetch Time and Transactions per Second.

Clicking one of the business group's name from the left list will pop-up a new window that displays the business groups transactions in an X-View, more information can be obtained by selecting transactions from the X-View.

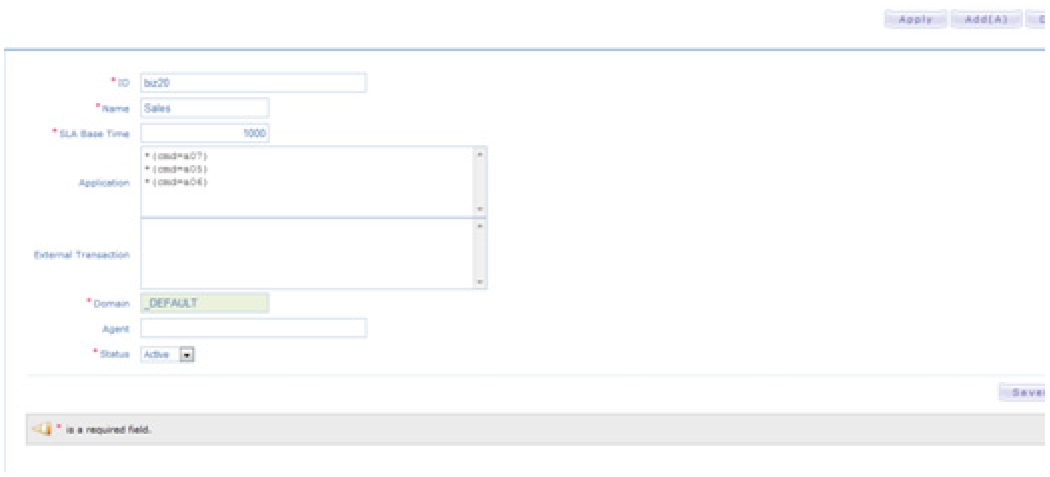
Figure 6-26:Business Monitoring Settings



6.11.1.Business Monitoring Settings

The business group monitoring can be configured from the [Business Group] option located under the [PROPERTIES] tab in JENNIFER server. You can edit existing group's settings by clicking the group's ID, or you can add new settings by clicking the [ADD] button under the group list. When a user wants to add a new group, a few settings should be configured.

Figure 6-27:Business Monitoring Settings



*ID	biz20
*Name	Sales
*SLA Base Time	1000
Application	*(cmd=a07) *(cmd=a05) *(cmd=a06)
External Transaction	
*Domain	_DEFAULT
Agent	
*Status	Active

* is a required field.

A unique ID and Name for the business group should be specified in the ID and Name fields respectively. The SLA Base Time is used in monitoring the sla_fail transactions. The Status field is required filed and you should select the group's status whether Active or Inactive. Also you may specify a list of applications in the application field, you will need to add the application's prefix or postfix such as *(cmd=a07), if you want to add multiple applications postfix, each of them should be added in a new line. Finally you should click the [SAVE] button to save the new group's settings. Note that the newly added group will not appear in the Business Monitoring page unless you select it and click the [APPLY] button.

7 Charts and Dashboards

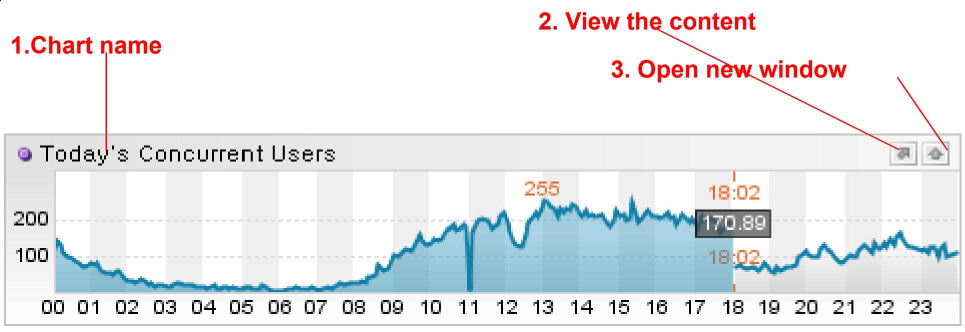
JENNIFER provides various charts, which allows the user to intuitively monitor the performance data, as well as the dashboard, where the user can comprehensively monitor various performance data. JENNIFER also allows user-defined dashboard functions, so that the user can configure the dashboard by his or her own specifications.

7.1. Charts

7.1.1. General Charts

Most of the charts are organized as follows.

Figure 7-1:Chart



- Chart name - The chart name describes the type and characteristics of the performance data displayed by the chart. A run-time line chart name starts with [Recent], while an equalizer chart name starts with [Real time]. And when a bar or line chart displays today's performance data, its names starts with [Today].
- View the content - By clicking this button, you can view the chart in a new window.
- Open new window - If you click this button, you can view the chart in a new window. The user can adjust the chart size using this window.

Notice: If you press CTRL + C button while clicking the chart, the chart image is copied in the clip board.

Set the font size of a chart name using the ui_chart_text_size option of the JENNIFER server. 11 is default. If you set this option, you can also change the font of text message such as alert message on the ;aert list chart.

```
ui_chart_text_size = 12
```

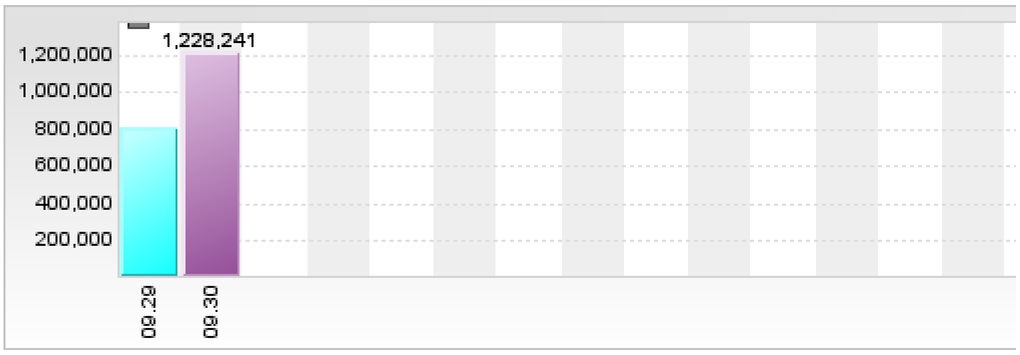
Set the figure size of X and Y axis using the ui_chart_number_size option of the JENNIFER server. 10 is default.

```
ui_chart_number_size = 11
```

7.1.2. Bar Chart

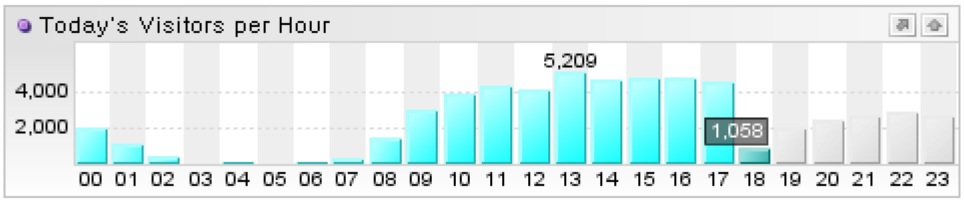
A bar chart displays the cumulative quantitative value depending on the time change. For instance, the bar chart in the following figure represents the number of daily hits. In this case, the X-axis represents the date.

Figure 7-2:Daily Hits



In addition, a bar chart can be used to represent the hourly data such as the number of visitors per hour. In this case, the X-axis represents the time.

Figure 7-3:Visitors per Hour on Selected Date



Notice: In general, a bar chart is used to express the number of calls or visitors.

The maximum of a bar chart is displayed on top of the chart. By clicking a bar in the bar chart, you can see the value displayed by the bar.

If the chart name begins with Today, like the daily visitors per hour, it means that the performance data collected on the day is displayed. In this case, the current value is displayed in the box on top of each bar. The bar in the right of the current bar displays yesterday's performance data, while the bar in the left of the current bar displays today's performance data. Different colors are used for the bars that represent today and yesterday's data.

If you click the right mouse button in the bar chart, the context menu appears. Using this menu, you can hide the max value and current value.

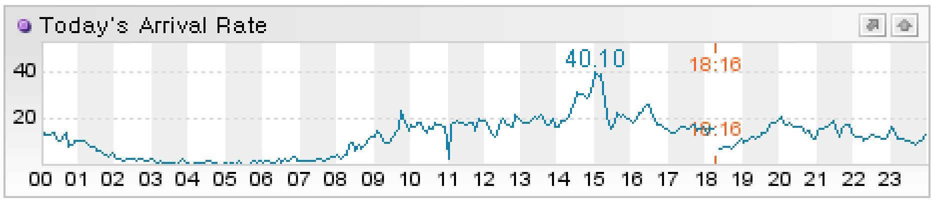
7.1.3. Line Chart

A line chart represents the change in the five-minute average data for the day on a line. If you divide 24 hours into 5-minute segments, you have 288 segments. Thus, the line chart basically connects 288 points in a line.

Notice: The average response time for a certain application during the day is collected once every 10 minutes. So, when the data is displayed in a line chart, 144 points are connected in a line. The application, the SQL and the external transactions are examples of performance data collection every 10 minutes.

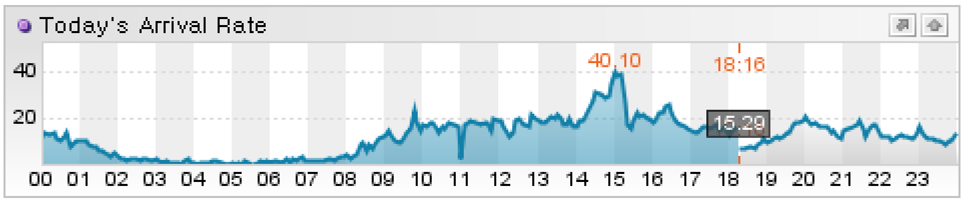
For instance, the line chart in the following figure represents the change in the arrival rate for the day on a line.

Figure 7-4:Today's Arrival Rate



Notice: An area chart displays the same date as a line chart. However, the difference is that the background color is displayed below the line.

Figure 7-5:Today's Arrival Rate - Area Chart



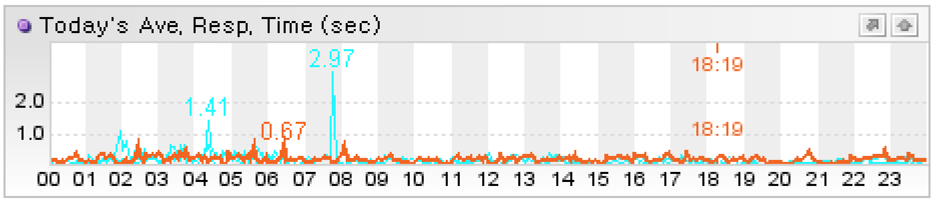
The maximum of a line chart is displayed as a number on the coordinate.

If the chart name begins with Today, like today's arrival rate, the performance data collected today is displayed. In this case, the current value is displayed in the box on the coordinate. The line to the right of the current line displays yesterday's performance data. On the other hand, the line to the left of the current coordinate displays today's performance data. Different colors are used for the charts that display today's and yesterday's data.

If you click the right mouse button in the area chart, the context menu appears. Using this menu, you can hide the max value and current value.

Multiple lines can be displayed in a line chart. In this case, the performance data for each JENNIFER agent is displayed. If you select a line, it will be highlighted.

Figure 7-6:Today's Average Response Time per Server

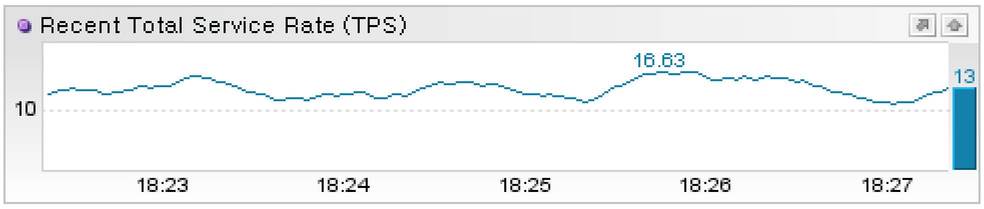


7.1.4. Runtime Line Chart

A runtime line chart displays the change in the 30-second average data during the most recent 5 minutes on a line.

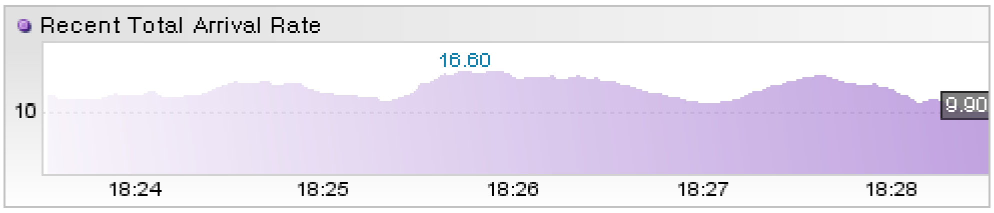
For instance, the runtime chart in the following figure displays the change in the 30 second average service rate during the recent 5 minutes on a line.

Figure 7-7:Recent Total Service Rate



Notice: A runtime area chart displays the same information as a runtime line chart. However, the only difference is that the background color is displayed below the line.

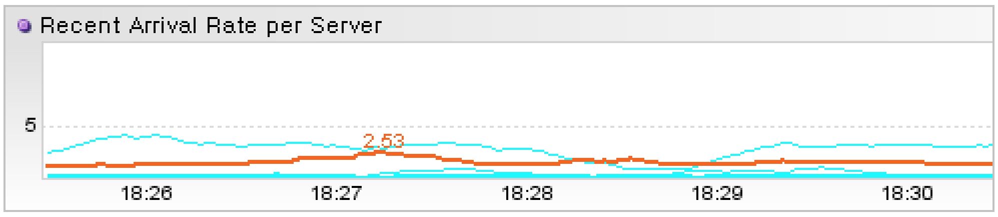
Figure 7-8:Recent Total Arrival Rate - Run-time Area Chart



The maximum of a runtime line chart is displayed as a number on the coordinate.

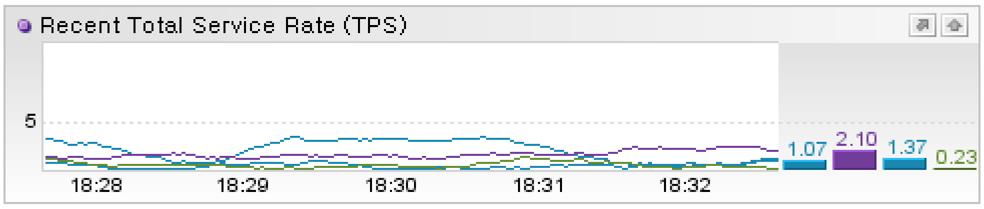
If you click the right mouse button in the run-time area chart, the context menu appears. Using this menu, you can hide the max value and current value. Multiple lines can be displayed in a runtime line chart. In this case, the performance data for each JENNIFER agent is displayed. If you select a line, it will be highlighted.

Figure 7-9:Recent Arrival Rate per Server



A runtime line chart can display the current value on the bars in the right.

Figure 7-10:Recent Total Service Rate - Displaying the Current Value



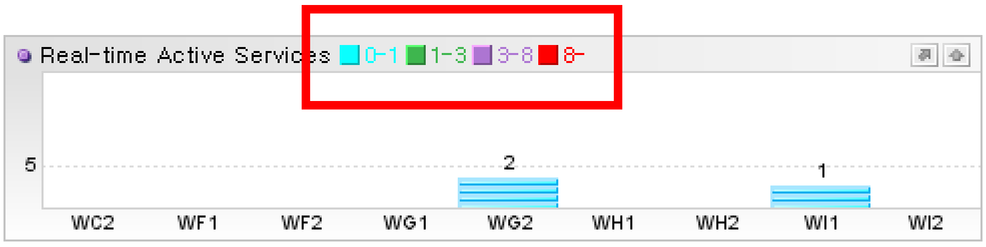
You can save the data which is displayed in the runtime chart and runtime area chart as a CSV format file. Press the CTRL + E button whlie you are clicking the chart.

7.1.5. Equalizer Chart

An equalizer chart displays the real-time 30-second average in an equalizer format.

For instance, the equalizer chart in the following figure represents the real-time 30-second average active services.

Figure 7-11:Real-time Active Services



In addition, an equalizer chart can show the detailed data for certain criteria. For instance, the number of active services for the JENNIFER agent can be detailed for certain time regions. Basically, the time regions for the active services are comprised of 0 to one second, one second to three seconds, three seconds to eight seconds and over eight seconds.

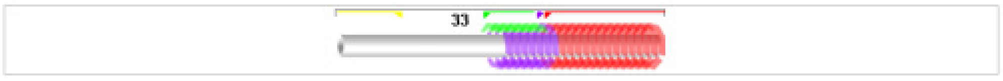
Notice: When the performance data is classified for certain criteria, the criteria will be displayed next to the equalizer chart name.

The CPU utilization can be classified by the CPU utilization regions (WAIT, NICE, USER, SYS) and the real-time JDBC connections can be classified by the JDBC connection status(IDLE, ALLOCATED, ACTIVE).

7.1.6. Speed Bar

A speed bar intuitively represents the load condition of an application based on performance data such as the arrival rate, the service rate and the active services.

Figure 7-12:Speed Bar



The cylinder in the middle of the speed bar represents the Java application while the ring surrounding it represents the active services. Like an equalizer, it partitions the data into time regions and visualizes the active services.

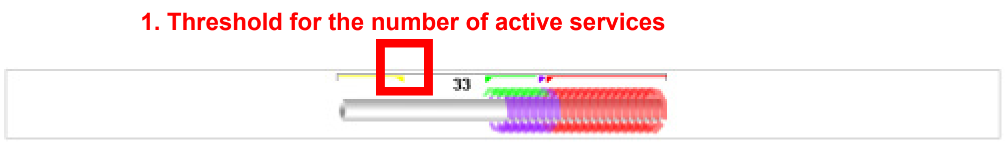
Notice: The number of rings does not represent the number of active services. The proportion of the active services occupying the full active services is displayed by using the rings with the same color and size.

When the number of active services exceeds the threshold, the (ERROR_SERVICE_QUEUING) alert is generated. The threshold is set using the active_service_alert_limit option of the JENNIFER server.

```
active_service_alert_limit = 70
```

However, even when the number of active services far exceeds the threshold, the ring still stays inside the cylinder.

Figure 7-13:Threshold for the Number of Active Services



- Threshold for the number of active services. - This is the threshold for the number of active services shown by the speed bar. When it exceeds the threshold and surrounds the cylinder, the ERROR_SERVICE_QUEUING alert is generated.

The bar moving from left to right in the speed bar represents the transaction. The number of individual transactions is calculated from the arrival rate in the left and the service rate in the right.

Depending on the number of transactions, the speed bar will show different speeds. It is necessary to set the speed sensitivity according to the arrival rate of each Java application. For instance, if the average arrival rate for Java applications A and B are 10 and 30 respectively, then the speed corresponding to the arrival rate of 30 in Java application seems very high, while the speed corresponding to the arrival rate of 30 in Java application B seems mediocre.

For this reason, the user should adjust the speed sensitivity of the speed bar in consideration of the average arrival rate for the monitored Java application. This can be done using the speed_threadhold option of the JENNIFER server. The default is 1000.

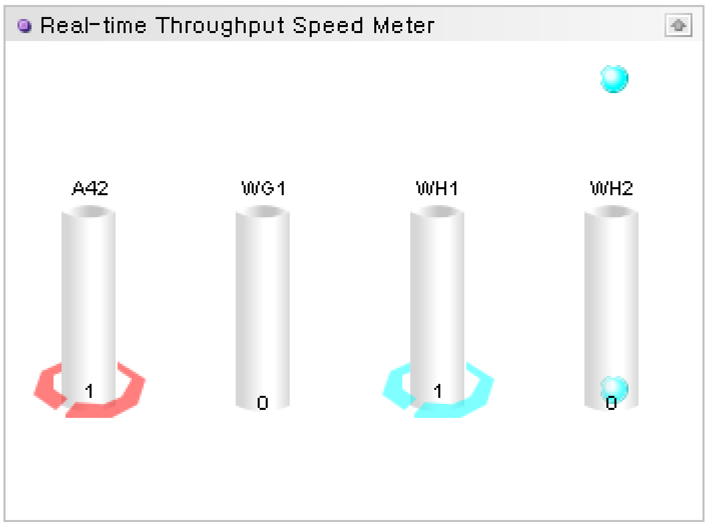
```
speed_threadhold = 1000
```

If you set the speed_threadhold option to higher than 1000, the same arrival rate in the speed bar seems slow, but when lower than 1000, it seems faster. To apply this option, you need to log out and then log in again.

7.1.7. Speedmeter

A speedmeter is conceptually the same as a speed bar, but displays the load conditions for all of the JENNIFER agents simultaneously.

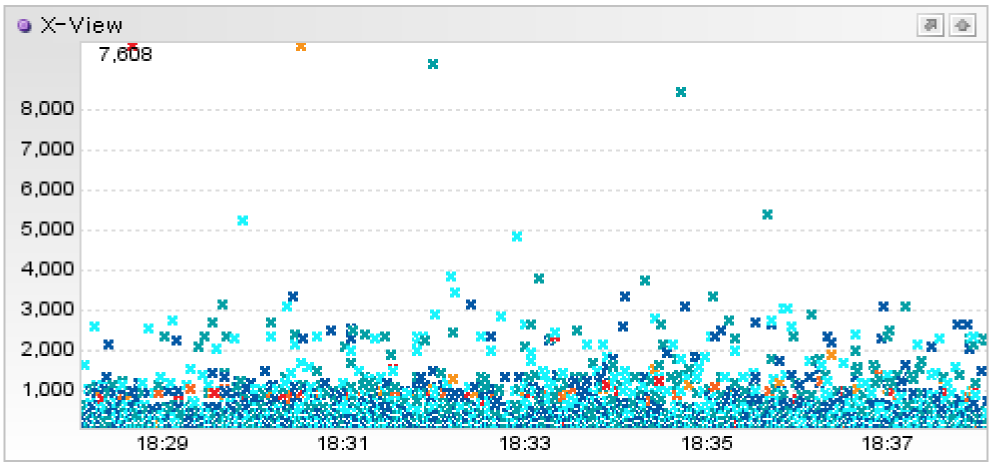
Figure 7-14:Speedmeter



7.1.8. X-View Chart

The X-View chart is a distribution chart of response time that displays the beginning and end of the transaction as a point.

Figure 7-15:X-View Chart



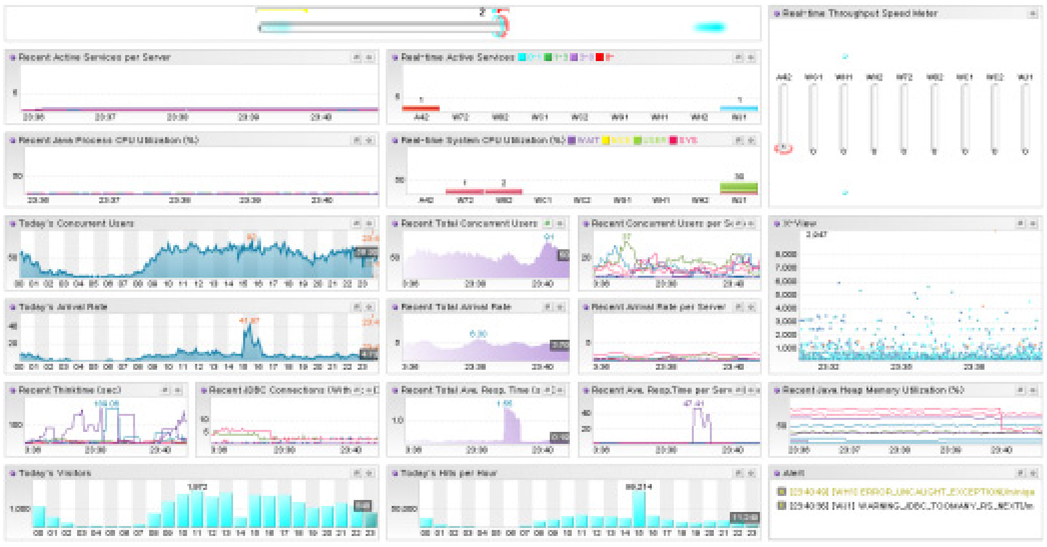
For more details on the X-View chart, please refer to [X-view and Profile].

7.2. Dashboard

7.2.1. JENNIFER Dashboard

By selecting **[Dashboard | JENNIFER Dashboard]** from the menu, you can comprehensively monitor a range of performance data. The JENNIFER dashboard focuses on the effective monitoring of the Java application.

Figure 7-16:JENNIFER Dashboard



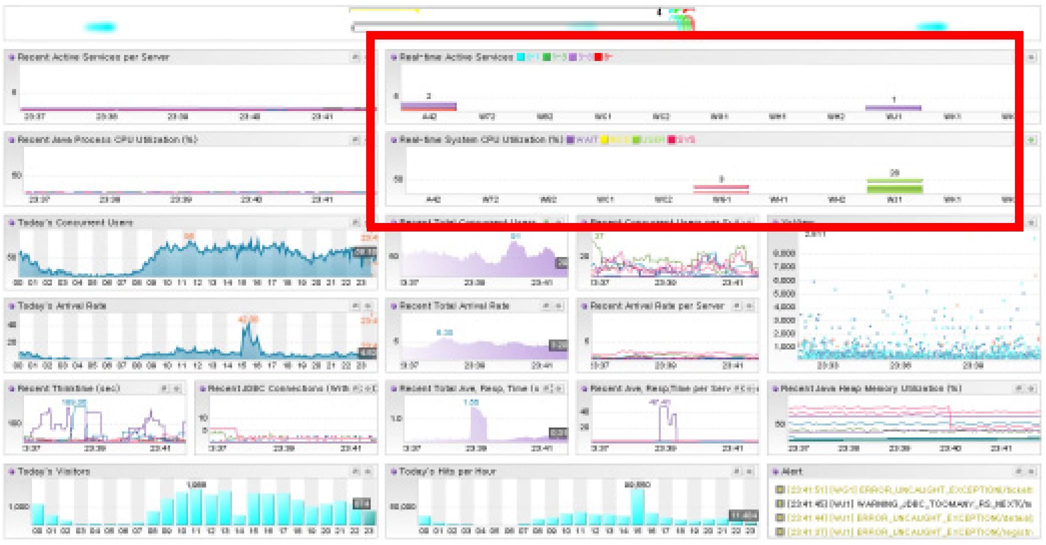
You can adjust the size of the equalizer chart displaying real-time active services and real-time CPU utilization based on the number of JENNIFER agent. If you are monitoring the exceeded number of JENNIFER agents set by the dashboard_equalyzer_min option, the speedmeter is not displayed due to space limitations. The default is 9.

```
dashboard_equalyzer_min = 9
```

In addition, if you are monitoring the exceeded number of JENNIFER agents set by the dashboard_equalyzer_max option, the run-time line chart is not displayed. The default is 29.

```
dashboard_equalyzer_max = 29
```

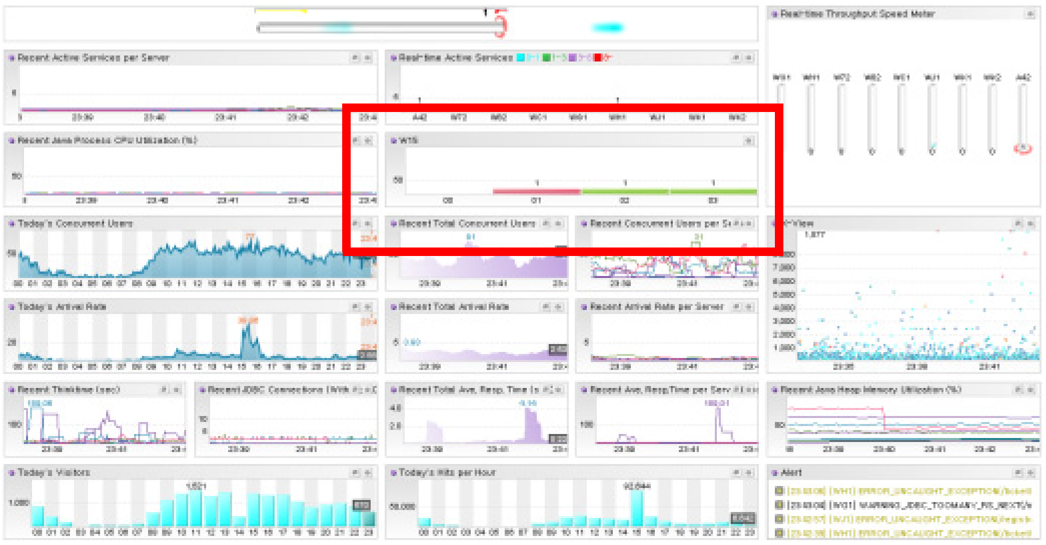
Figure 7-17:JENNIFER Dashboard - with more than 10 JENNIFER Agents



Instead of the real-time system CPU utilization equalizer in the upper middle area, you can use the equalizer with the data collected by WMOND. If you want to use the equalizer with the WMOND data, you should set the dashboard_show_wmond option of the JENNIFER server to ‘true’.

```
dashboard_show_wmond = true
```

Figure 7-18:JENNIFER Dashboard - Using the Equalizer with the System CPU Utilization collected by WMOND



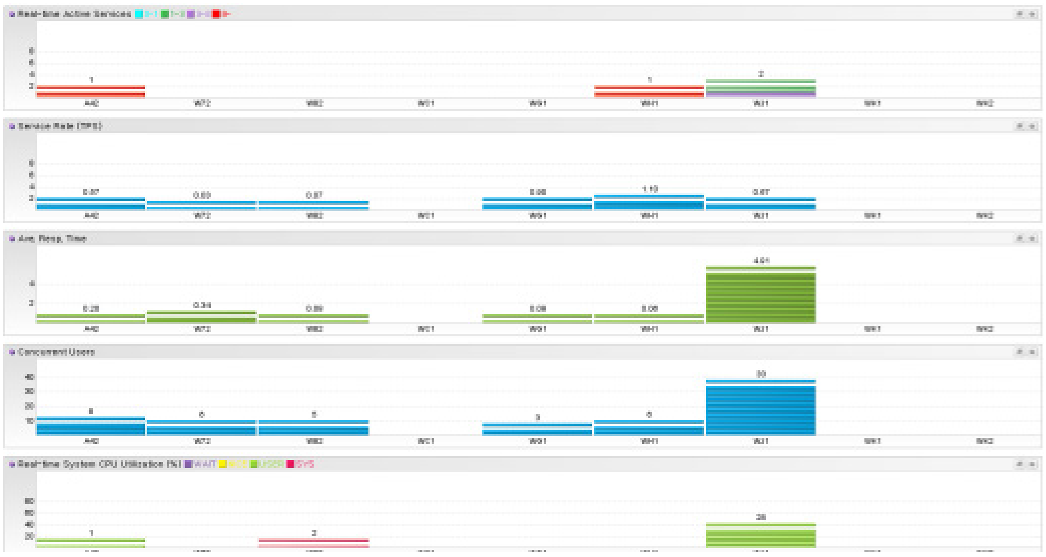
Instead of the recent Java heap memory utilization run-time line chart in the lower right area, you can use the recent Java heap memory utilization run-time line chart displaying both total Java heap memory utilization and Java heap memory utilization. If you want to use the run-time line chart, you should set the dashboard_show_tot option of the JENNIFER server to ‘true’.

```
dashboard_show_heap_tot = true
```

7.2.2. Equalizer Dashboard

By selecting **[Dashboard | Equalizer]** from the menu, you can use an equalizer to monitor the main performance data. The equalizer dashboard is optimized for real-time monitoring of various performance data for each JENNIFER agent.

Figure 7-19:Equalizer Dashboard



The equalizer dashboard provides the following real time performance data.

- Real-time Active Services - Displays the number of active services for individual JENNIFER agents in each time segment.
- Service Rate(TPS)- Displays the service rate for individual JENNIFER agents in each time segment.
- Average Rsponse Time - Displays the average response time for individual JENNIFER agents in each time segment.
- Concurrent Users- Displays the number of concurrent users for individual JENNIFER agents in each time segment.

- Real-time system CPU utilization - Displays the system CPU utilization for individual JENNIFER agents in each time segment.
- Real-time Java heap memory utilization - Displays the Java heap memory utilization for individual JENNIFER agents, or the proportion of the full Java heap memory being utilized.
- Real-time JDBC connections - Displays the number of JDBC connections for individual JENNIFER agents in various connection modes.

7.3. User-defined Dashboard

A user can organize an arbitrary drag-and-drop type of dashboard using JENNIFER independent agents like REMON. Dashboards like these are called user-defined dash-boards.

7.3.1. Creating the User-defined Dashboard Menu

If you want to organize a user-defined dashboard, you should first add a user-defined dashboard menu.

- Select the **[Properties | Menu Setting]** menu.
- Click on the upper menu on which the user-defined dashboard menu will be located below and select the context menu and click on the **[Add Chile Menu]** menu.
- Enter the name in the right menu input form and select the **[User-defined Dashboard Menu]** type.
- Click on the **[save]** button.

After adding the user-defined dashboard menu, move to the menu.You will see that the newly added user-defined dashboard is empty. But in the lower right corner, you will see the **[Edit]** and **[Change]** buttons.

Notice: Using the **[Dashboard | User Defined Dashboard Demo]** menu option, you can practice organizing the user-defined dashboard.

7.3.2. Editing the User-defined Dashboard

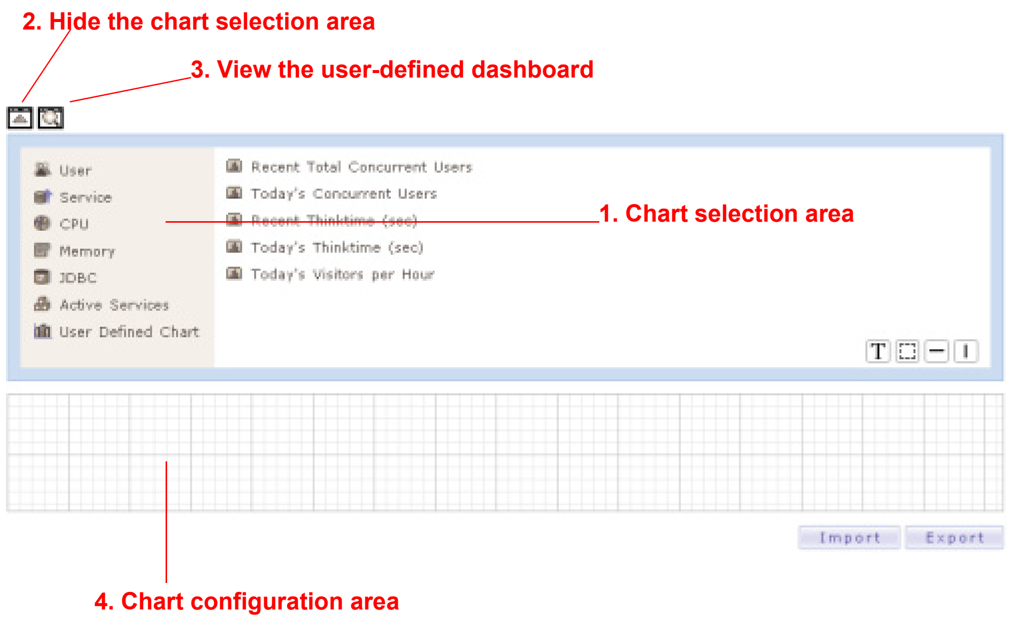
If you want to edit the user-defined dashboard, click the [Edit] button in the lower right corner.

Notice: Only the users with pageedit privileges will see the [Edit] button.

7.3.2.1. Organizing the User-Defined Dashboard Edit Screen

The user-defined dashboard editi screen is organized as follows.

Figure 7-20:User-Defined Dashboard Edit Screen



1. Chart selection area - In this area, you can select the charts that you want to have on your user-defined dashboard. When you select a group of charts on the left, all charts belonging to the group will appear. If you select a chart, it will be displayed on the right.
2. Hide the chart selection area - If you click this button, the chart selection area will be hidden. If you click on it while the area is hidden, it will reappear.
3. View the user-defined dashboard - If you click the button, you view the current state of the user-defined dashboard. You can see it using the upper area menu.
4. Chart configuration area - You can drag a chart from the chart selection area and drop it in the chart configuration area. If you drop it outside of the chart configuration area, it will not be displayed correctly.

7.3.2.2. Adding a Chart

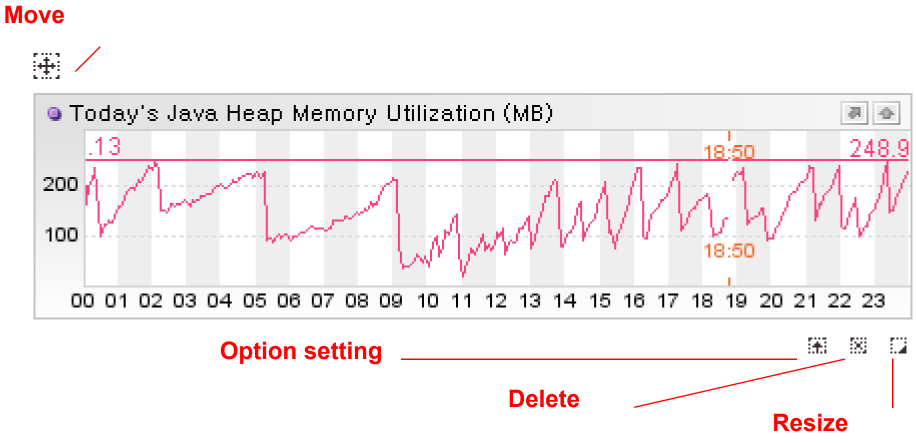
If you want to add a chart, you should first select a chart from the chart selection area to be placed in the user-defined dashboard. The charts are comprised of several groups with different characteristics.

Table 7-1: Chart Groups

Groups	Descriptions
Uer	# of concurrent users, Think time, # of visit users, etc
Service	Arrival rate, Service rate(TPS), Average response time, Hits per hour, etc
CPU	System CPU utilization, Java process CPU utilization, etc
Memory	Java Heap memory utilization, System memory utilization, Java process memory utilization, etc
JDBC	# of JDBC connections, etc
Active Service	# of active services, Speed bar, Speed meter, X-View, etc
User -defined Chart	A chart which is used to display the performance data collected by JENNIFER independent agent such as REMON.

Once a chart has been placed in the user-defined dashboard edit screen, several icons will be displayed in the upper left and lower right corners. The user can move the chart or adjust its size using these icons.

Figure 7-21:Charts and Icons in the User-Defined Dashboard Editing Screen



A user can add a chart in the user-defined dashboard as follows.

- In the chart selection area, you should select a chart to add to the user-defined dashboard. The chart will then appear at the right.

- After clicking on the move icon in the chart, simply drag-and-drop to place the chart in the chart configuration area.

7.3.2.3. Moving a Chart

You can move a chart as follows.

- Click on the move icon in the upper area of the chart to be moved.
- Drag and drop the chart to the desired location.

7.3.2.4. Resizing Chart Size

You can resizing a chart as follows

- Click on the resizing icon in the lower area of the chart to be resized.
- Drag and drop the chart to the desired size.

7.3.2.5. Deleting a Chart

You can delete a chart as follows.

- Click on the delete icon in the lower area of the chart to be deleted.

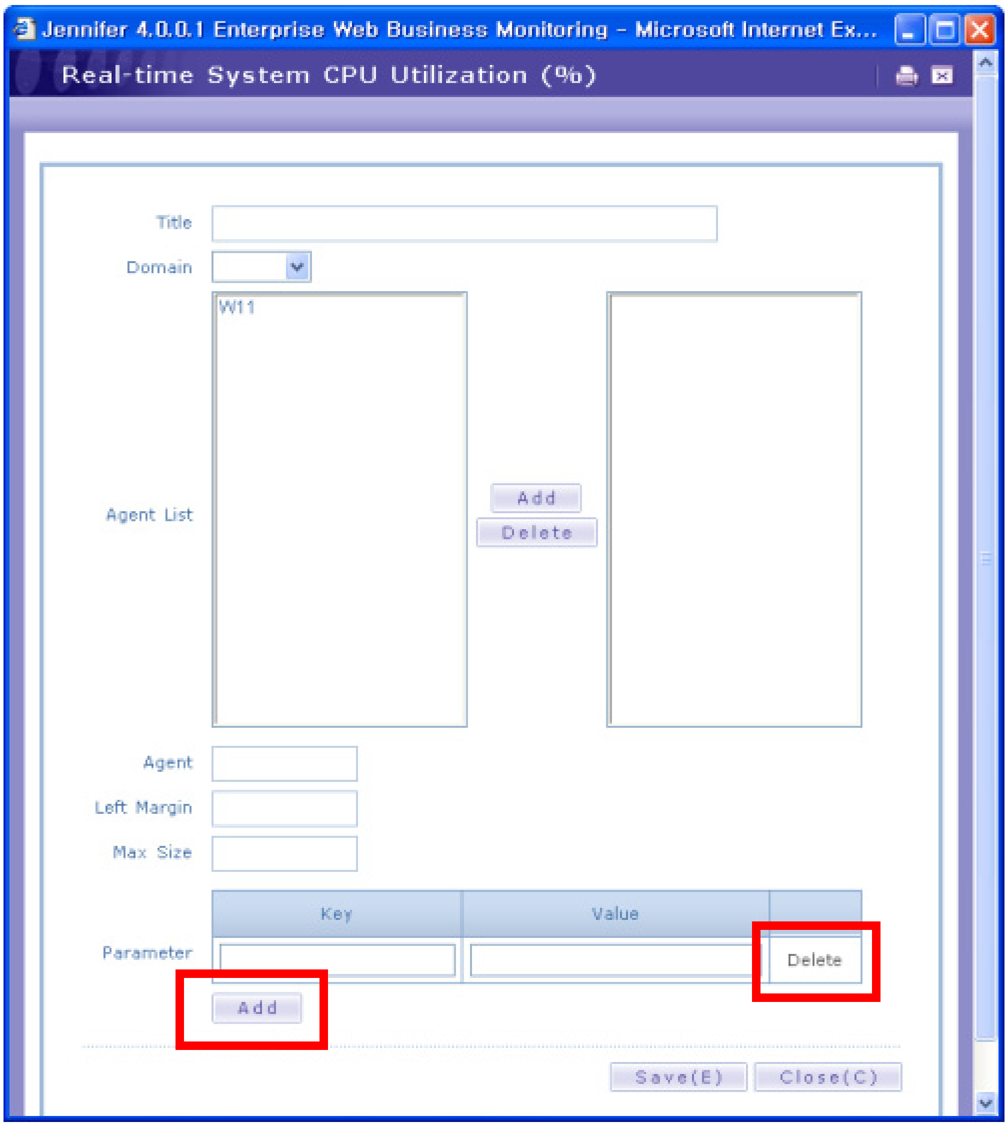
Warning: The charts are implemented in Java applets, while the icons are in HTML tags. Due to technical limitations, an HTML tag cannot be placed over a Java applet chart. Therefore, if charts overlap one another, some icons (resize, move) could be hidden by other charts. This must be considered when designing a user-defined dashboard.

7.3.3. Setting the Chart Options

If you want to change the chart name or assign the maximum to the Y-axis, you should first set the chart options. The following steps describe how you can set the chart options.

- Click on the option setting icon of the chart.
- Set the chart options in the option setting pop-up window.

Figure 7-22:Option Setting Pop-up Window


Jennifer 4.0.0.1 Enterprise Web Business Monitoring - Microsoft Internet Ex...
Real-time System CPU Utilization (%)
Title
Domain
W11
Agent List
Add
Delete
Agent
Left Margin
Max Size
Key
Value
Parameter
Delete
Add
Save(E)
Close(C)

Notice: If the chart is in the chart configuration area, rather than the chart selection area, you can use this option setting icon.

Options are comprised of the field option, where you can enter the title, the left margin and the maximum; and the parameter option, where you can enter the keys and the formats. Available field options will vary depending on the characteristics of the chart.

The following is a list of field options that you can set in most charts.

- Click on the [Add] button and input the key and value in the fields.

- Click on the [Delete] button if you want to delete the parameter option.

Notice: After setting the options, you should click the [Save] button to apply the changes.

The following field options can be used in the most chart.

Table 7-2: Common Field Options

Field option	Descriptions
Title	This is the chart name. You can use this option in every chart.
Domain	Select a JENNIFER server. If there is one JENNIFER server, then [Default] is selected. If there are multiple JENNIFER servers, then the JENNIFER agent may vary depending on the domain selection. You can use this option in every chart.
Agent	Select a JENNIFER agent.
Agent list	Used to display multiple JENNIFER agents in a line chart, runtime line chart or equalizer chart. However, you should select [All] for the JENNIFER option.
Right bar display	Use to display the current value using the right bar in a runtime chart.
Right bar size	Used to set the right bar size after setting the right bar display option.
Left margin	Used to set the area size on the Y-axis coordinate. According to the max value of the data, the area in the Y-axis would be changed. In this case, you can adjust the arbitrary value using the left margin.
Maximum	Used to set the maximum for the Y-axis coordinate.

The following parameter options can be used in every chart.

Table 7-3: Common Parameter Options

Parameter option	Descriptions
OMIT_TITLE	If set to true, the chart title will not be displayed.
OMIT_BORDER	If set to true, the chart border will not be displayed.
ENABLE_POPUP	If set to true, the screen will be displayed in a new window when the chart pop-up icon is clicked.
URL_LINK	If the ENABLE_POPUP option is set to true, the screen designated by URL_LINK will open in a pop-up window. If the URL_LINK is not set, then the /pop_remon_detail.jsp screen will open.

7.3.4. General Charts

7.3.4.1. User

The following is the charts offered by the User group.

- Real-time Concurrent Users - Displays the # of current concurrent users through a equalizer chart.
- Recent Total Concurrent Users- Displays the # of concurrent users for the recent 5 minutes through a runtime chart.
- Today's Concurrent Users - Displays the # of today’s concurrent users through a runtime chart.
- Recent Thinktime - Displays the think time for the recent 5 minutes through a runtime chart.
- Today's Thinktime -Displays the today’s think time through a line chart.
- Today’s Visitors per Hour - Displays the # of today’s visit users per hour through a bar graph.

7.3.4.2. Service

The following is the charts offered by the Service group.

- Recent Total Arrival Rate - Displays the arrival rate for the recent 5 minutes through a runtime chart.
- Today's Arrival Rate - Displays the today’s arrival rate through a line chart.
- Real-time Service Rate (TPS) - Displays current service rate (TPS) through a equalizer chart.
- Recent Total Service Rate (TPS) - Displays the service rate for the recent 5 minutes through a runtime line chart.
- Today's Service Rate (TPS) - Displays the today’s service rate through a line chart.
- Real-time Ave. Resp. Time (sec) - Displays ccurrent average response time(sec) through a equalizer chart.
- Recent Total Ave. Resp. Time - Displays the average respose time for the recent 5 minutes through a runtime line chart.
- Today's Ave. Resp. Time - Displays the today’s average response time through a line chart.
- Today's Hits per Hour - Displays the # of today’s hits per hour through a line chart.

7.3.4.3. CPU

The following is the charts offered by the CPU group.

- Real-time System CPU Utilization - Diplays the current system's CPU utilization through a equalizer chart.
- Recent System CPU Utilization- Displays the system CPU utilization for the recent 5 minutes through a runtime line chart.
- Today's System CPU Utilization - Displays the today's system CPU utilization through a line chart.
- Real-time Java Process CPU Utilization - Diplays the current Java process CPU utilization through a equalizer chart.
- Recent Java Process CPU Utilization -Displays the Java process CPU utilization for the recent 5 minutes through a runtime line chart.
- Today's Java Process CPU Utilization - Displays the today's Java process CPU utilization through a line chart.

7.3.4.4. Memory

The following is the charts offered by the Memory group.

- Real-time Java Heap Memory Utilization - Displays current Java Heap Memory utilization through a equalizer chart.
- Real-time Java Process CPU Utilization - Displays current Java process CPU utilization through a equalizer chart.
- Recent Java Heap Memory Utilization - Displays current Java Heap Memory utilization for the recent 5 minutes through a runtime line chart.
- Recent Java Heap Memory Utilization - Displays the Java heap memory utilization for the recent 5 minutes through a runtime line chart.
- Today's Java Heap Memory Utilization- Displays the today's Java heap memory utilization through a line chart.
- Today's Java Heap Memory Rate - Displays the today's Java heap memory utilization through a line chart.
- Recent System MEM Utilization - Displays the system memory utilization for the recent 5 minutes through a runtime line chart.
- Today's System Memory Utilization - Displays the today's system memory utilization through a line chart.
- Recent Java Process Memory Utilization - Displays the Java process memory for the recent 5 minutes through a runtime line chart.

- Today's Java Process Memory Utilization - Displays the today’s Java process memory utilization through a line chart.

7.3.4.5. DB

The following is the charts offered by the DB group.

- Real-time DB Connections - Diplays the # of current DB connections through a equalizer chart.
- Recent DB Connections - Displays the # of DB connections for the recent 5 minutes through a runtime line chart.

7.3.4.6. Active Service

The following is the charts offered by the Active Service group.

- Real-time Active Services - Diplays the # of current active services through a equalizer chart.
- Recent Total Active Services - Displays the # of active servoces for the recent 5 minutes through a runtime line chart.
- Today's Active Services - Displays the # of today’s active services through a line chart.
- Real-time Throughput Speed Bar - Displays the total arrival rate, service rate(TPS) and the # of active services.
- Real-time Throughput Speed Meter - Displays the total arrival rate per each JENNIFER agent, service rate(TPS) and the # of active services per each JENNIFER agent.
- X-View - Displays the each transaction’s profile data.

In the X-View chart, you can specify whether or not to display the SQL parameter in the profile tap through the chart option, [Display SQL Parameter].

7.3.5. User-Defined Charts

Groups such as users, services, CPU, memory, JDBC and active services provide necessary charts for organizing the user-defined dashboard with the performance data collected from the JENNIFER agent. In addition, the user-defined charts are needed for organizing the user-defined dashboard with non-standard performance data collected from a JENNIFER independent agent, such as REMON.

Notice: CPU, node or alert charts belonging to the user-defined chart group are not related to the REMON module.

The JENNIFER client gets the REMON data from the JENNIFER server at regular time intervals. If the REMON data is not transmitted within that intervals, the value becomes zero(0). The time interval is defined based on the time that occurs between when a JENNIFER server sends the data to the JENNIFER client.

In addition, when the user-defined chart such as the Line graph has the X-axis, you can adjust the value of the X-axis through a buffer field. The value size of the X-axis is defined based on the value that multiplies option value in the buffer field by time intervals about REMON data collection. For instance, if the buffer value is 30 and the time intervals is 1 munite, the range of the X-axis becomes 30 minutes.

If you do not set the buffer option, the default is 288. We recommend you set the proper buffer time when you use the user-defined chart that displays consecutive values like LINE, STACKED LINE , TABLE, BOX LINE, and BOX BAR chart.

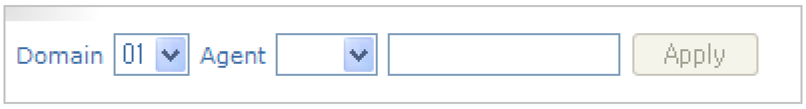
However, set the buffer time to 3 when you use the other user-defined chart that displays the only current value such as EQUALIZER, HORIZONTAL BAR, PIE, ON/OFF CHECK, NUMBER, GAUGE chart in order to reduce the Java Plug-in memory utilization.

The reason to set the buffer time to 3 not 1 for displaying the current value is to indicate zero(0) in case the JENNIFER server fails to gether the REMON data.

7.3.5.1. AGENT SELECTOR

AGENT SELECTOR chart controls the other chasrts by selecting the JENNIFER agent.

Figure 7-23:AGENT SELECTOR



If you want to use the AGENT SELECTOR chart, set the below parameter in the user-defined chart.

```
AGENT_SELECTABLE = true
```

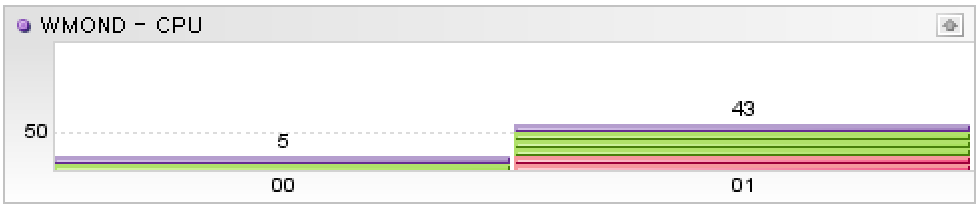
7.3.5.2. CPU

The CPU user-defined chart displays the CPU utilization collected by WMOND on an equalizer chart.

If you want to use a user-defined CPU chart, you should set the chart option, [Agent].

Notice: Since WMOND is not related to the JENNIFER agent, you cannot select the agent from a drop-down list, and must directly enter the WMOND process ID in the text input field.

Figure 7-24:CPU User-Defined Chart



The following table represents the field options that can be set.

Table 7-4: CPU User-Defined Chart Field Options

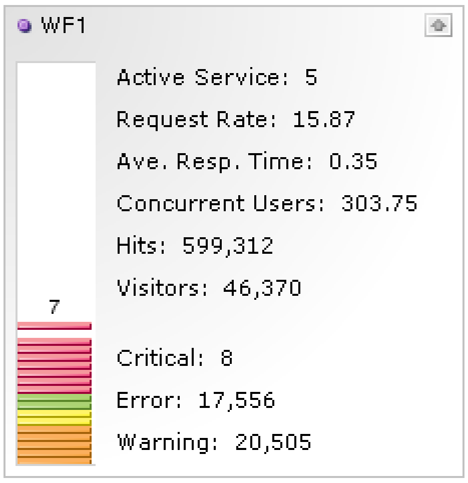
Field option	Descriptions
Agent	Enter the WMOND process ID. It is not relevant to the JENNIFER agent.

7.3.5.3. Node

The node user-defined chart provides a summary of a certain JENNIFER agent.

If you want to use the node user-defined chart, you should set the chart option, [Agent].

Figure 7-25:Node User-Defined Chart



Active Service: 5
Request Rate: 15.87
Ave. Resp. Time: 0.35
Concurrent Users: 303.75
Hits: 599,312
Visitors: 46,370

Critical: 8
Error: 17,556
Warning: 20,505

The following table represents the field options that can be set.

Table 7-5: Node User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the JENNIFER agent ID.

The following table represents the parameter options that can be set.

Table 7-6: Node User-Defined Chart Parameter Options

Parameter option	Descriptions
FILTER_ITEM	If you want to view specific performance data only, you should enter them in this option. Use a comma as a separator for multiple items.
SHOW_CHART	Displays the number of active services for the JENNIFER agent on an equalizer chart. If you do not want to display this, you should set the SHOW_CHART parameter option to false.

Refer to the following table for the performance data numbers.

Table 7-7: Performance Data Item Number

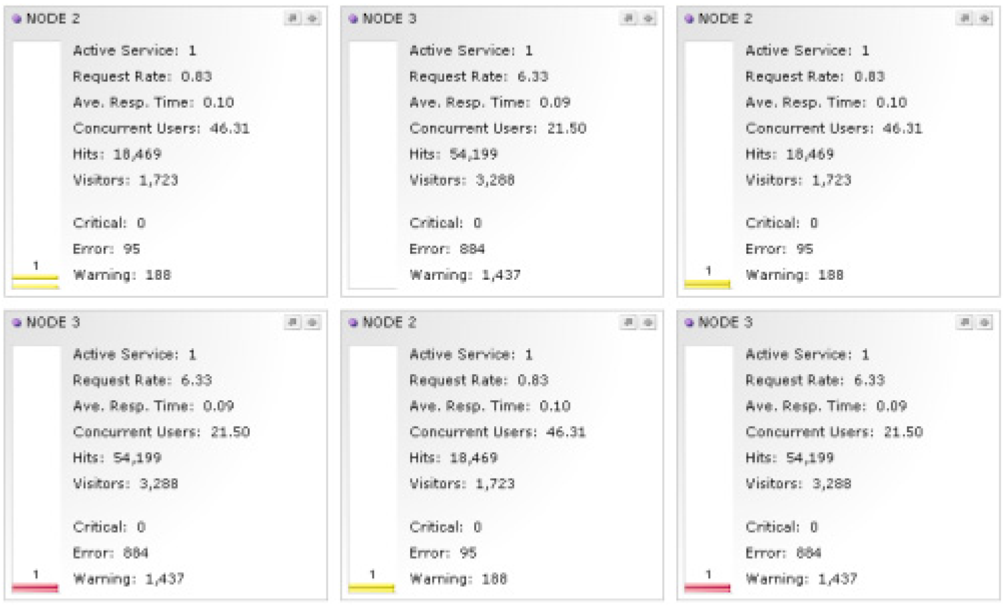
Item Number	Description
1	# of active service
2	Service rate
3	Average response time
4	# of concurrent user

Item Number	Description
5	# of hit
6	# of visit user
7	Critical
8	Error
9	Warning

7.3.5.4. Node Group

The node group user-defined chart provides multiple node charts at once.

Figure 7-26:Node Group User-Defined Chart



NODE 2
Active Service: 1
Request Rate: 0.83
Ave. Resp. Time: 0.10
Concurrent Users: 46.31
Hits: 18,469
Visitors: 1,723
Critical: 0
Error: 95
Warning: 188
1

NODE 3
Active Service: 1
Request Rate: 6.33
Ave. Resp. Time: 0.09
Concurrent Users: 21.50
Hits: 54,199
Visitors: 3,288
Critical: 0
Error: 884
Warning: 1,437

NODE 2
Active Service: 1
Request Rate: 0.83
Ave. Resp. Time: 0.10
Concurrent Users: 46.31
Hits: 18,469
Visitors: 1,723
Critical: 0
Error: 95
Warning: 188
1

NODE 3
Active Service: 1
Request Rate: 6.33
Ave. Resp. Time: 0.09
Concurrent Users: 21.50
Hits: 54,199
Visitors: 3,288
Critical: 0
Error: 884
Warning: 1,437
1

NODE 2
Active Service: 1
Request Rate: 0.83
Ave. Resp. Time: 0.10
Concurrent Users: 46.31
Hits: 18,469
Visitors: 1,723
Critical: 0
Error: 95
Warning: 188
1

NODE 3
Active Service: 1
Request Rate: 6.33
Ave. Resp. Time: 0.09
Concurrent Users: 21.50
Hits: 54,199
Visitors: 3,288
Critical: 0
Error: 884
Warning: 1,437
1

The following table represents the parameter options that can be set.

Table 7-8: Node Group User-Defined Chart Parameter Options

Parameter option	Descriptions
COL	Set column number.
ROW	Set row number.
NODE_LIST	Set the JENNIFER agent lists for node configuration by using comma[,] as a delimiter. If you have different domain, add your domain ID prior to the JENNIFER agent ID using colon[:] as a delimiter.

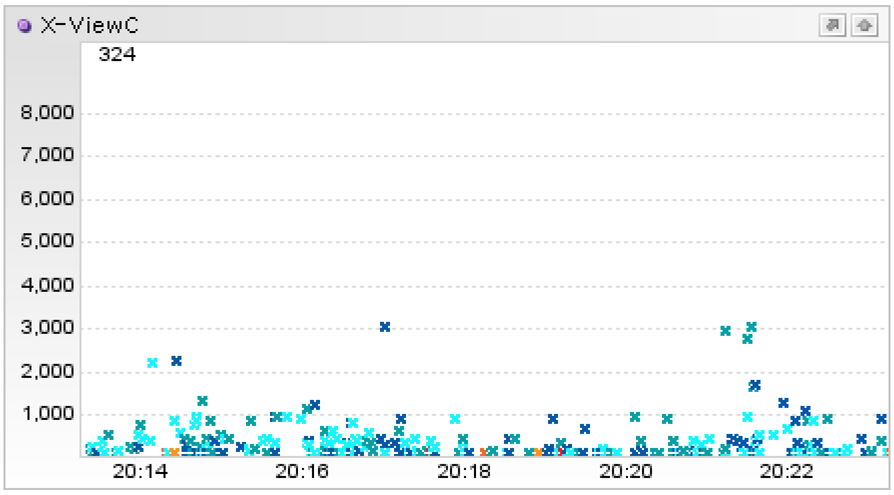
Also, you can use the FILTER_ITEM and SHOW_CHART option as like a node user-defined chart.

7.3.5.5. X-ViewC

Displays the data collected from REMON on the X-View chart.

Notice: Not all REMON data can be displayed on the X-ViewC chart. Only REMON data that has been properly set for the X-ViewC chart can be displayed.

Figure 7-27:X-ViewC User-Defined Chart



The field options that can be set are as follows;

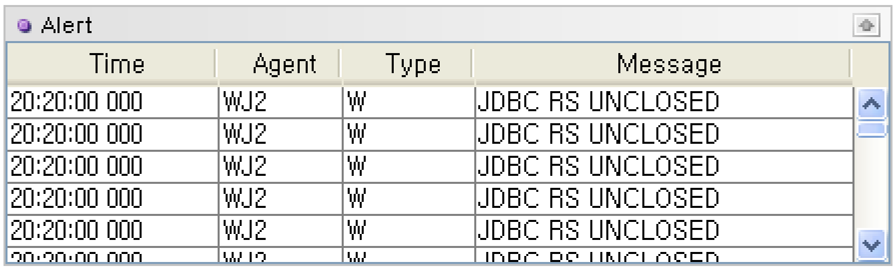
Table 7-9: X-ViewC User-Defined Chart Options.

Field option	Descriptions
Agent	If you do not set the agent, every X-ViewC data is displayed. However, if you want to see the X-ViewC data only collected by REMON agent, set this option. If you want to set more than 2 agents, use the comma[,] as as a delimiter.

7.3.5.6. Alerts

The alert user-defined chart displays alert messages.

Figure 7-28:Alert User-Defined Chart - Grid

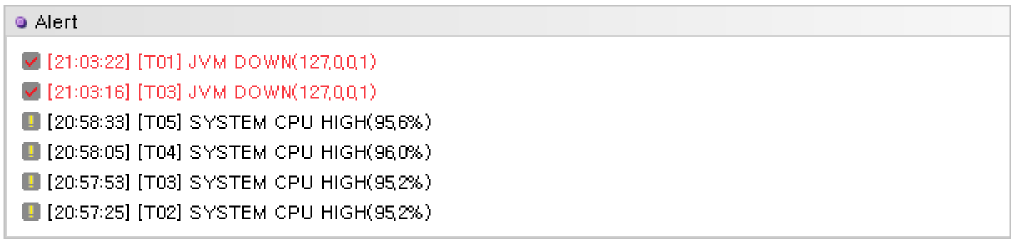


Alert

Time	Agent	Type	Message
20:20:00 000	WJ2	W	JDBC RS UNCLOSED
20:20:00 000	WJ2	W	JDBC RS UNCLOSED
20:20:00 000	WJ2	W	JDBC RS UNCLOSED
20:20:00 000	WJ2	W	JDBC RS UNCLOSED
20:20:00 000	WJ2	W	JDBC RS UNCLOSED
20:20:00 000	WJ2	W	JDBC RS UNCLOSED

If you set the IS_TEXT parameter option to true, it will be displayed in text instead of a grid.

Figure 7-29:Alert User-Defined Chart - Text



Alert

```
[21:03:22] [T01] JVM DOWN(127.0.0.1)
[21:03:16] [T03] JVM DOWN(127.0.0.1)
[20:58:33] [T05] SYSTEM CPU HIGH(95.6%)
[20:58:05] [T04] SYSTEM CPU HIGH(96.0%)
[20:57:53] [T03] SYSTEM CPU HIGH(95.2%)
[20:57:25] [T02] SYSTEM CPU HIGH(95.2%)
```

The parameter options that can be set are as follows.

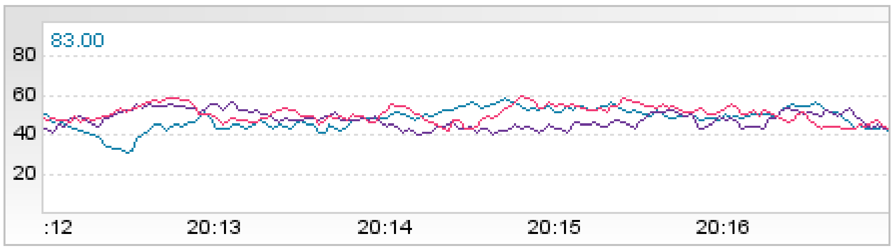
Table 7-10: Alert User-Defined Chart Parameter Options

Parameter option	Descriptions
IS_TEXT	If set to true, the alert user-defined chart will be displayed in text form.
VIEW_AGENTS	If you want to filter the JENNIFER agent with alerts, you should enter them in this option. Use a comma (separator) between each JENNIFER agent ID. This option is available only in a grid format.
FILTER_TYPE	Used to set the filtering condition for alert types. Critical, error and warning are expressed as C, E and W, respectively. If you want to display critical alerts only, then you should set FILTER_TYPE to C. If you want to display critical and error alerts only, you should enter C and E, separated by a comma. This is available only in a grid format.
FILTER_NAME	Used to set the filtering condition for alert messages. It supports LIKE searching. This is available only in a grid format.

7.3.5.7. LINE

The LINE user-defined chart displays the data collected from REMON on a runtime line chart.

Figure 7-30:LINE User-Defined Chart



The field options that can be set are as follows.

Table 7-11: LINE User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator(,). If not entered, the data type is used as its name.
Index	Specify whether or not to display the index.

The parameter options that can be set are as follows.

Table 7-12: LINE User-Defined Chart Parameter Options

Parameter option	Descriptions
LINE_WIDTH	Used to set the line thickness.
COLORS	Enter the line color. The RGB(255,255,255) format is used for colors. For more than two colors, you should use a / as a separator. For instance, if you want to use red and blue for the line color, then you should enter 255,0,0/0,0,255.
FORMAT	Enter the data format. The default is %,.0f.

7.3.5.8. TIME LINE

The TIME LINE user-defined chart displays the data collected from REMON for 24 hours on a line chart.

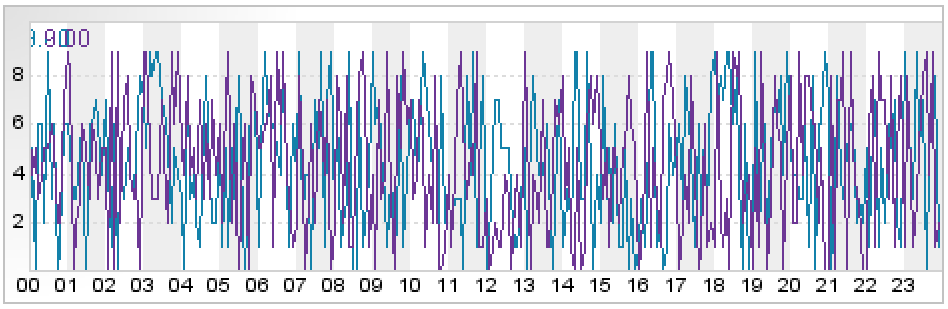


Figure 7-31:TIME LINE Uer-defined Chart

If you want to use the TIME LINE user-defined chart, write the REMON as follows:

```
packet.addField("today", new INT(1)); // Use the field name. Date
meaningless
packet.addField("yesterday", new INT(2));

for (int i = 0; i < 288; i++) {
  TYPE[] types = new TYPE[2];
  types[0] = new INT(random.nextInt(10)); // today
  types[1] = new INT(random.nextInt(10)); // yesterday
  packet.addAttachedRow(types);
}
```

The field options that can be set are as follows..

Table 7-13: TIME LINE User-Defined Chart Parameter Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator(,). If not entered, the data type is used as its name.
Index	Specify whether or not to display the index.

The parameter options that can be set are as follows.

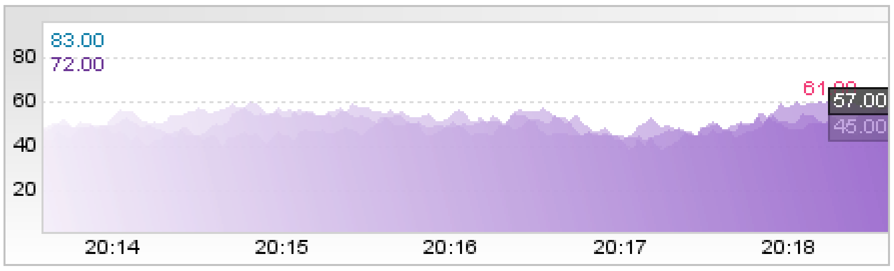
Table 7-14: TIME LINE User-Defined Chart Parameter Options

Parameter option	Descriptions
LINE_WIDTH	Used to set the line thickness.
COLORS	Enter the line color. The RGB(255,255,255) format is used for colors. For more than two colors, you should use a / as a separator. For instance, if you want to use red and blue for the line color, then you should enter 255,0,0/0,0,255.
FORMAT	Enter the data format. The default is %,.0f.

7.3.5.9. STACKED LINE

The STACKED LINE user-defined chart displays the data collected from REMON on a runtime area chart.

Figure 7-32:STACKED LINE User-Defined Chart



The field options that can be set are as follows.

Table 7-15: STACKED LINE User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator(,). If not entered, the data type is used as its name.
Index	Specify whether or not to display the index.

The parameter options that can be set are as follows.

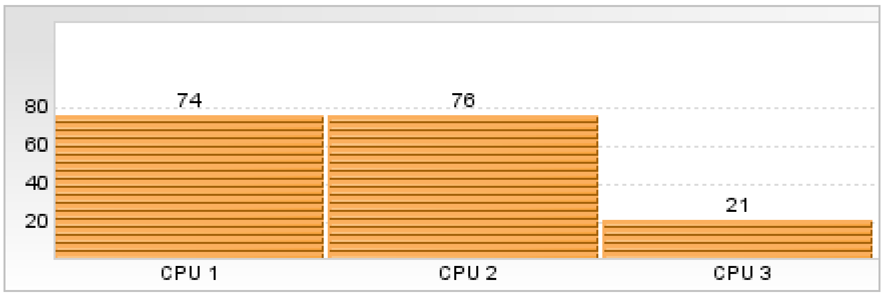
Table 7-16: STACKED LINE User-Defined Chart Parameter Options

Parameter option	Descriptions
FORMAT	Enter the data format. The default is %,.0f.

7.3.5.10. EQUALIZER

The EQUALIZER user-defined chart displays the data collected from REMON on an equalizer chart.

Figure 7-33:EQUALIZER User-Defined Chart



The field options that can be set are as follows.

Table 7-17: EQUALIZER User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator(,). If not entered, the data type is used as its name.

The parameter options that can be set are as follows.

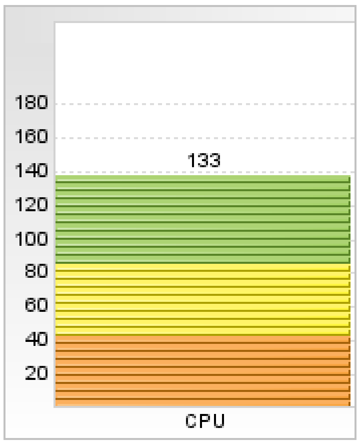
Table 7-18: EQUALIZER User-Defined Chart Parameter Options

Parameter options	Descriptions
COLORS	Enter the equalizer color. The RGB (255,255,255) format is used for colors. For more than two colors, you should use a / as a separator. For instance, if you want to use red and blue for the line color, then you should enter 255,0,0/0,0,255.
_COLORS	Enter the color for an inactive equalizer. The same format as for COLORS is used.
FORMAT	Enter the data format. The default is %,.0f.

7.3.5.11. STACKED EQUALIZER

The STACKED EQUALIZER user-defined chart displays the data collected from REMON on an equalizer chart.

Figure 7-34:STACKED EQUALIZER User-Defined Chart



The field options that can be set are as follows.

Table 7-19: STACKED EQUALIZER User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.

Field option	Descriptions
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator[,]. If not entered, the data type is used as its name.

The parameter options that can be set are as follows.

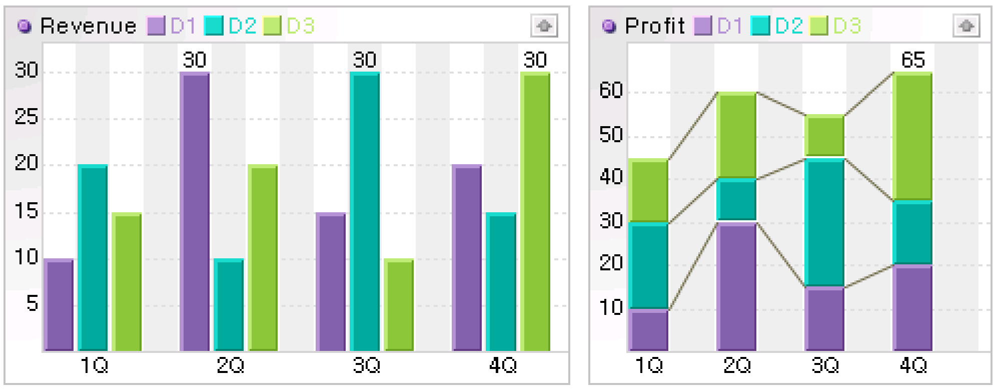
Table 7-20: STACKED EQUALIZER User-Defined Chart Parameter Options

Parameter options	Descriptions
FORMAT	Enter the data format. The default is %,.0f.

7.3.5.12. BAR

The BAR user-defined chart displays the data collected from REMON on a bar chart.

Figure 7-35:BAR User-Defined Chart



The field options that can be set are as follows.

Table 7-21: BAR User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator[,]. If not entered, the data type is used as its name.

Table 7-21: BAR User-Defined Chart Field Options

Field option	Descriptions
Maximum value	Input the maximum value.

The parameter options that can be set are as follows.

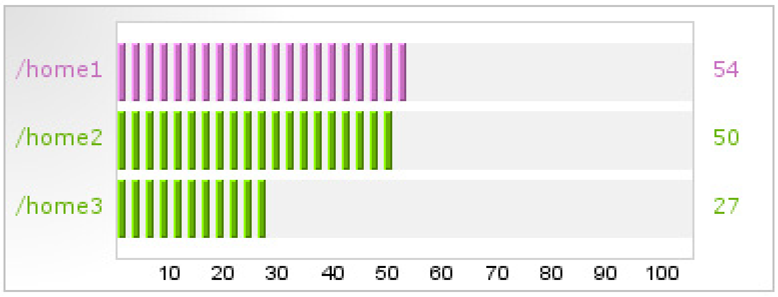
Table 7-22: BAR User-Defined Chart Parameter Options

Parameter option	Descriptions
GROUP_SIZE	If you want to define the data as a group, input the group size. If not, the data is binded as a group.
ORIENTATION	After defining a data group, to input HORIZONTAL makes the bar horizontal and VERTICAL makes vertical. The default is HORIZONTAL.
GROUP_LABEL	Use a comma as a separator for the group name. If entered, all the title and the name will be displayed as a group lable of the chart.
SHOW_LINE	This option applied only when you set ORIENTATION to VERTICAL. If you set to true, the bar will be linked as a line. The default is true.
SHOW_VALUE	If you set to true, every value of the bar will be displayed. The default is false.

7.3.5.13. HORIZONTAL BAR

The HORIZONTAL BAR user-defined chart displays the data collected from REMON on a horizontal bar chart. This chart is most commonly used to monitor the disk utilization.

Figure 7-36:HORIZONTAL BAR User-Defined Chart



The field options that can be set are as follows.

Table 7-23: HORIZONTAL BAR User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.

Table 7-23: HORIZONTAL BAR User-Defined Chart Field Options

Field option	Descriptions
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator(,). If not entered, the data type is used as its name.
Maximum value	Input the maximum value.

The parameter options that can be set are as follows.

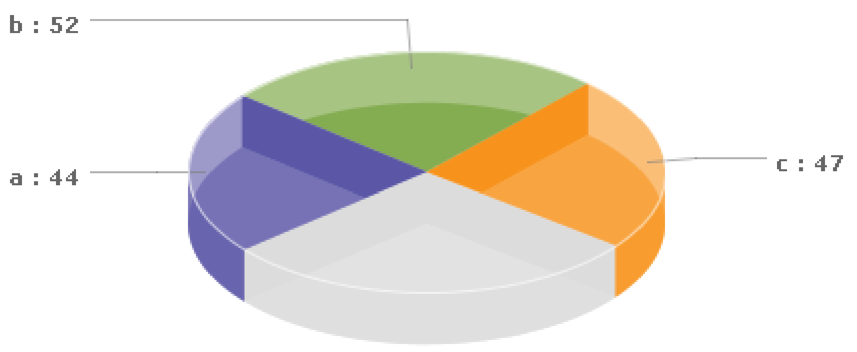
Table 7-24: HORIZONTAL BAR User-Defined Chart Parameter Options

Parameter option	Descriptions
IS_SIMPLE	If you set true, it will displayed as a simple bar on the chart.
RANGES	Set the different colors depending on the value. Use a comma as a separator and start to set from the large figure like 80,60,40. The default is 90, 70 for the JENNIFER 3.2 disk chart.
COLORS	Set the different colors depending on the value. Add one more option than RANGES. Use a slash[/] as a separator to set RGB color such as 255,0,0/ 0,255,0/0,0,255.The default is 255,35,15/198,111,190/101,178,4 for the JENNIFER 3.2 disk chart.
FORMAT	Enter the figure format on the right. The default is %,.0f. If you enter % on the chart like the disk chart in JENNIFER 3.2, set to %,.0f%%.

7.3.5.14. PIE

The PIE user-defined chart displays the data collected from REMON on a pie chart. The existing meter chart is integrated into this pie chart.

Figure 7-37:PIE User-Defined Chart



The field options that can be set are as follows.

Table 7-25: PIE User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator[,]. If not entered, the data type is used as its name.
Maximum value	Input the maximum value. If you do not input this value, the sum of data becomes the maximum value.

The parameter options that can be set are as follows.

Table 7-26: PIE User-Defined Chart Parameter Options

Parameter option	Descriptions
IS_3D	Set to true and then the chart is displayed as a 3D type. The default is false.
TOTAL_ANGLE	Set the total angle of the PIE chart such as 360, 180, etc.
START_ANGLE	Turn the data on a clockwise rotation with first angle. The default is 220. (down: 270, left: 180, up: 90 and right: 0)
DONUT_RADIUS	To display the PIE chart into donut type, set the value from 0.1 to 0.9. This value indicates the proportion of the donut size to the total diameter.

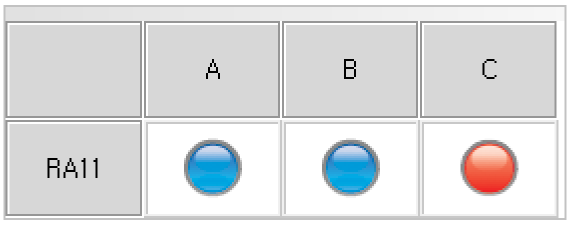
Table 7-26: PIE User-Defined Chart Parameter Options

Parameter option	Descriptions
IS_FILL	Set to true and then the chart is filled than MAX_VALUE. The default is false. In case of the 3D PIE chart, if you set DONUT_RADIUS, IS_FILL is always set to true.
IS_FIXED	Not in case of 3D PIE chart, set to true to make the figure fixed. The default is false.
IS_PERCENT	Set to true to display the maximum value as a percentage. The default is false.
COLORS	Set the each PIE color through RGB like 255,0,0/0,255,0/0,0,255, etc.
FORMAT	Enter the data format. The default is %2$s : %1$,.0f. The 2$ means the data name and 1$ indicates data value.
IS_SORT	Set to true in order to arrange the pie based on the data size. If the data isze is zero(0), the pie does not displayed. The default is false.

7.3.5.15. ON/OFF CHECK

The ON/OFF CHECK user-defined chart displays the data collected from REMON on the alarm lamp.

Figure 7-38:ON/OFF CHECK User-Defined Chart



	A	B	C
RA11	🔵	🔵	🔴

The field options that can be set are as follows.

Table 7-27: ON/OFF CHECK User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator[,]. If not entered, the data type is used as its name.

The parameter options that can be set are as follows.

Table 7-28: ON/OFF CHECK User-Defined Chart Parameter Options

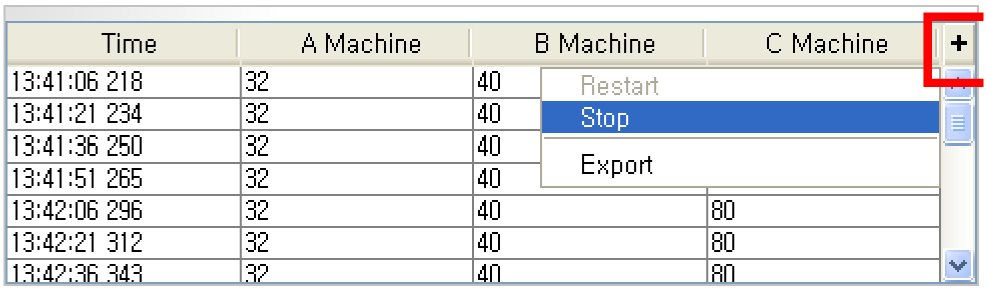
Parameter option	Descriptions
FORMAT	Enter the data format. The default is %,.0f.
SHOW_ICON	Specify whether to display the alarm lamp icon.
SHOW_VALUE	Specify whether to display the value..
PADDING	Set the distance between the lamp and the chart border. Using a comma[,] as a separator, set the upper, lower, right and left margins.
WARNING_VALUE_OVER_THAN	If the value is exceeded, the lamp will turn red.
WARNING_VALUE_LOWER_THAN	If lower than the value, then the lamp will turn red.

Notice: If both the WARNING_VALUE_OVER_THAN and WARNING_VALUE_LOWER_THAN options are not set, the lamp will turn red when the value is zero.

7.3.5.16. TABLE

The TABLE user-defined chart displays the data collected from REMON on a grid.

Figure 7-39:TABLE User-Defined Chart



Time	A Machine	B Machine	C Machine
13:41:06 218	32	40	
13:41:21 234	32	40	
13:41:36 250	32	40	
13:41:51 265	32	40	
13:42:06 296	32	40	80
13:42:21 312	32	40	80
13:42:36 343	32	40	80

The field options that can be set are as follows.

Table 7-29: TABLE User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.

Field option	Descriptions
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator[,]. If not entered, the data type is used as its name.

The parameter options that can be set are as follows.

Table 7-30: TABLE User-Defined Chart Parameter Options

Parameter options	Descriptions
FORMAT	Enter the data format. The default is %,.0f.
SHOW_TIME	The data collected time is displayed in the first colum as a default value. If you set this option to false, the time dees not displyed.
COLUMN_WIDTHS	Set the each colum size using a comma[,] as a separator. Unit is pixel.
ADD_MODE	The default is BOTTOM and new data is added on the bottem area. If you set this option to TOP, the new data is added on the top area.

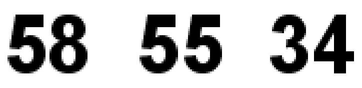
You can use the some features using [+] icon on the upper area of TABLE user-defined chart.

- Stop- Updates the newly added data automatically. You can stop this updata process.
- Restart- Restart the auto update task.
- Export - Export the data as s CSV type.

7.3.5.17. NUMBER

The NUMBER user-defined chart displays the data collected from REMON as numbers.

Figure 7-40:NUMBER User-Defined Chart



The field options that can be set are as follows.

Table 7-31: NUMBER User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.

The parameter options that can be set are as follows.

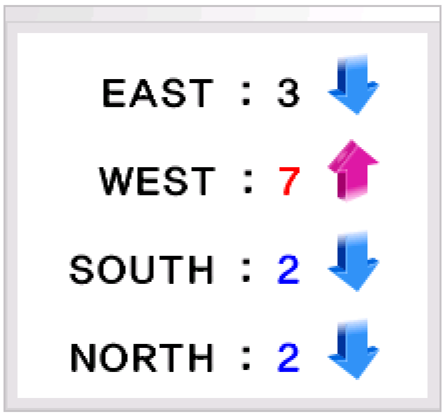
Table 7-32: NUMBER Use-Defined Chart Parameter Options

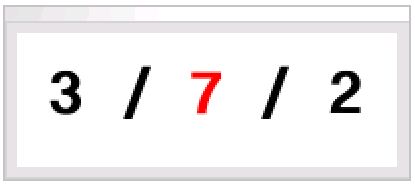
Parameter options	Descriptions
FORMAT	Enter the data format. The default is %,.0f.
FONT	Set the font name, style and size. Enter the font name, style and size, separated by commas. For style, 0, 1 and 2 stand for plain, bold and italics, respectively. If you enter arial, 1, 11 then a bold 11-point arial font is used.
COLOR	Enter the color. The color format is RGB(255,255,255).
TYPE	Set to digital or plain. The certain options such as EFFECT, EFFECT_COLOR, IS_REFLECTION, etc is adjusted when you set these types.
TEXT_ALIGN	Set the text alignment (right and left). There are right, left and center type. The default is center.
EFFECT	Set the glow and shadow.
EFFECT_COLOR	Set the color effect through RGB(255,0,0) type.
IS_REFLECTION	Set true and you can see the reflection effect. The default is true.

7.3.5.18. TEXT

The TEXT user-defined chart displays the data collected from REMON as text.

Figure 7-41:TEXT User-defined Chart





The field options that can be set are as follows.

Table 7-33: TEXT User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator[,]. If not entered, the data type is used as its name.

The parameter options that can be set are as follows.

Table 7-34: TEXT User-Defined Chart Parameter Options

Parameter option	Descriptions
FORMAT	Enter the data format. The default is %,.0f.
TEXT_PATTERN	Set the text format. The default is ${name} : ${value}. In this case, every data has the text pattern. You can use the arbitrary stings out from ${} . When you output the only feild, use ${fieldname.name} and ${fieldname.value} together. But, every data is not adjusted in this case. For instance, you can set as ${F1.value} / ${F2.value} / ${F3.value}.
TEXT_ALIGN	Set the text alignment (right and left). There are right, left and center type. The default is left.
LINE_HEIGHT	If you do not set the line interval, it is set by font size.
PADDING	Set the margine(right or left). Unit is pixel.

Table 7-34: TEXT User-Defined Chart Parameter Options

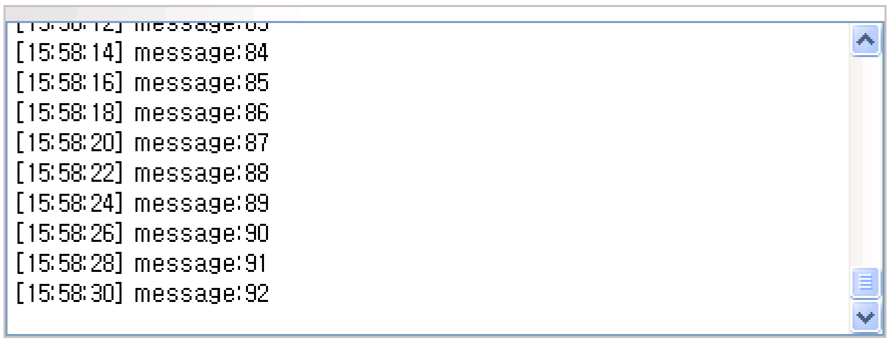
Parameter option	Descriptions
FONT	Set the font name, style and size. Use a comma as a separator when you input the font name, style and size. For style, 0 is PLAIN, 1 is BOLD and 2 is ITALIC. Therefore, Arial,1,11 means font style is Arial with bold and 11 font size.
COLOR	Enter the color. The color format is RGB(255,255,255).
MAX_COLOR	Enter the color for the maximum value. The format to set a color is RGB(255,255,255). The defalut is the one set by COLOR option.
MIN_COLOR	Enter the color for the minimum value. The format to set a color is RGB(255,255,255). The defalut is the one set by COLOR option.
SHOW_FRAME	Choose to show the frame. The default is true.
SHOW_IMAGE	Choose to output the up/down image. The default is false.
DELIMIT	Choose the delimit value of the up/down image. The default is 0.

7.3.5.19. TEXT AREA

The TEXT AREA user-defined chart displays the data collected from REMON as text.

Notice: You can use this chart in case the data of REMON is stream type.

Figure 7-42:TEXT AREA User-Defined Chart



```
[15:58:14] message:84
[15:58:16] message:85
[15:58:18] message:86
[15:58:20] message:87
[15:58:22] message:88
[15:58:24] message:89
[15:58:26] message:90
[15:58:28] message:91
[15:58:30] message:92
```

The field options that can be set are as follows.

Table 7-35: TEXT AREA User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.

The parameter options that can be set are as follows.

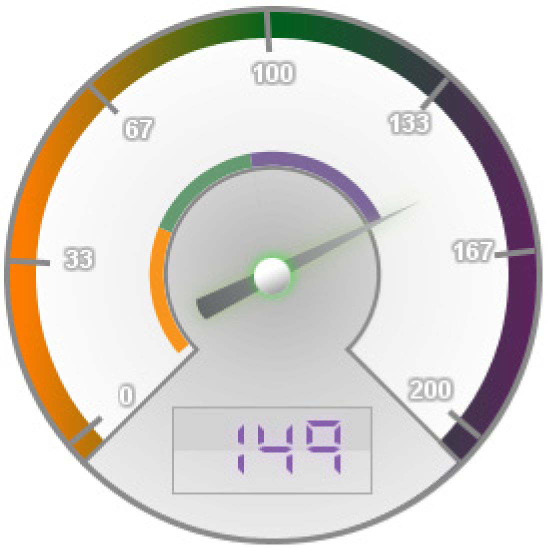
Table 7-36: TEXT AREA User-Defined Chart Parameter Options

Parameter option	Descriptions
IS_TAIL	Set to add the new text on the lower area or upper area. The default is true.
SHOW_TIME	Set to display time.The default is true.
MAX_LENGTH	Limit the number of text. The default is 5000.
COLOR	Enter the text color.
COLOR_PATTERN	Set the color through the text pattern based on the message format as follows; ERROR=255,0,0;WARN=0,0,255

7.3.5.20. GAUGE1

The GAUGE1 user-defined chart displays the data collected from REMON on a gauge type chart.

Figure 7-43:GAUGE1 User-Defined Chart



The field options that can be set are as follows.

Table 7-37: GAUGE1 User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.

Table 7-37: GAUGE1 User-Defined Chart Field Options

Field option	Descriptions
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Maximum value	Input the maximum value. If you do not input this value, the sum of data becomes the maximum value.

The parameter options that can be set are as follows.

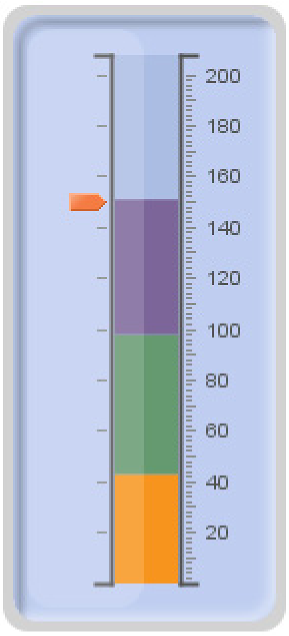
Table 7-38: GAUGE1 User-Defined Chart Parameter Options

Parameter option	Descriptions
FORMAT	Enter the data format. The default is %,.0f.
IS_SHADOW	Set true and the drop shadow effect is adjusted.
IS_PERCENT	Set to true to display the maximum value as a percentage. The default is false.
RANGES	When you select only one data, you can set the margin value. Input the margin value using comma[,] as 1,3,5,9.
COLORS	Set the color of the bar and line through RGB. Input the value as follows: 255,0,0/0,255,0/0,0,255.

7.3.5.21. GAUGE2

The GAUGE2 user-defined chart displays the data collected from REMON on a gauge type chart.

Figure 7-44:GAUGE2 User-Defined Chart



The field options that can be set are as follows.

Table 7-39: GAUGE2 User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Maximum value	Input the maximum value. If you do not input this value, the sum of data becomes the maximum value.

The parameter options that can be set are as follows.

Table 7-40: GAUGE2 User-Defined Chart Parameter Options

Parameter option	Descriptions
FORMAT	Enter the data format. The default is %,.0f.
IS_PERCENT	Set to true to display the maximum value as a percentage. The default is false.
RANGES	When you select only one data, you can set the margin value. Input the margin value using comma[,] as 1,3,5,9.

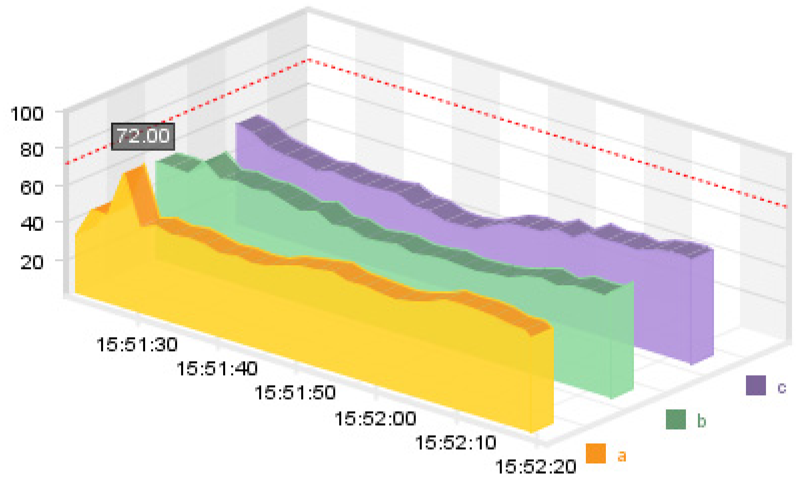
Table 7-40: GAUGE2 User-Defined Chart Parameter Options

Parameter option	Descriptions
COLORS	Set the color of the bar and line through RGB. Input the value as follows: 255,0,0/0,255,0/0,0,255.

7.3.5.22. BOX LINE

The BOX LINE user-defined chart displays the data collected from REMON on a 3D line chart.

Figure 7-45:BOX LINE User-Defined Chart



The field options that can be set are as follows.

Table 7-41: BOX LINE User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator(,). If not entered, the data type is used as its name.
Maximum value	Input the maximum value.

The parameter options that can be set are as follows.

Table 7-42: BOX LINE User-Defined Chart Parameter Options

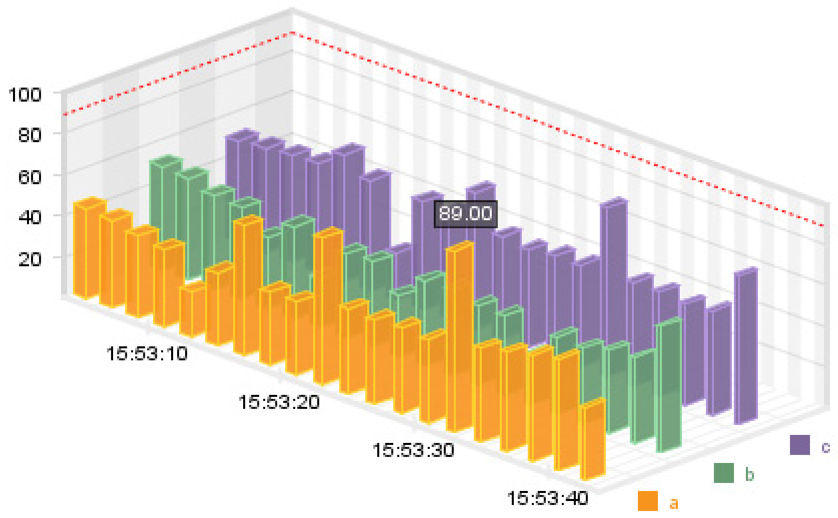
Parameter option	Descriptions
WIDTH_RATE	Set the width rate(from 0.1 to 0.9) of the left side. This is the proprotion of the left side to the total width.
HEIGHT_RATE	Set the hight rate (from 0.1 to 0.9) of the upper side. This is the proprotion of the hight to the upper side.
START_ANGLE	Set the angle(from 10 to 90) of the box.
IS_FILL	If you set to true, the lower area is filled. The default is false.
COLORS	Set the color of the bar and line through RGB. Input the value as follows: 255,0,0/0,255,0/0,0,255.

Click the certain area of the BOX LINE chart and drag the mouse point to the right or left area. Then, the START_ANGLE option is adjusted. In addition, after clicking the top point, drag the mouse point to the right or left area. Then the WIDTH_RATE is changed depends on your movement.

7.3.5.23. BOX BAR

The BOX BAR user-defined chart displays the data collected from REMON on a 3D bar chart.

Figure 7-46:BOX BAR User-Defined Chart



The field options that can be set are as follows.

Table 7-43: BOX BAR User-Defined Chart Field Options

Field option	Descriptions
Agent	Enter the REMON agent.
Script	Enter the REMON script.
Data	Enter the data type collected from REMON. Use a comma as a separator for multiple entries. If not entered, all the data types will be displayed on a line chart.
Display	Used to assign an arbitrary name to the data type. You should enter an arbitrary name with a separator[,]. If not entered, the data type is used as its name.
Maximum value	Input the maximum value.

The parameter options that can be set are as follows.

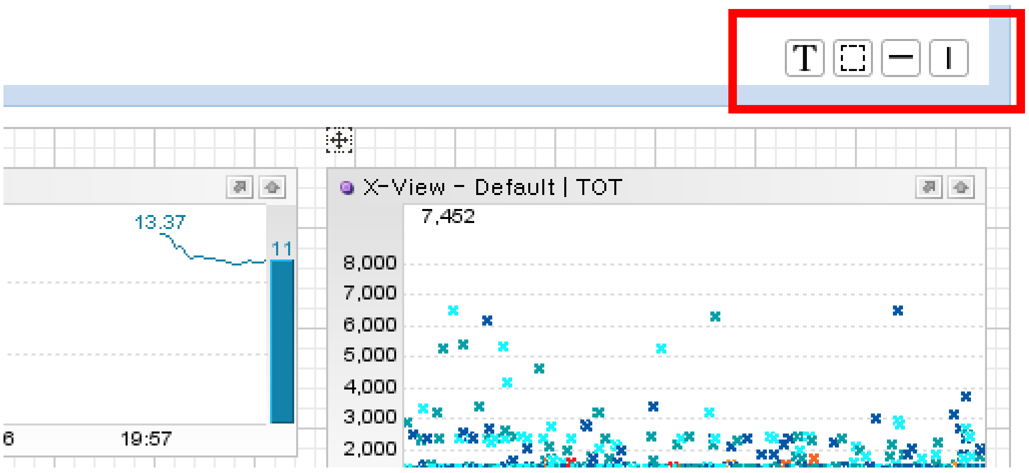
Table 7-44: BOX BAR User-Defined Chart Parameter Options

Parameter option	Descriptions
WIDTH_RATE	Set the width rate(from 0.1 to 0.9) of the left side. This is the proprotion of the left side to the total width.
HEIGHT_RATE	Set the hight rate (from 0.1 to 0.9) of the upper side. This is the proprotion of the hight to the upper side.
START_ANGLE	Set the angle(from 10 to 90) of the box.
COLORS	Set the color of the bar and line through RGB. Input the value as follows: 255,0,0/0,255,0/0,0,255.

Click the certain area of the BOX LINE chart and drag the mouse point to the right or left area. Then, the START_ANGLE option is adjusted. In addition, after clicking the top point, drag the mouse point to the right or left area. Then the WIDTH_RATE is changed depends on your movement.

7.3.6. Using Text, Boxes and Lines

Figure 7-47:Using Text, Boxes and Lines



Notice: The charts are implemented in Java applets, while the icons are in HTML tags. Due to technical limitations, an HTML tag cannot be placed over a Java applet chart. If a chart overlaps text, boxes or lines, some icons (resize, move) may be hidden by other Java applet charts. This should be considered when designing a user-defined dashboard.

7.3.6.1. Text

If you want to add text to the user-defined dashboard, click the text icon in the lower right corner of the chart selection area. The method for adding text to the dashboard is same as for general charts.

The field options that can be set are as follows.

Table 7-45: Text Field Option

Field option	Description
Text	Enter the text.
Type	Set the text type such as title01, title02, etc. In this case, the defined-text format appears and you can not use the below options.
Text alignment	Set the left/right text alignment.
Font size	Set the font size.
Bold	Specify whether to use bold text.
Font color	Set the text color.
Line color	Set the color for the edge of text.

Table 7-45: Text Field Option

Field option	Description
Line thickness	Set the thickness of the edge of text.
Background color	Set the background color for text.

7.3.6.2. Box

If you want to add a box to the user-defined dashboard, click the box icon in the lower right corner of the chart selection area. The method for adding boxes in the dashboard is same as that for general charts.

The field options that can be set are as follows.

Table 7-46: Box Field Options

Field options	Descriptions
Line color	Set the color for the edge line of the box.
Line thickness	Set the thickness of the edge line of the box.
Background color	Set the background color for the box.

7.3.6.3. Horizontal Lines

If you want to add a horizontal line to the user-defined dashboard, click the horizontal line icon in the lower right corner of the chart selection area. The method for adding horizontal lines in the dashboard is same as for general charts.

The field options that can be set are as follows.

Table 7-47: Horizontal Line Field Options

Field options	Descriptions
Line color	Set the color for the horizontal line.
Line thickness	Set the thickness of the horizontal line.

7.3.6.4. Vertical Lines

If you want to add a vertical line to the user-defined dashboard, click the vertical line icon in the lower right corner of the chart selection area. The method for adding vertical lines in the dashboard is same as for general charts.

The field options that can be set are as follows.

Table 7-48: Vertical Line Field Options

Field options	Descriptions
Line color	Set the color for the vertical line.
Line thickness	Set the thickness of the vertical line.

Warning: The charts are implemented in Java applets, while the icons are in HTML tags. Due to technical limitations, an HTML tag cannot be placed over a Java applet chart. If a chart overlaps text, boxes or lines, some icons (resize, move) may be hidden by other Java applet charts. This should be considered when designing a user-defined dashboard.

7.3.7. Import & Export

You can export and import a user-defined dashboard to and from an XML file.

If you want to export the user-defined dashboard to an XML file, click the **[Export]** button in the lower right corner of the editing screen. The XML file is encoded by UTF-8, and its structure is as follows.

The top tag is charts. This charts tag includes the version attribute representing the JENNIFER server version, and the exportDate attribute representing the time of export.

```
<?xml version="1.0" encoding="UTF-8"?>
<charts version='4.0' exportDate='20080716/214809'>


</charts>
```

The charts tag has multiple lower rank tags representing individual charts.

```
<chart chartId='user.recent'
       left='358' top='9' width='300' height='80'>
</chart>
```

The chart tag's attributes have the following definitions:

- chartId - Chart type
- left - The X coordinate for the chart. The closer to zero, the further to the left the chart is located.
- top - The Y coordinate for the chart. The closer to zero, the higher the chart is located.
- width - Chart width

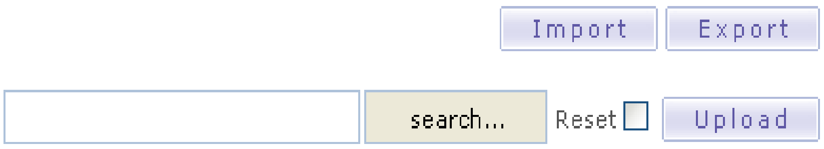
- height - Chart height

The chart tag describes the options for individual charts using the lower-ranked config tags. In this case, various options are separated by the separator (|).

```
<chart>
    <config>SERVER=ALL_EXCEPT_TOT|MARGIN_LEFT=32|TITLE=MEMORY</config>
</chart>
```

If you want to import an XML file, click the [Import] button at the bottom of the user-defined dashboard edit screen to launch the upload form.

Figure 7-48:Import Screen



Notice: If the initialization box is checked, all of the existing charts in the user-defined dashboard screen will be deleted, and the chart from the XML file will be opened. If the initialization box is not checked, the existing charts will not be deleted, and the chart in the XML file will be opened.

7.3.8. Setting the Background Image

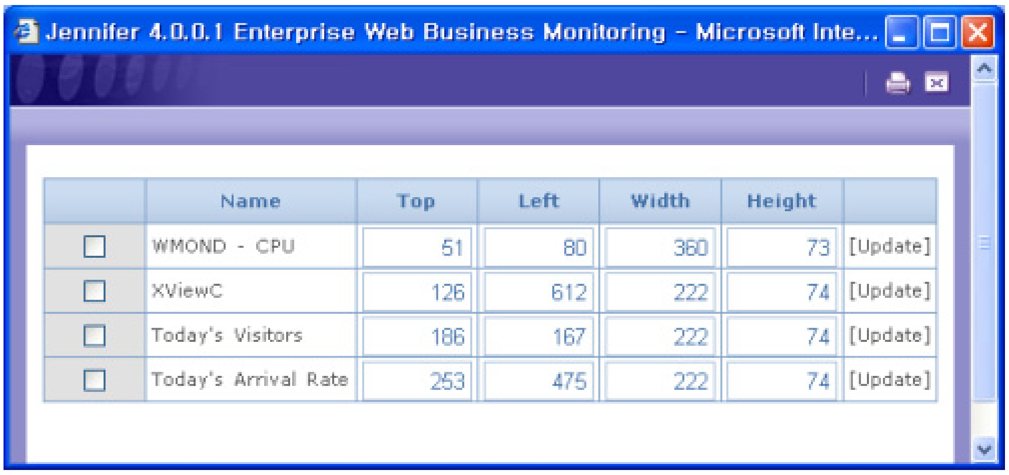
If you want to set the background image on the user-defined dashboard, use the uplaod form by clicking the **[Config]**. If you select the initialize check box and save, the background image is deleted.

7.3.9. Adjusting the Chart Location and Size

In the user-defined dashboard screen, you can move individual charts and adjust their sizes as well. You should click the [Change] button in the lower right of the user-defined dashboard screen to launch a pop-up window.

Notice: This task should be done in the user-defined dashboard screen, not the user-defined dashboard editing screen.

Figure 7-49:Pop-up Window for adjusting Chart Location and Size


Jennifer 4.0.0.1 Enterprise Web Business Monitoring - Microsoft Inte...

	Name	Top	Left	Width	Height	
☐	WMOND - CPU	51	80	360	73	[Update]
☐	XViewC	126	612	222	74	[Update]
☐	Today's Visitors	186	167	222	74	[Update]
☐	Today's Arrival Rate	253	475	222	74	[Update]

In the pop-up window for adjusting chart location and size, the locations and sizes of all charts, text, boxes and lines constituting the user-defined dashboard are displayed in a table. After changing the specific chart location or size, you should click the [Change] button or the [Enter] key to reflect the changes.

Table 7-49: Chart Location and Size Fields

Field	Descriptions
Top	The Y coordinate for the chart. The closer to zero, the higher the chart is located.
Left	The X coordinate for the chart. The closer to zero, the further to the left the chart is.
Width	The chart width
Height	The chart height

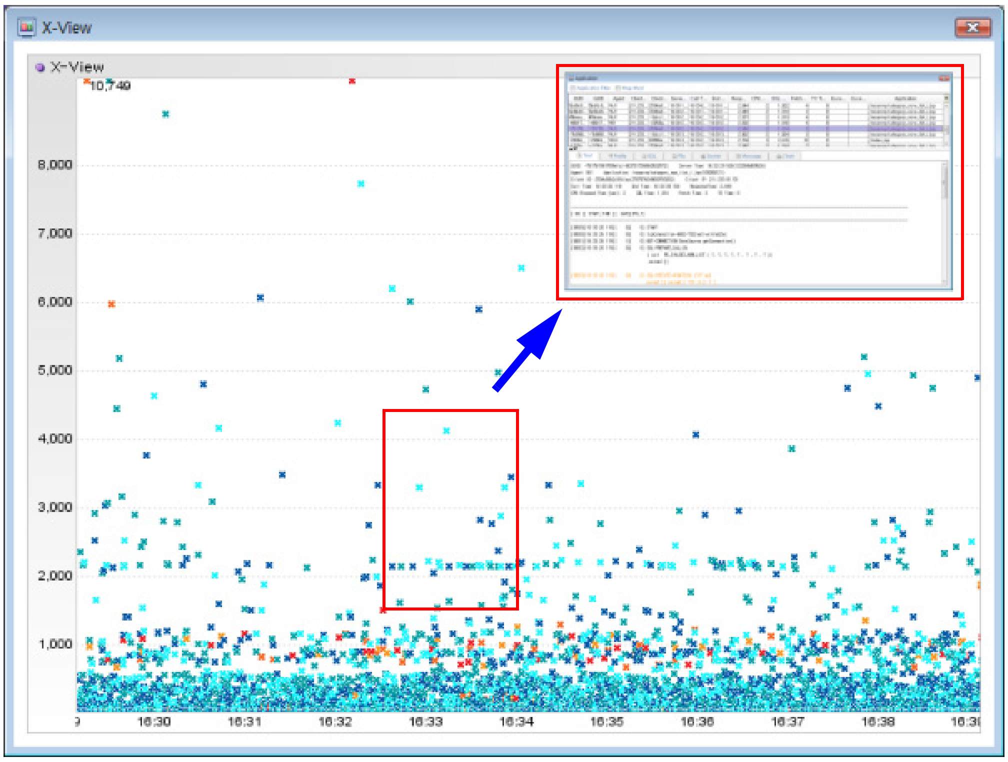
If you click the checkbox for a certain chart in the pop-up window for adjusting the chart location and size, the chart will be displayed as dotted lines in the user-defined dash-board screen.

8 X-View & Profiling

8.1. Response Time Distribution Graph and X-View

A response time distribution graph is a graph that plots the end times of individual transactions on the X-axis, and the response times on the Y-axis. Each point represents a transaction, and by selecting a point you can inquire about the details of its execution (profile). In JENNIFER, the response time distribution graph is called XView or X-View. .

Figure 8-1:Sample Graph of Response Time



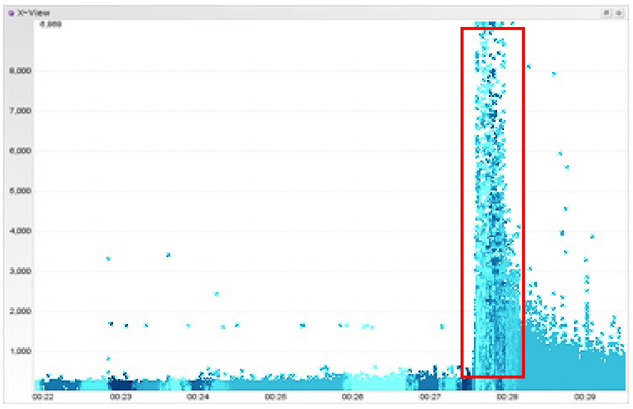
8.1.1. Response Time Distribution Patterns

The response time for a transaction can be affected by back-end performance problems. For this reason, abnormal performance is sometimes manifested in abnormal response patterns. In this section, This chapter describes patterns that you should be aware of in order to properly identify such problems.

8.1.1.1. Simple Congestion

This pattern can be observed when the event or the popular product ticket system is instantaneously crowded by traffic.

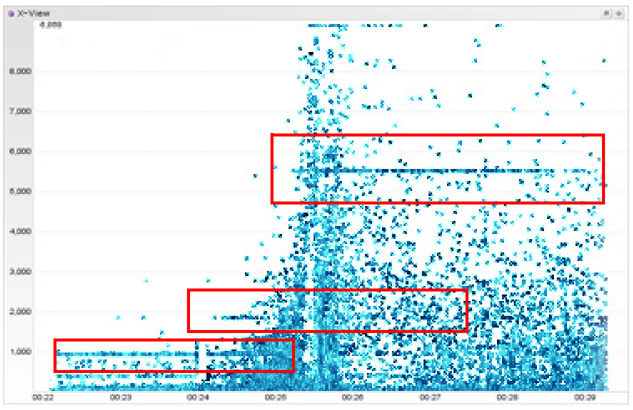
Figure 8-2:Simple Congestion



8.1.1.2. Steamed Rice Cake Effect

This pattern can be observed when the system fails to recover after an initial surge of service requests. The distribution following the congestion has a different pattern than simple congestion. Unlike simple congestion, there are many horizontal lines. Prior to the initial surge, the number of horizontal lines increases, and after the congestion, a significant number of horizontal lines can be found in the middle of the graph. We call it the ‘streamed rice cake’ effect, since the layer formed in the graph resembles a steamed rice cake. The multiple lines form layers, and the number of these horizontal lines increases as the load increases.

Figure 8-3:Steamed Rice Cake Effect



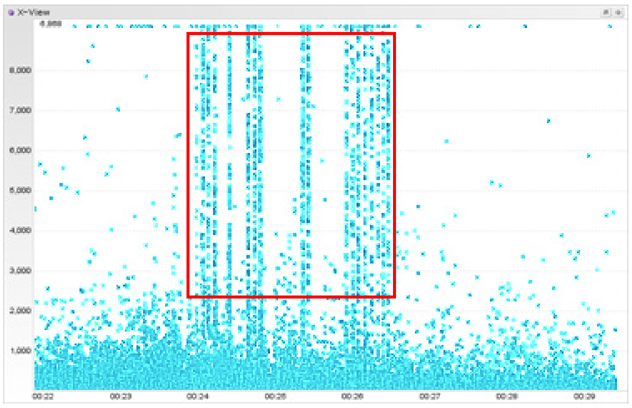
Notice: The appearance of one or two horizontal lines does not constitute a steamed rice cake effect.

The steamed rice cake effect is a phenomenon that occurs when the transaction repeatedly waits and attempts to acquire certain resource. In other words, if there is no resource for the transaction to acquire, then it will wait, and attempt again to acquire the resource, a cycle that is repeated continually. As a result, multiple layers of response time are generated. Therefore, as the load increases, the scarcity of resource occurs frequently. As a result, horizontal lines with steps equal to the standby time are formed in the graph.

8.1.1.3. Waterfall Effect

The waterfall effect refers to repetitive vertical lines in the graph. This phenomenon can be observed when a certain resource is released as it has reached its limit, and all of the related transactions are terminated simultaneously. The waterfall effect can occur when a certain resource is very sensitive to the limit condition.

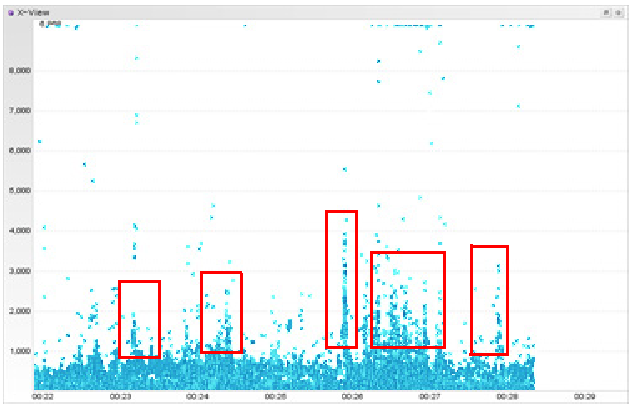
Figure 8-4:Waterfall Effect



8.1.1.4. Waterdrop Effect

We refer to this phenomenon as the 'waterdrop' effect, because the distribution of many spots on the graph resembles waterdrops bouncing off the ground on a rainy day. As shown in the graph, many rough and short vertical lines are formed in the graph. This can occur when the bottleneck effect of the system is not clearly displayed due to a small amount of load. If the load increases, then the bottleneck effect can have a severely impact on the system.

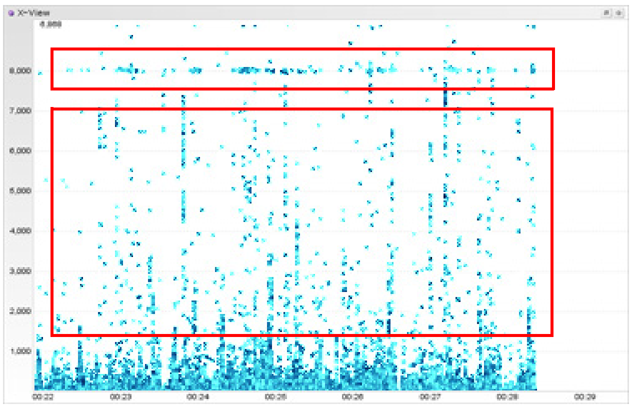
Figure 8-5:Waterdrop Effect



8.1.1.5. Matrix Effect

When many spots flow downward on the graph , resembling a visual effect that was frequently used in the movie The Matrix, we call it the matrix effect. This occurrs when brief but frequent lock effects occur over all of the transactions. Each lock affects several transactions, which are is usually located in arbitrary application services. In general, this effect can occur when the isolation level of the database or entity bean is very high.

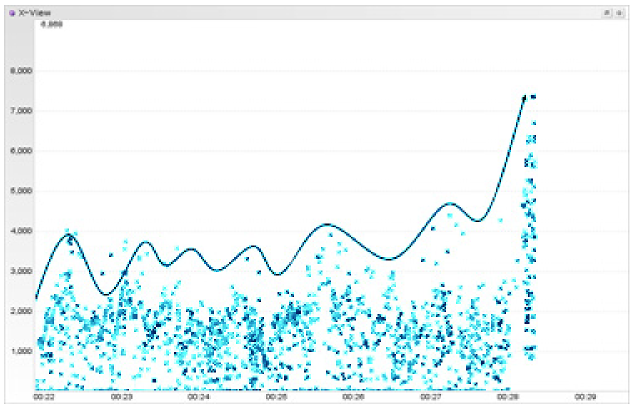
Figure 8-6:Matrix Effect



8.1.1.6. Wave Effect

When the distribution of transaction response times forms the shape of a wave, we call it the wave effect. It can occur when the amount of service requests fluctuates, but it is also likely to occur due to a scarcity of resources to process the transaction. In particular, it is likely to occur when the CPU usage of the external system connected with the application server reaches 100%.

Figure 8-7:Wave Effect



8.2. X-View Data

8.2.1. X-View Data Organization

The transaction related data that the JENNIFER server collects is called the X-view data, and it can be further classified into transaction data and profile data.

- The transaction data is composed of an application name, a call time and a duration.
- The profile data shows which method is invoked, which SQL is executed and which file or socket is used while handling the transaction.

The transaction data is organized as follows.

Table 8-1: Transaction Data

Item	Descriptions
Domain	Domain name of JENNIFER server
UUID	Unique transaction ID
GUID	A global key to group multiple transactions
Agent	JENNIFER agent IT of the Java application in which the transaction is executed
Client IP	IP address of the client computer requesting the transaction
Client ID	Unique client ID, managed by HTTP cookies
User ID	User ID for tasking and should be set separately.

Table 8-1: Transaction Data

Item	Descriptions
Server time	The time at which the JENNIFER server collects the transaction data. Based on the JENNIFER server time.
Call time	The beginning time of the transaction. Based on the time of the Java application in which the JENNIFER agent is installed.
End time	The end time of the transaction. Based on the time of the Java application in which the JENNIFER agent is installed.
Response time	The difference between the beginning and end time of the transaction. It is the total time elapsed during execution of the transaction, and it is expressed in milliseconds.
CPU time	The CPU time used to execute the transaction. It is expressed in milliseconds.
SQL time	The time required to process the SQL while executing the transaction. It is expressed in milliseconds.
Fetch time	The time required to execute the next method of the java.sql.ResultSet object while executing the transaction. It is expressed in milli seconds.
TX time	The time required to execute the external transaction. It is expressed in milliseconds.
Error item	A type of exception alert occurring during execution of the transaction (critical, error, warning)
Error	An exception name occurring during execution of the transaction
Application	Application name of the transaction

The profile data is comprised of various items. The profile items are made of a method item related to method execution and a message item implying a specific event occurring during transaction handling and a message item is further divided into detailed items such as SQL, files and sockets.

Table 8-2: Reserved Messages in a Profile

Message	Descriptions
FILE-WOPEN filename	Opens the file in write mode
FILE-ROPEN filename	Opens the file in read mode
SOCKET-ISTREAM address	Opens the read stream in the TCP socket
SOCKET-OSTREAM address	Opens the write stream in the TCP socket
[cpu] GET-CONNECTION method or data source [time ms]	Gets a DB Connection from the connection pool, or makes a new one

Message	Descriptions
[cpu] CLOSE-CONNECTION	Returns a DB Connection to the pool or closes it
[cpu] FETCH # of patches/# of cumulative patches [time ms]	Patches the data from the DB. The FETCH time is included in GAP, and the cumulative CPU time of the FETCH item includes the CPU time required for FETCH.
[cpu] TX-CALL external service name [time ms]	Calls an external service module (such as Tuxedo)
THREAD-INIT thread name	A new sub-thread begins
PARAM value	Method parameter value
RETURN value	Method return value
[cpu] SQL-PREPARE_STMT*{n}*	PreparedStatement is generated
[cpu] SQL-PREPARE_CALL*{n}*	CallableStatement is generated
[cpu] SQL-EXECUTE-QUERY *{n}* [time ms]	{n} executeQuery()is executed
[cpu] SQL-EXECUTE-UPDATE*{n}* [time ms]	{n} executeUpdate() is executed
[cpu] SQL-EXECUTE*{n}* [time ms]	{n} execute() is executed
[cpu] SQL-EXECUTE-QUERY [time ms]	(Prepared/Callable)Statement.executeQuery(SQL) is called
*[cpu]*SQL-EXECUTE-UPDATE [time ms]	(Prepared/Callable)Statement.executeUpdate(SQL) is called

- If [cpu] is displayed in front of a message, then when this message is displayed, the cumulative CPU is displayed as well.
- If [time(ms)] is displayed at the end of a message, then the time spent to execute the logic is displayed.
- In the SQL related reserved word, {n} implies a JDBC object separator within a transaction. Multiple PreparedStatement objects can be made and used at the same time. In this case, it notifies which object actual ‘execute’ corresponds to.

8.2.2. X-View Data Transmission

The JENNIFER server collects the performance data for each service transaction in the port set by server_udp_runtime_port in the setting file, and collects the profile data in the port set by server_udp_lwst_call_stack_port.

The profile data size is affected by the profile setting. Since the maximum data size that can be sent via UDP is 64kB, the JENNIFER agent partitions the large profile data into several packets for transmission. Each packet size is determined in the xview_profile_udp_packet_size option of the JENNIFER agent. The default value is 32757, and the unit is bytes. .

```
xview_profile_udp_packet_size=32757
```

Notice: . If the data size exceeds the value set in the xview_profile_udp_packet_size option of the JENNIFER agent, the agent partitions the data before transmitting it to the JENNIFER server. So, if the xview_profile_udp_packet_size option is set to a small size, then the number of transmission packets will be increased. To improve server performance, it is recommended to set the xview_profile_udp_packet_size option to as large a size as possible.

Notice: The protocol header size is also counted, so you should not set the xview_profile_udp_packet_size option to larger than 60000.

The JENNIFER server stores the packets of profile data sent by the JENNIFER agent in a temporary queue, and combines them when all of the packets have been received. The size of the temporary queue is set in the xview_profile_multi_packet_queue_size option of the JENNIFER server. Default is 303.

```
xview_profile_multi_packet_queue_size=303
```

If not every packet has been received after passing the threshold has been passed (default is 2 seconds), the JENNIFER server will organize the profile data using the packets stored in the temporary queue, and generate a time-out message. When the time-out occurs, some portions of the profile data for the transaction are lost. You can set the threshold using the xview_profile_multi_packet_time_out option. Default is 1000. It is expressed in miliiseconds.

```
xview_profile_multi_packet_time_out = 1000
```

If the response time of the transaction has been delayed than the threshold (default is 0), the JENNIFER agent does not send the profile data about the transaction to the JENNIFER server. You can set the threshold using the xview_profile_ignore_resp_time option. Default is 0(ms).

```
xview_profile_ignore_resp_time = 0
```

Notice: The xview_profile_ignore_resp_time option can be used in the JENNIFER agent and JENNIFER server. Use it in the agent to set the time for sending data into JENNIFER server and use it in the server to set the time for saving the data..

- For more details on the network settings, please refer to [Network setting] and [JENNIFER server network organization].

Notice: The maximum data size that can be sent via UDP(UDP Send Buffer Size) is 64 KB. The default for Sun Microsystems Solaris and HP HP-UX is 64KB, but a smaller value is used for IBM AIX. If the data size that the JENNIFER agent sends via UDP exceeds the maximum capacity, then the data is not sent and is lost. If the UDP transmission data size defined in the OS is exceeded, an error message will be recorded in the log file. However, if the buffer size of the network equipment between the JENNIFER agent and the server is exceeded, then no error message will be recorded and the packet will disappear.

8.2.3. Reliable collection of the large sized profiled data

Even if we divide the large sized profile data into multiple packets for transmission, due to the UDP characteristics, some losses are unavoidable. Although the profile data is small, losses due to the UDP characteristics can occur.

Therefore, JENNIFER provides a method to prevent UDP transmission losses based on the profile data size or the transaction response time. Without sending the legitimate profile data to the JENNIFER Server, it saves it in the hardware where the JENNIFER Agent is installed and when a request arrives, it retrieved the profile data by calling it in the reverse direction. To use this function, AgentDB should be used.

Using the JENNIFER Agent's agent_db_enabled option, you can specify whether to use AgentDB. Set this option to true.

```
agent_db_enabled = true
```

AgentDB saves the data in an asynchronous type. Using the JENNIFER Agent's agent_db_max_queue option, you can set the standby queue size. If the storage standby queue is full, then the profile data is not stored in AgentDB but instead, it sends them to the JENNFIER Serve same as before. However, if the standby queue with the default of 10 gets hardly filled up and if we increase the queue size then it can lead to an increase in the memory use. So you are not recommended to modify this option.

```
agent_db_max_queue = 10
```

You can use the JENNFIER Agent's agent_db_rootpath option to set the directory location used by AgentDB. The default is the current directory.

```
agent_db_rootpath = .
```

The JENNIFER Agent makes a ‘db’ directory in the location set by the agent_db_rootpath option and makes sub directories for each date to store the data. The directories set by this option are not automatically created. So, you can set an arbitrary directory only if it exists. In addition, a write privilege should be assigned. Multiple JENNIFER Agents may not use the same directory to save AgentDB. If the same JENNIFER Agent ID is used temporarily, then it can damage the file. And, to change this value, you must restart the Java application where the JENNIFER Agent is installed. If the directory is changed, simply copy the previous files to the newly changed directory for use in the future. But, you can only copy them after stopping the Java application.

Using the JENNIFER Agent’s xview_profile_dump_entry option, you can set the profile data size to be stored in AgentDB. You can save the profile data with a size greater than what is set by this option in AgentDB. The default is a max integer value and the unit is in bytes.

```
xview_profile_dump_entry = 10240000
```

Using the JENNIFER Agent’s xview_profile_dump_elapsed option, you can set the transaction response time of the profile data to be stored in AgentDB. The default is a max integer value and the unit is in milliseconds.

```
xview_profile_dump_elapsed = 300000
```

For instance, if some transaction using too much CPU is hanging for more than five minutes and the active profile data size shown in the active service detail screen is very large, then to resolve this problem, you need to collect the profile data for the transaction. If you rely on UDP transmission, then the data can be lost. In this case, set the xview_profile_dump_elapsed option to 300000 which allows the transaction profile data to be saved in AgentDB. You can easily check it here.

Notice: The xview_profile_dump_elapsed and xview_profile_dump_entry options have an OR operation.

The JENNIFER Agent’s log_xview_profile_dump option is a debug option to check whether the profile data is stored in AgentDB. The default is false.

```
log_xview_profile_dump = true
```

The number of profile data may not exceed the number set by using the profile_buffer_size option. If you want to collect the profile data with many items then you must set this option to large enough. For more details, refer to [Profiling Control].

8.2.4. Saving the X-View Data

The JENNIFER server saves the X-View data in a file. You can set the directory to which the X-view data is saved using the data_directory option of the JENNIFER server.

```
data_directory=../../data/file/
```

To prevent the loss of data file, stop the JENNIFER server and modify the option in the configuration file.

8.2.4.1. Transaction Data

After storing the recently collected transaction data in the memory queue, the JENNIFER server reads in the transaction data from the memory queue every two seconds. Therefore, the memory queue size should be set to about twice the maximum TPS. The queue size can be set using the xview_server_queue_size option of the JENNIFER server. After changing this option, you should restart the JENNIFER server to apply the change. .

```
xview_server_queue_size = 512
```

JENNIFER does not save the performance data of the transaction that surpassed the setting time(ms) in the xview_point_ignore_resp_time option.

```
xview_point_ignore_resp_time=0
```

8.2.4.2. Profile Data

After storing the profile data collected from the JENNIFER agent in a temporary queue, the JENNIFER server saves it to a file once every half-second. The memory queue size can be set by the xview_profile_cache_queue_size option. If the queue is filled before the profile data has been saved in a file, then some of profile data could be lost. Therefore, since the storage period is 0.5 seconds, it is desirable to set the xview_profile_cache_queue_size option to about 50% of the maximum TCP. After changing this option, you should restart the JENNIFER server to apply the change..

```
xview_profile_cache_queue_size = 512
```

The JENNIFER Server saves the profile data in the real time X-View profile file and the daily X-View profile file. The real time X-View profile file stores the profile data for the recently executed transaction consecutively. The period is affected by the real time X-View profile data size and the service request rate. The daily X-View profile file is to facilitate inquiries about the past profile data. The real time X-View profile file size can be set by the JENNIFER Server’s xview_profile_isam_file_max_size option. The default is 128mb. If this option is changed, the JENNIFER Server should be restarted..

```
xview_profile_isam_file_max_size=10mb
xview_profile_isam_file_max_size=512mb
xview_profile_isam_file_max_size=1gb
```

This option is set with the units of mb(mage bytes) or gb(gigab bytes). When the JENNIFER Server is executed for the first time, this options sets the size of a file that is created.

If the profile data has a response time smaller than what is set by the xview_profile_ignore_resp_time option of the JENNIFER Server, then it is not saved in the real time X-View profile file. The default is 0 and the unit is in milliseconds. .

```
xview_profile_ignore_resp_time = 100
```

It is more effective to set this option in the JENNIFER Agent than in the JENNIFER Server. If set in the JENNIFER Agent, then the profile data is not transmitted to the network at all.

Although the profile data is stored in the real time X-View profile file, but depending on the condition, you can specify not to save the profile data in the daily X-View profile file. First of all, if the profile data has a smaller response time than what is set in the JENNIFER Server’s xvdaily_ignore_resp_time option, it is not saved in the daily X-View profile file. The default is 500 and the unit is in milliseconds.

```
xvdaily_ignore_resp_time = 500
```

Second, you can specify whether to save thel profile data for each JENNIFER Agent. For instance, when monitoring the same applications made of clustering, you can choose to save the profile data only for the representative JENNIFER Agent for efficient use of the file storage space. This can be set by the JENNIFER Server’s xvdaily_agents option.

```
xvdaily_agents = W11,W12
```

If the value is not set, JENNIFER does not define the agent when it saves the profile data.

8.3. Analysis of the X-View Screen

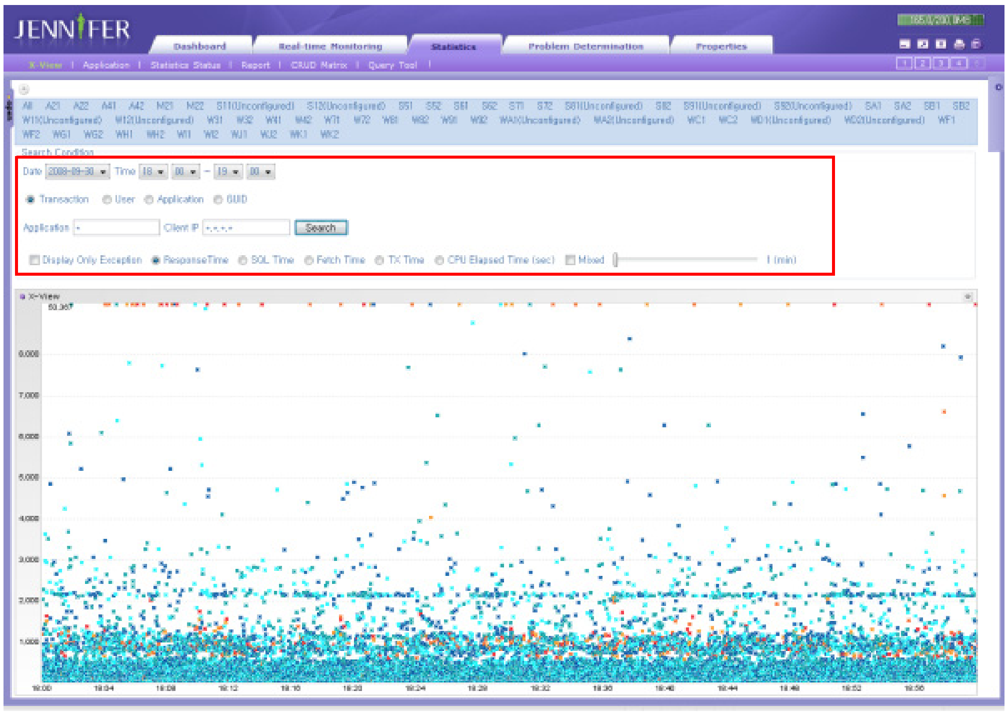
8.3.1. X-View Chart

A user can check the X-View data on an X-View chart. The X-View charts can be accessed via the **[Dashboard | JENNIFER dashboard]** menu, the **[Real time monitor-ing| X-View]** menu, and the **[Statistical analysis | X-View]** menu.

- **[Dashboard | JENNIFER dashboard]** - This is an X-view chart that exists on the dashboard. The transaction types are fixed, and it is not possible to search by an application or a client IP address. This chart shows the status of transaction processing in real time.
- **[Real time monitoring | X-View]** - This chart shows the status of transaction processing in real time. You can select from various types of charts, and also search by an application or a client IP address.
- **[Statistical analysis | X-View]** - In this chart, a user can specify the time to view the status of transaction processing. You can select from various types of charts, and also search by an application or a client IP address. The searchable data is decided based on the saving period that is set by CleanActor.

Notice: When you view the status of transaction processing in the [Statistical analysis | X-View] menu, it is possible to browse through the transaction data recently executed. However, if less than 10 minutes has elasped since the relevant transaction, you may not be able to find the profile data. The profile index is generated every 10 minutes, and if no index has been made, then you may not search for it.

Figure 8-8:X-View Chart



Without moving the [Real Time Monitoring| X-View] menu and the [Statistical Analysis | X-View] menu, you can use the real time monitoring check box in the upper area in order to inquire about the real time X-View data and the daily X-View data.

8.3.1.1. Y-axis

The Y-axis in an X-View chart represents the execution time. However, you can set different references for the Y-axis, depending on the type of X-view. For instance, in principle, the Y-axis represents the elapsed time, but you can change this to SQL time or fetch time.

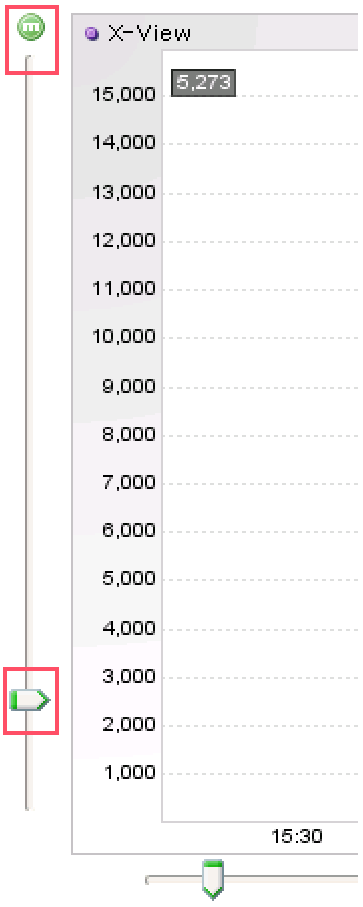
The limits for the Y-axis are ranged from 0 sec to 9 sec. The user can change these limits arbitrarily. It is expressed in milliseconds.

- Up-cursor key - If you press this key, then the magnitude of the Y-axis will increase. The change in the magnitude is determined by the current magnitude of the Y-axis.
- Low-cursor key - If you press this key, then the magnitude of the Y-axis will decrease. The change in the magnitude is determined by the current magnitude of the Y-axis.

If you press the up(low)-cursor key with Shift, the magnitude of the X(Y)-axis will increase(decrease) by 10 times.

Notice: The lower limit for the magnitude of the Y-axis is 100 mili seconds.

Figure 8-9:X-View Profile



If the transaction execution time exceeds the maximum of the Y-axis, then it will be displayed in the upper area of the X-view. To find out the accurate time, you need to increase the magnitude of the Y-axis by using the up-cursor key.

8.3.1.2. X-axis

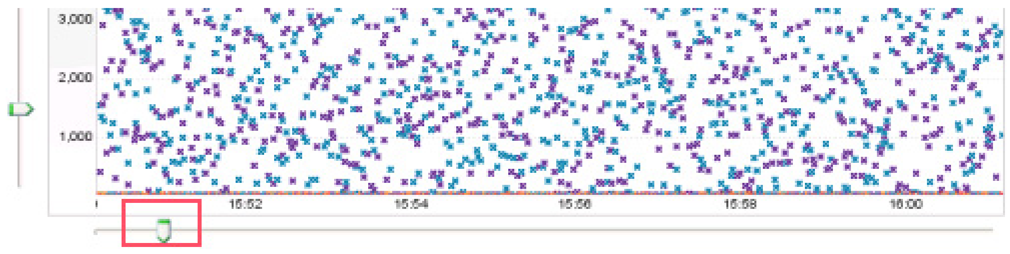
The X axis of the X-View chart represents the transaction execution time. Since in the **[Statistical analysis | X-View]** menu, the user specifies the time period to query, the X-axis is fixed.

Notice: When you view the status of transaction processing in the **[Statistical analysis | X-View]** menu, you are recommended to set the time period to less than one hour. A Java application with a high TPS creates a large amount of transaction data, so the searching time takes longer, and the heap memory of the Java plug-in will tend to be insufficient.

Notice: In the **[Dashboard | JENNIFER dashboard]** menu and the **[Real time monitoring | X-View]** menu, the default value for the X-axis is 10 minutes. The user can change this value arbitrarily.

- Left-cursor key - If you press this key, the magnitude of the X-axis will increase by one minute. For instance, if the current time period is from 10:10 to 10:20, when you press the left-cursor key, the time period will shift to 10:09 to 10:20. If you press the left -cursor key while holding down the Shift key, the magnitude will increase by 10 minutes. For instance, if the current time period is from 10:10 to 10:20, when you press the left-cursor key and the Shift key, the time period will shift to10:00 to 10:20.
- Right-cursor key - If you press this key, the magnitude of the X-axis will decrease by one minute. For instance, if the current time period is from 10:09 to 10:20, when you press the right-cursor key, the time period will shift to 10:10 to 10:20. If you press the right-cursor key while holding down the Shift key, the magnitude will decrease by 10 minutes. For instance, if the current time period is from 10:00 to 10:20, when you press the right-cursor key and the Shift key, the time period will shift to 10:10 to 10:20

Figure 8-10:.X axos of the X-View



Using a slider, you can set the entire region size of the X-axis.

Notice: When displaying a transaction on the X-View chart, the X coordinates are determined based on the time when the JENNIFER Server collects the transaction information. Since the JENNIFER Server can monitor more than one Java application with different system time, it uses the X coordinates as the JENNIFER Server time when displaying a transaction on the X-view chart.

In the pervious versions, you could only use the plus/minus keys to change the area while fixing the X coordinate area. However, now, the plus/minus keys operate in the same way as the left/right directional keys. Instead, using the area selection function of the X-view chart, you can analyze specific areas. The following illustrates how you can use the area selection function on the X-View chart.

- On the X-View chart, move your mouse cursor to the starting point of the area and double click on it. Then a dotted line will appear in the location.
- Move your cursor to the end of the area and double click on your mouse. Again, a dotted line will appear on it.
- Now, press the Enter key to magnify the views of the selected area only.

8.3.2. Types of X-View Charts

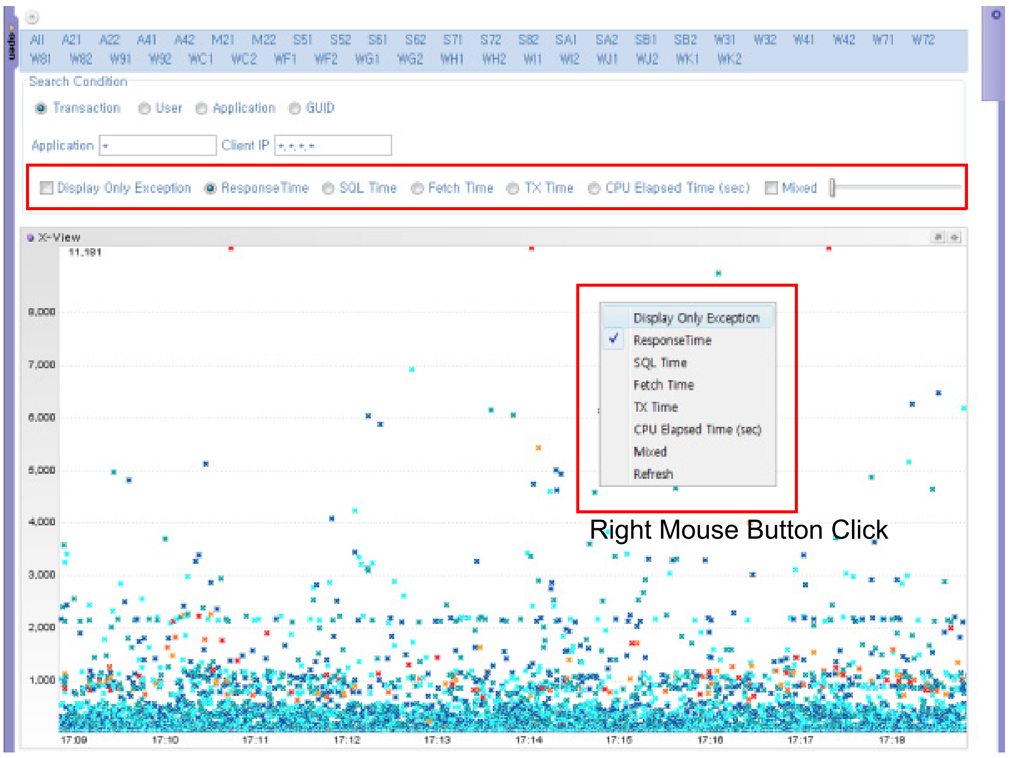
The different types of X-View charts offer different ways to display the transaction. For instance, transaction type is used to display all the individual transactions on an X-view chart, but user type is used to group transactions with the same client ID and display it on the X-view chart. Using the [Real time monitoring | X-View] menu and the [Statistical analysis | X-View] menu, you can select the desired type of X-view.

Notice: In the **[Dashboard | JENNIFER dashboard]** menu, the transaction type of X-view is used.

8.3.2.1. Transaction

The transaction type is used to display the transaction as a X-shaped dot on the X-view chart.

Figure 8-11:Transaction Type of X-View Chart



Search Condition: [✓ Transaction] User Application GUID

Application: * Client IP: *.*.*.*

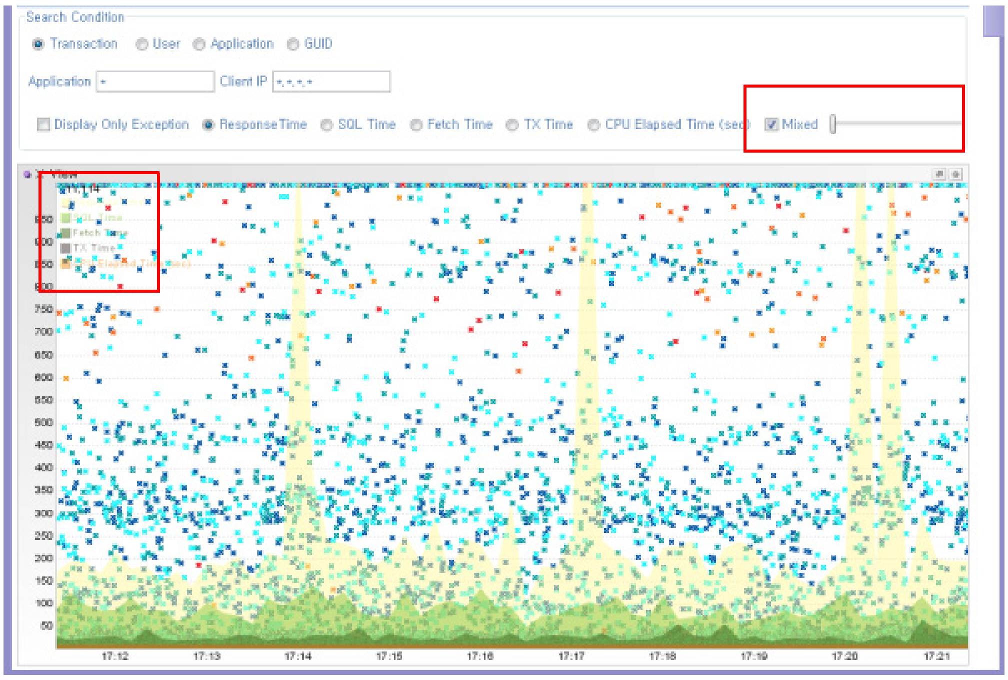
Display Only Exception [✓ ResponseTime] SQL Time Fetch Time TX Time CPU Elapsed Time (sec) Mixed

Normal transactions are shown in blue, while transactions with exceptions are displayed in red. You can change various settings using the searching conditions or the contextual menu.

- Display only exceptions - Only transactions with exceptions are displayed.

- Response time - The Y-axis represents the response time. This is the default setting for the Y-axis.
- SQL time - The Y-axis represents the SQL time.
- Fetch time - The Y-axis represents the Fetch time.
- TX time - The Y-axis represents the external transaction time.
- CPU time - The Y-axis represents the CPU time.
- Mixed - You can obtain the average of elapsed time, SQL time, Fetch time, TX time and CPU time, and display it to describe the region. The detailed regional values can be set using the slider.
- Refresh- Refresh the total X-View transaction data.

Figure 8-12:Transaction Type of X-View Chart (Mixed)



Search Condition

[✓ Transaction] User Application GUID

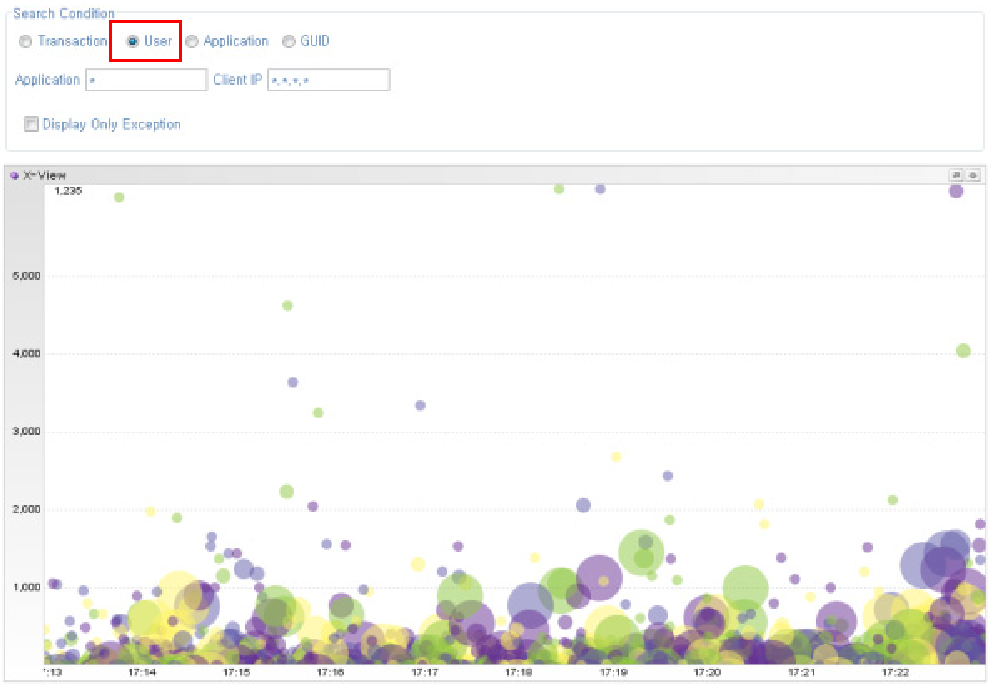
Application: *   Client IP: *.*.*.*

Display Only Exception [✓ Response Time] SQL Time Fetch Time TX Time CPU Elapsed Time (sec) [✓ Mixed]

8.3.2.2. User

The user type of X-view is used to group transactions with the same client ID and represent the group as a circle on the X-view chart. The larger the diameterof the circle is, the more the transactions are being executed by a certain user. However, the color of the circle is meaningless in this chart.

Figure 8-13:User Type of X-View Chart

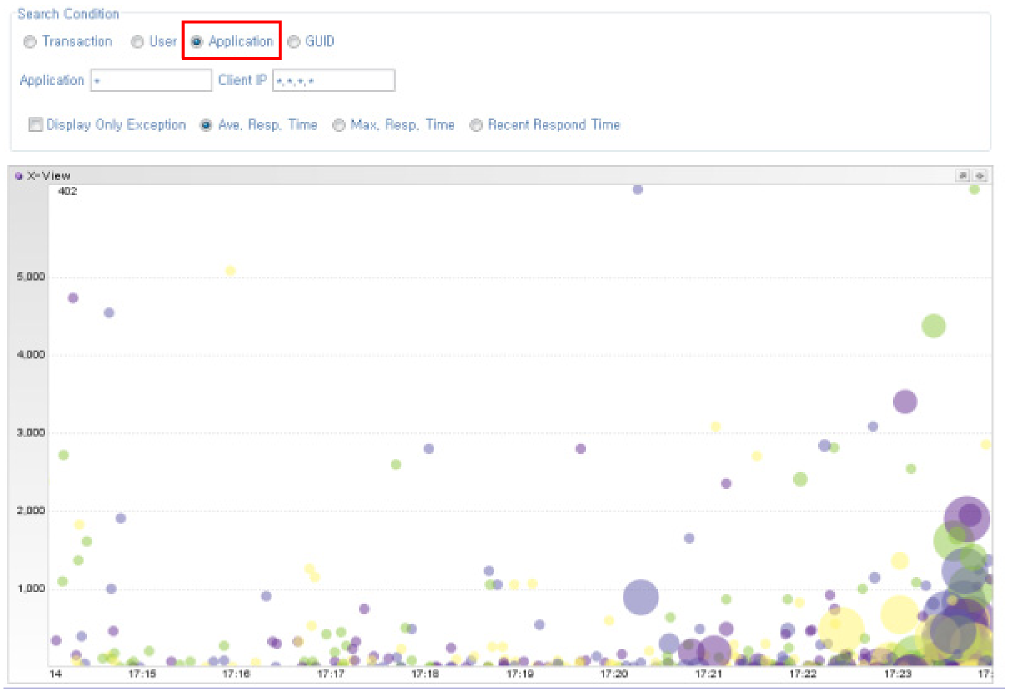


You can change various settings using the searching conditions and the contextual menu. Display only exception - Only transactions with exceptions are displayed.

8.3.2.3. Application

The user type of X-view is used to group transactions with the same application and represent the group as a circle on the X-view chart. The larger the diameter of the circle is, the more the transactions are being executed by the application. However, the color of the circle is meaningless in this chart.

Figure 8-14:Application Type of X-View Chart


Search Condition

Transaction User [✓ Application] GUID

Application * Client IP *.*.*.*

Display Only Exception [✓ Ave. Resp. Time] Max. Resp. Time Recent Respond Time

You can change various settings using the searching conditions and the contextual menu.

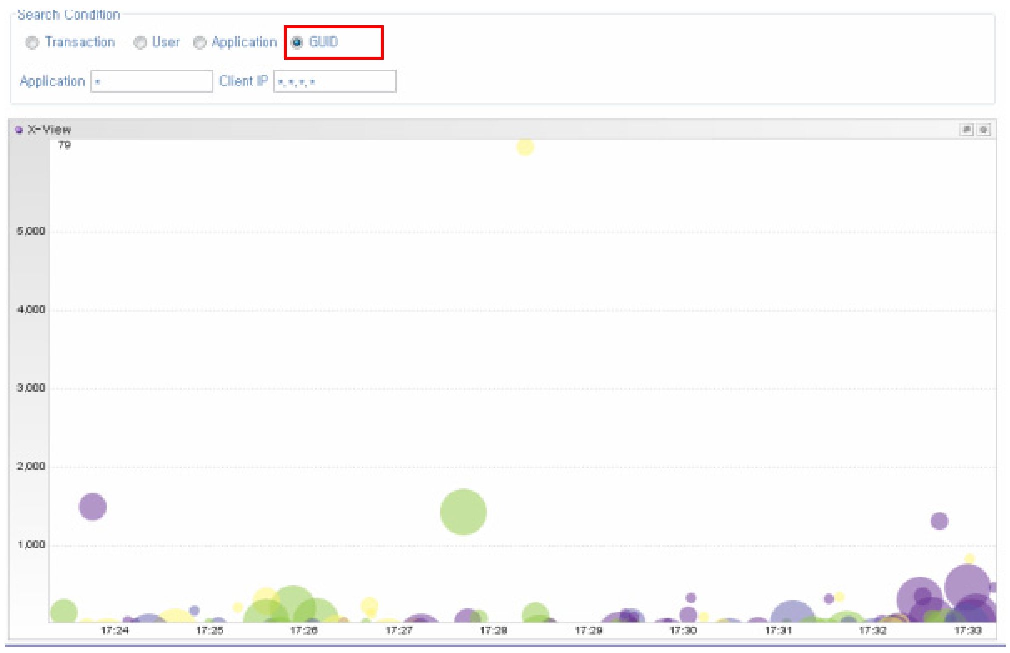
- Display only exception - Only transactions with exceptions are displayed.
- Average response time - The Y-axis represents the average response time. This is the default setting for the Y-axis.
- Maximum response time - The Y-axis represents the maximum response time.
- Recent response time - The Y-axis represents the recent response time.

8.3.2.4. GUID

The GUID type of X-view is used to group transactions with the same GUID keys and represent the group as a small circle on the X-view chart. The circles have the same diameters. However, the color of the circle is meaningless.

Notice: If you select a certain circle, then the transaction data with the same GUID key will be displayed in the transaction list. But if the transaction with the same GUID key is located outside of the selected range of the X-axis, then it will not be displayed.

Figure 8-15:GUID Type of X-View Chart



The maximum (end time) and minimum (start time) response times for each transaction with the same GUID are calculated.

8.3.3. Searching by Application & Client IP Address

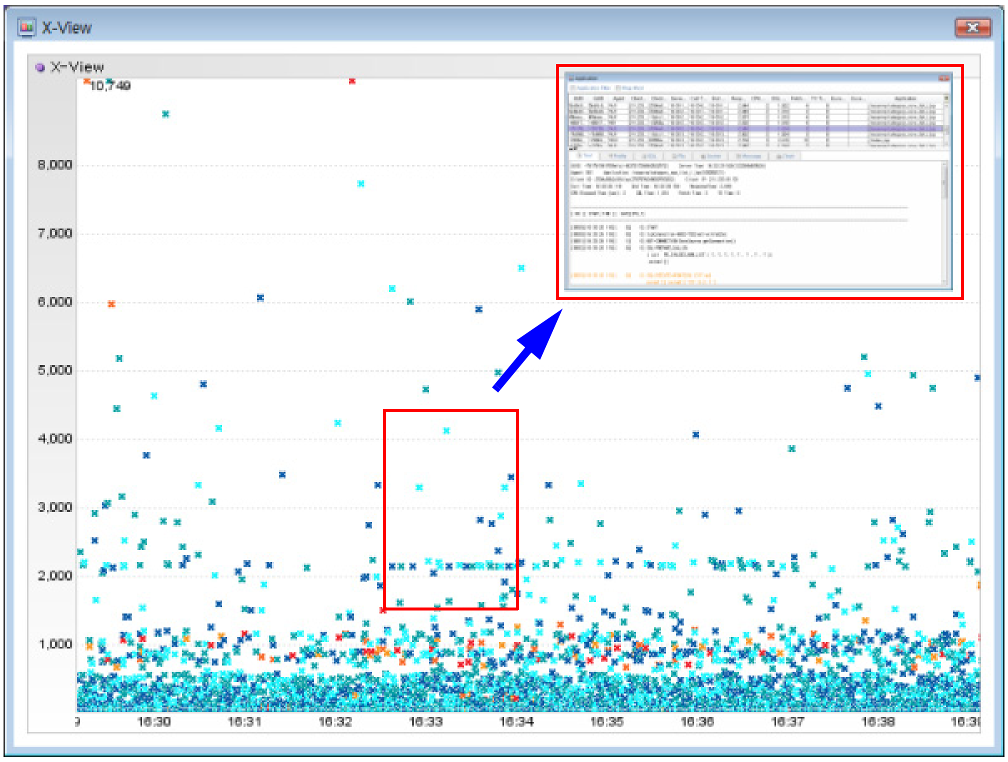
You can filter out the transactions shown on the X-View chart using the application and the client IP address. Application name will be shown on the X-view chart. LIKE searching is not supported. You can use a wild card (*) option for the client IP address.

```
192.168.0.1
192.*.0.1
192.*.*.*
```

You can use the filtering function after inputing the searching conditions in the application and client IP field and pressing the enter key. If you click the [Search] button in the **[Statistics | X-View]** menu, transaction is not filter out. You can also use the filtering function after inputing the searching conditions in the application and client IP field and pressing the enter key in this menu.

8.3.4. Transaction List

You can check the transaction data by selecting an arbitrary point on the X-view. As shown in the figure below, the list in the pop-up window represents the transaction performance information.



For items shown after the transaction has been selected, please refer to [Transaction data].

Notice: If the type of your X-View chart is 'transaction', then only one list is displayed. In other cases, two lists are displayed. The first list display the data with a certain type and the second list displays the transactions included in the item selected in the first list.

Please note that not all of the information is displayed. For instance, if it is not a WAS system, the Client IP or ID will be zero. The following is a list of restricted items.

Table 8-3: Limitations for Each Item of Transaction Performance Information

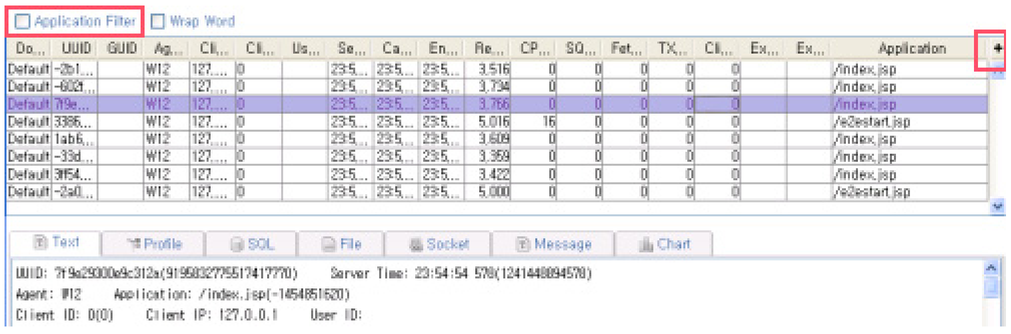
Item	Collection Condition
GUID	Display only if GUID is set.
Clent IP	The monitoring target is a WAS system.

Table 8-3: Limitations for Each Item of Transaction Performance Information

Item	Collection Condition
Client ID	The monitoring target is a WAS system.
User ID	The monitoring target is a Java application server. In case that the user ID is set to be traced.
CPU Time	The jennifer20.so file is installed.
SQL Time	JDBC resource tracking is set.
Fetch Time	JDBC resource tracking is set.
Tx Time	The system is interconnected to tx_client or TP monitors.
Client Response Time	The monitoring target is a Java application server. In case that the client response time is set to be traced.

In the upper right corner of your transaction list, press the [+] icon in order to export a transaction list as a CSV file or a text file containing the profile data.

Figure 8-16:



Do...	UUID	GUID	Ag...	Cl...	Cl...	Us...	Se...	Ca...	En...	Re...	CP...	SQ...	Fet...	TX...	Cl...	Ex...	Ex...	Application
Default	-2b1...		W12	127....	0		23:5...	23:5...	23:5...	3,516	0	0	0	0	0			/index.jsp
Default	-602f...		W12	127....	0		23:5...	23:5...	23:5...	3,734	0	0	0	0	0			/index.jsp
Default	7f9e...		W12	127....	0		23:5...	23:5...	23:5...	3,766	0	0	0	0	0			/index.jsp
Default	3386...		W12	127....	0		23:5...	23:5...	23:5...	5,016	16	0	0	0	0			/e2estart.jsp
Default	1ab6...		W12	127....	0		23:5...	23:5...	23:5...	3,609	0	0	0	0	0			/index.jsp
Default	-33d...		W12	127....	0		23:5...	23:5...	23:5...	3,359	0	0	0	0	0			/index.jsp
Default	3f54...		W12	127....	0		23:5...	23:5...	23:5...	3,422	0	0	0	0	0			/index.jsp
Default	-2a0...		W12	127....	0		23:5...	23:5...	23:5...	5,000	0	0	0	0	0			/e2estart.jsp

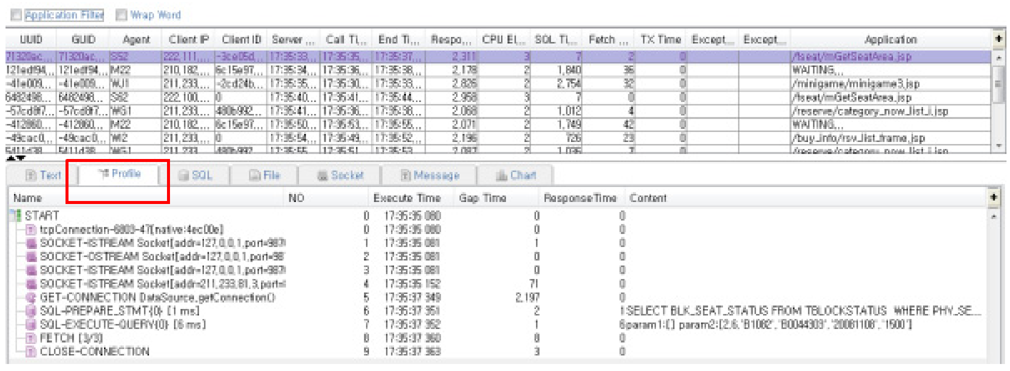
```
UUID: 7f9e29300e9c312a(9195832775517417770)     Server Time: 23:54:54 578(1241448894578)
Agent: W12     Application: /index.jsp(-1454851620)
Client ID: 0(0)     Client IP: 127.0.0.1     User ID:
```

Check the application filter in the upper area. Only the transactions corresponding to the same application as the transactions selected in the list will be displayed on the X-View chart.

8.3.5. Profile Tab Area

In the profile tab below the transaction list, the profile data of the selected transaction is displayed.

Figure 8-17:Profile Tab Area



Application Filter   Wrap Word

UUID	GUID	Agent	Client IP	Client ID	Server ...	Call Ti...	End Ti...	Respo...	CPU El...	SQL Ti...	Fetch ...	TX Time	Except...	Except...	Application
71320ac...	71320ac...	S52	222.111....	-3ce05d...	17:35:33...	17:35:35...	17:35:37...	2,311	3	7	2	0			/tseat/mGetSeatArea.jsp
121edf94...	121edf94...	M22	210.182....	6c15e97...	17:35:34...	17:35:36...	17:35:38...	2,178	2	1,840	36	0			WAITING...
-41e009...	-41e009...	WJ1	211.233....	-2cd24b...	17:35:35...	17:35:30...	17:35:33...	2,826	2	2,754	32	0			/minigame/minigame3.jsp
6482498...	6482498...	S62	222.100....	0	17:35:40...	17:35:41...	17:35:44...	2,958	3	7	0	0			/tseat/mGetSeatArea.jsp
-57cd8f7...	-57cd8f7...	WG1	211.233....	480b992...	17:35:41...	17:35:36...	17:35:38...	2,068	2	1,012	4	0			/reserve/category_now_list_i.jsp
-412860...	-412860...	M22	210.182....	6c15e97...	17:35:50...	17:35:53...	17:35:55...	2,071	2	1,749	42	0			WAITING...
-49cac0...	-49cac0...	WI2	211.233....	0	17:35:54...	17:35:49...	17:35:52...	2,196	2	726	23	0			/buy_info/rsv_list_frame.jsp

Text | Profile | SQL | File | Socket | Message | Chart

Name	NO	Execute Time	Gap Time	ResponseTime	Content
START	0	17:35:35 080	0	0	
tcpConnection-6803-47[native:4ec00e]	0	17:35:35 080	0	0	
SOCKET-ISTREAM Socket[addr=127.0.0.1,port=987	1	17:35:35 081	1	0	
SOCKET-OSTREAM Socket[addr=127.0.0.1,port=98	2	17:35:35 081	0	0	
SOCKET-ISTREAM Socket[addr=127.0.0.1,port=987	3	17:35:35 081	0	0	
SOCKET-ISTREAM Socket[addr=211.233.81.3,port=	4	17:35:35 152	71	0	
GET-CONNECTION DataSource.getConnection()	5	17:35:37 349	2,197	0	
SQL-PREPARE_STMT{0} [1 ms]	6	17:35:37 351	2	1	SELECT BLK_SEAT_STATUS FROM TBLOCKSTATUS WHERE PHY_SE...
SQL-EXECUTE-QUERY{0} [6 ms]	7	17:35:37 352	1	6	param1:[] param2:[2,6,'B1082','B0044303','20081108','1500']
FETCH [3/3]	8	17:35:37 360	8	0	
CLOSE-CONNECTION	9	17:35:37 363	3	0	

Notice: If no profile data exists for the transaction, inner area of the text tab will be activated, making it impossible for the user to select another tab.

8.3.5.1. Text

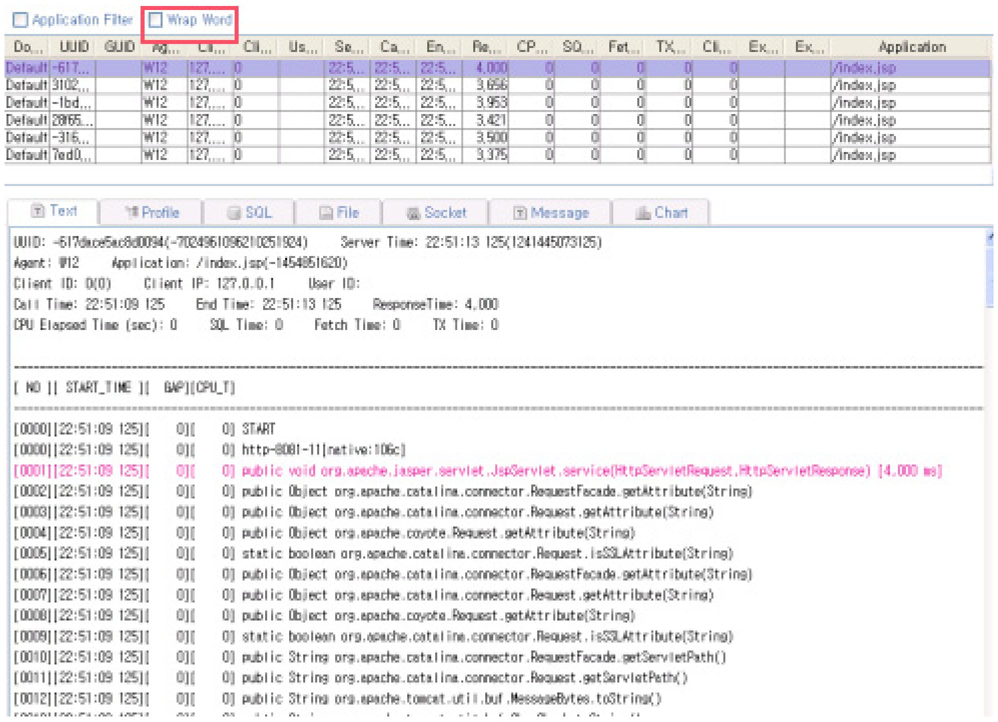
In the text tab, the transaction data for the transaction selected from the transaction list and every profile item are displayed as text.

The color of the text will vary depending on the elapsed time of the profile item. It is expressed in milliseconds.

- If the elapsed time is greater than1000 - profile color
- If the elapsed time is between 500 and 1000 - profile color
- If the elapsed time is between 100 and 500 - profile color
- If the elapsed time is between 10 and 100 - profile color

If you wish to hide the left/right scroll bar from the text tab, you should select the [Wrap Word] option.

Figure 8-18:Wrap Word



Do...	UUID	GUID	Ag...	Cl...	Cli...	Us...	Se...	Ca...	En...	Re...	CP...	SQ...	Fet...	TX...	Cli...	Ex...	Ex...	Application
Default	-617...		W12	127,...	0		22:5...	22:5...	22:5...	4,000	0	0	0	0	0			/index.jsp
Default	3102...		W12	127,...	0		22:5...	22:5...	22:5...	3,656	0	0	0	0	0			/index.jsp
Default	-1bd...		W12	127,...	0		22:5...	22:5...	22:5...	3,953	0	0	0	0	0			/index.jsp
Default	28f65...		W12	127,...	0		22:5...	22:5...	22:5...	3,421	0	0	0	0	0			/index.jsp
Default	-316...		W12	127,...	0		22:5...	22:5...	22:5...	3,500	0	0	0	0	0			/index.jsp
Default	7ed0...		W12	127,...	0		22:5...	22:5...	22:5...	3,375	0	0	0	0	0			/index.jsp

```
UUID: -617dace5ac8d0094(-7024961096210251924)     Server Time: 22:51:13 125(1241445073125)
Agent: W12     Application: /index.jsp(-1454851620)
Client ID: 0(0)     Client IP: 127.0.0.1     User ID:
Call Time: 22:51:09 125     End Time: 22:51:13 125     ResponseTime: 4,000
CPU Elapsed Time (sec): 0     SQL Time: 0     Fetch Time: 0     TX Time: 0

---------------------------------------------------------------------------------------------------------------------------------
[ NO || START_TIME ||  GAP][CPU_T]
---------------------------------------------------------------------------------------------------------------------------------
[0000][22:51:09 125][    0][    0] START
[0000][22:51:09 125][    0][    0] http-8081-11[native:106c]
[0001][22:51:09 125][    0][    0] public void org.apache.jasper.servlet.JspServlet.service(HttpServletRequest,HttpServletResponse) [4,000 ms]
[0002][22:51:09 125][    0][    0] public Object org.apache.catalina.connector.RequestFacade.getAttribute(String)
[0003][22:51:09 125][    0][    0] public Object org.apache.catalina.connector.Request.getAttribute(String)
[0004][22:51:09 125][    0][    0] public Object org.apache.coyote.Request.getAttribute(String)
[0005][22:51:09 125][    0][    0] static boolean org.apache.catalina.connector.Request.isSSLAttribute(String)
[0006][22:51:09 125][    0][    0] public Object org.apache.catalina.connector.RequestFacade.getAttribute(String)
[0007][22:51:09 125][    0][    0] public Object org.apache.catalina.connector.Request.getAttribute(String)
[0008][22:51:09 125][    0][    0] public Object org.apache.coyote.Request.getAttribute(String)
[0009][22:51:09 125][    0][    0] static boolean org.apache.catalina.connector.Request.isSSLAttribute(String)
[0010][22:51:09 125][    0][    0] public String org.apache.catalina.connector.RequestFacade.getServletPath()
[0011][22:51:09 125][    0][    0] public String org.apache.catalina.connector.Request.getServletPath()
[0012][22:51:09 125][    0][    0] public String org.apache.tomcat.util.buf.MessageBytes.toString()
```

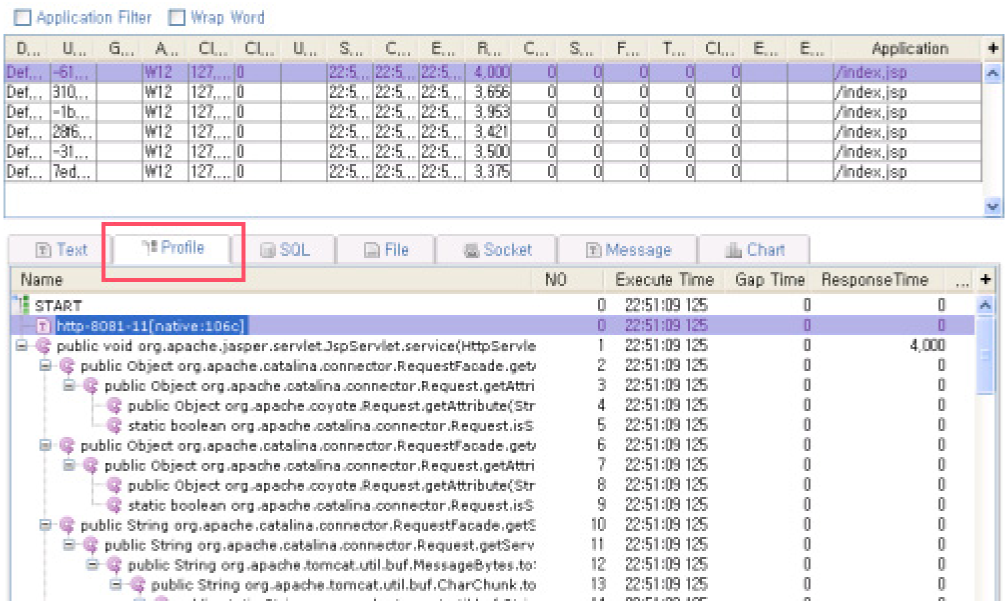
The profile information begins with ‘START’ and ends with ‘END’. ‘START’ and ‘END’ are additional information messages that are only displayed on the screen to specify the beginning and the end of a file. The profile can be classified by types of method calls and execution messages. The following is a list of predefined execution messages in JENNIFER.

8.3.5.2. Profile

The profile tab displays a tree of every profile item for the transaction selected in the transaction list. The following is a description of the profile list column.

Column Description Name Profile item name. It displays a method name or message type name. NO Profile item’s unique number. Execution time It is the start time of the profile item. Based on the Java application time where the JENNIFER Agent is installed. Gap time The start dime difference between the profile item and the previous profile item. Response time Time spent to execute the profile item. Content Relevant content.

Figure 8-19:Profile



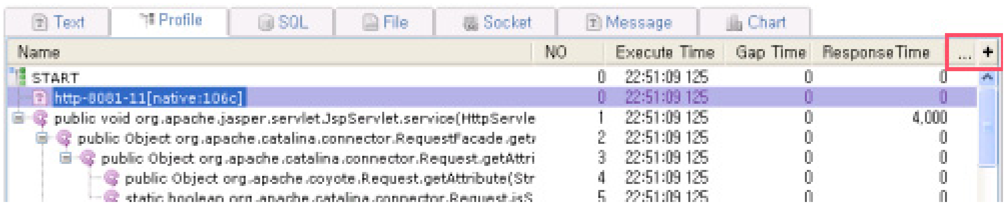
D...	U...	G...	A...	Cl...	Cl...	U...	S...	C...	E...	R...	C...	S...	F...	T...	Cl...	E...	E...	Application
Def...	-61...		W12	127,...	0		22:5...	22:5...	22:5...	4,000	0	0	0	0	0			/index.jsp
Def...	310...		W12	127,...	0		22:5...	22:5...	22:5...	3,656	0	0	0	0	0			/index.jsp
Def...	-1b...		W12	127,...	0		22:5...	22:5...	22:5...	3,953	0	0	0	0	0			/index.jsp
Def...	2816...		W12	127,...	0		22:5...	22:5...	22:5...	3,421	0	0	0	0	0			/index.jsp
Def...	-31...		W12	127,...	0		22:5...	22:5...	22:5...	3,500	0	0	0	0	0			/index.jsp
Def...	7ed...		W12	127,...	0		22:5...	22:5...	22:5...	3,375	0	0	0	0	0			/index.jsp

Name	NO	Execute Time	Gap Time	ResponseTime
START	0	22:51:09 125	0	0
http-8081-11[native:106c]	0	22:51:09 125	0	0
public void org.apache.jasper.servlet.JspServlet.service(HttpServle	1	22:51:09 125	0	4,000
public Object org.apache.catalina.connector.RequestFacade.getA	2	22:51:09 125	0	0
public Object org.apache.catalina.connector.Request.getAttri	3	22:51:09 125	0	0
public Object org.apache.coyote.Request.getAttribute(Str	4	22:51:09 125	0	0
static boolean org.apache.catalina.connector.Request.isS	5	22:51:09 125	0	0
public Object org.apache.catalina.connector.RequestFacade.getA	6	22:51:09 125	0	0
public Object org.apache.catalina.connector.Request.getAttri	7	22:51:09 125	0	0
public Object org.apache.coyote.Request.getAttribute(Str	8	22:51:09 125	0	0
static boolean org.apache.catalina.connector.Request.isS	9	22:51:09 125	0	0
public String org.apache.catalina.connector.RequestFacade.getS	10	22:51:09 125	0	0
public String org.apache.catalina.connector.Request.getServ	11	22:51:09 125	0	0
public String org.apache.tomcat.util.buf.MessageBytes.to:	12	22:51:09 125	0	0
public String org.apache.tomcat.util.buf.CharChunk.to	13	22:51:09 125	0	0

Notice: The gap time helps to analyze the unprofiled area while executing a transaction. For instance, if a certain profile item has a large gap time, then it means that the long time is elapsed to handle the task between the current and previous profile items. Therefore, if you specify to profile the task between the two profile items, you can analyze the details.

After selecting a profile item with a name that starts with [SQL-EXECUTE] and click on your right mouse button to build or copy an SQL. For more details, refer to [SQL]. And, using the [+] icon in the upper right corner of your profile list, you can choose to open or close the entire tree.

Figure 8-20:Sample Profile



Name	NO	Execute Time	Gap Time	ResponseTime
START	0	22:51:09 125	0	0
http-8081-11[native:106c]	0	22:51:09 125	0	0
public void org.apache.jasper.servlet.JspServlet.service(HttpServle	1	22:51:09 125	0	4,000
public Object org.apache.catalina.connector.RequestFacade.getA	2	22:51:09 125	0	0
public Object org.apache.catalina.connector.Request.getAttri	3	22:51:09 125	0	0
public Object org.apache.coyote.Request.getAttribute(Str	4	22:51:09 125	0	0
static boolean org.apache.catalina.connector.Request.isS	5	22:51:09 125	0	0

After selecting a name column of an item with a long response time in the profile list, right click on your mouse button to launch the context menu. Select [Critical Path] to search for the item with the longest response time among the lower profile items in the list.

8.3.5.3. SQL

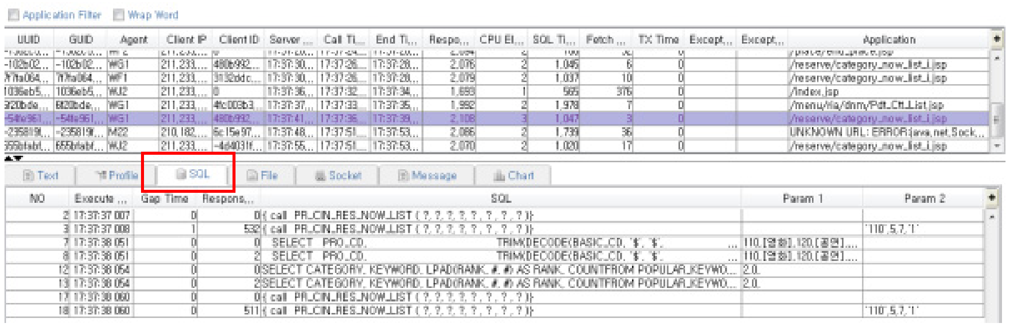
In the SQL tab, the SQL message type of profile items for the transaction selected from the transaction list is displayed. You can check the SQL list that is executed by the transaction.

The below table describes the SQL column list.

Table 8-4: SQL List Column

Column	Descriptions
NO	The sharing number for the profile item. It helps you to find the profile item in the text or profile tab.
Execution time	The time when the profile item begins. It is based on the Java application time where the JENNIFER agent is installed.
Gap time	The difference between the start times of the current and the previous profile item.
Elapsed time	The time spent by the profile item.
SQL query	SQL query
Parameter 1	A costant separated from the SQL query to prevent duplicate use of the SQL query.
Parameter 2	**A binding parameter for the SQL query**

Figure 8-21:SQL Tab

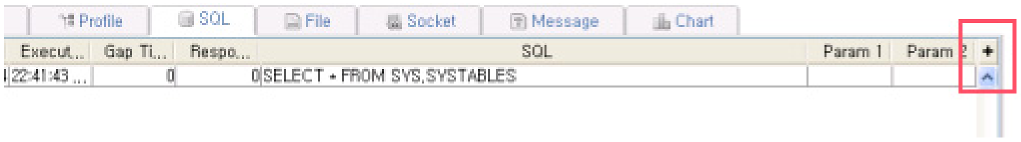


UUID	GUID	Agent	Client IP	Client ID	Server ...	Call Ti...	End Ti...	Respo...	CPU El...	SQL Ti...	Fetch ...	TX Time	Except...	Except...	Application
-102b02...	-102b02...	WG1	211.233....	480b992...	17:37:30...	17:37:26...	17:37:28...	2,076	2	1,045	6	0			/reserve/category_now_list_i.jsp
7f7fa064...	7f7fa064...	WF1	211.233....	3132ddc...	17:37:30...	17:37:26...	17:37:28...	2,079	2	1,037	10	0			/reserve/category_now_list_i.jsp
1036eb5...	1036eb5...	WJ2	211.233....	0	17:37:36...	17:37:32...	17:37:34...	1,693	1	565	376	0			/index.jsp
820bde...	820bde...	WG1	211.233....	4fc003b3...	17:37:37...	17:37:33...	17:37:35...	1,992	2	1,978	7	0			/menu/tla/dnm/Pdt_Ctt_List.jsp
-54fe961...	-54fe961...	WG1	211.233....	480b992...	17:37:41...	17:37:36...	17:37:39...	2,108	3	1,047	3	0			/reserve/category_now_list_i.jsp
-2358191...	-2358191...	M22	210.182....	5c15e97...	17:37:48...	17:37:51...	17:37:53...	2,086	3	1,739	36	0			UNKNOWN URL: ERROR:java.net.Sock...
655bfabf...	655bfabf...	WJ2	211.233....	-4d4031f...	17:37:55...	17:37:51...	17:37:53...	2,070	2	1,020	17	0			/reserve/category_now_list_i.jsp

NO	Execute ...	Gap Time	Respons...	SQL	Param 1	Param 2
2	17:37:37 007	0	0	{ call PR_CIN_RES_NOW_LIST (?, ?, ?, ?, ?, ?, ?, ?)}		
3	17:37:37 008	1	532	{ call PR_CIN_RES_NOW_LIST (?, ?, ?, ?, ?, ?, ?, ?)}		'110',5,7,'1'
7	17:37:38 051	0	0	SELECT PRO_CD, TRIM(DECODE(BASIC_CD, '$', '$', ...	110,[영화],120,[공연]....	
8	17:37:38 051	0	2	SELECT PRO_CD, TRIM(DECODE(BASIC_CD, '$', '$', ...	110,[영화],120,[공연]....	
12	17:37:38 054	0	0	SELECT CATEGORY, KEYWORD, LPAD(RANK, #, #) AS RANK, COUNTFROM POPULAR_KEYWO...	2,0,	
13	17:37:38 054	0	2	SELECT CATEGORY, KEYWORD, LPAD(RANK, #, #) AS RANK, COUNTFROM POPULAR_KEYWO...	2,0,	
17	17:37:38 060	0	0	{ call PR_CIN_RES_NOW_LIST (?, ?, ?, ?, ?, ?, ?, ?)}		
18	17:37:38 060	0	511	{ call PR_CIN_RES_NOW_LIST (?, ?, ?, ?, ?, ?, ?, ?)}		'110',5,7,'1'

By clicking on the [+] in the upper right corner of the SQL list, you can open or close the entire list.

Figure 8-22:SQL List



Execut...	Gap Ti...	Respo...	SQL	Param 1	Param 2
22:41:43 ...	0	0	SELECT * FROM SYS.SYSTABLES		

- After selecting the SQL, right-click on it
- Select [Query build] from the contextual menu, and the pop-up window with the SQL execution plan will appear.

For more details on the SQL execution plan, please, refer to [SQL execution plan].

To copy the certain SQL into a clipboard, proceed as follows.

- After selecting the SQL, right-click on it

Select [Copy] from the contextual menu, and the SQL and the parameter will be copied.

8.3.5.4. File

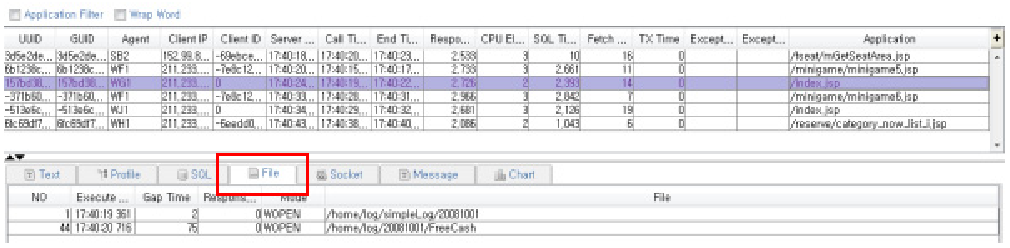
In the file tab, the file message type of the profile item for the transaction selected from the transaction list is displayed. You can check the file IO list executed by the transaction.

The following table describes the file list column.

Table 8-5: File List Column

Column	Descriptions
NO	The sharing number for the profile item. It helps you to find the profile item in the text or profile tab.
Execution time	The time when the profile item begins. It is based on the Java application time where the JENNIFER agent is installed.
Gap time	The difference between the start times of the current and previous profile item
Elapsed time	The time spent by the profile item
Mode	[WOPEN] is write mode, and [ROPEN] is read mode.
File	The path and name of the file that is the target of file IO

Figure 8-23:File Tab



UUID	GUID	Agent	Client IP	Client ID	Server ...	Call Ti...	End Ti...	Respo...	CPU El...	SQL Ti...	Fetch ...	TX Time	Except...	Except...	Application
3d5e2de...	3d5e2de...	SB2	152.99.8...	-69ebce...	17:40:18...	17:40:20...	17:40:23...	2,533	3	10	16	0			/tseat/mGetSeatArea.jsp
6b1238c...	6b1238c...	WF1	211.233....	-7e8c12...	17:40:20...	17:40:15...	17:40:17...	2,733	3	2,661	11	0			/minigame/minigame5.jsp
157bd38...	157bd38...	WG1	211.233....	0	17:40:24...	17:40:19...	17:40:22...	2,726	2	2,393	14	0			/index.jsp
-371b60...	-371b60...	WF1	211.233....	-7e8c12...	17:40:33...	17:40:28...	17:40:31...	2,966	3	2,842	7	0			/minigame/minigame6.jsp
-513a6c...	-513a6c...	WJ1	211.233....	0	17:40:34...	17:40:29...	17:40:32...	2,681	3	2,126	19	0			/index.jsp
8lc69df7...	8lc69df7...	WH1	211.233....	-6eedd0...	17:40:43...	17:40:38...	17:40:40...	2,086	2	1,043	6	0			/reserve/category_now_list_i.jsp

NO	Execute ...	Gap Time	Respons...	Mode	File
1	17:40:19 361	2	0	WOPEN	/home/log/simpleLog/20081001
44	17:40:20 716	75	0	WOPEN	/home/log/20081001/FreeCash

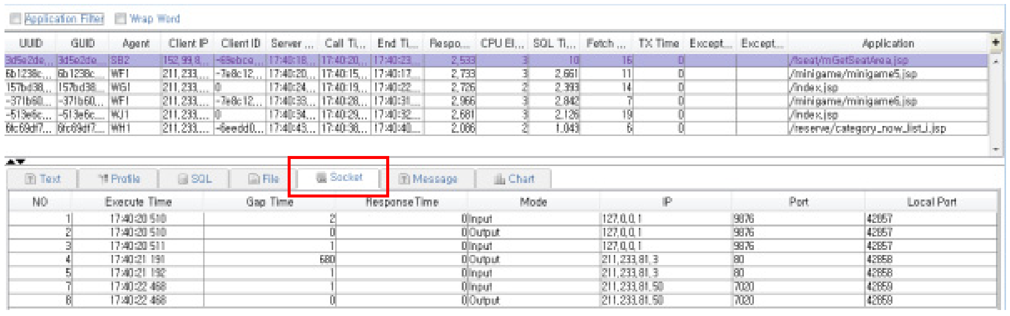
8.3.5.5. Socket

In the socket tab, the socket messages type of the profile item for the transaction selected from the transaction list is displayed. You can check the socket IO list executed by the transaction. The following table describes the socket list columns.

Table 8-6: Socket List Column

Column	Descriptions
NO	The sharing number for the profile item. It helps you to find the profile item in the text or profile tab.
Execution time	The time when the profile item begins. It is based on the Java application time where the JENNIFER agent is installed.
Gap time	The difference between the start times of the current and previous profile item
Elapsed time	The time spent by the profile item
Mode	[Output] is write mode, and [Input] is read mode.
IP	The IP address of the application that is the target of socket IO
Port	The port number for the application that is the target of socket IO.
Local port	The local port number used to communicate with the application that is the target of socket IO.

Figure 8-24:Socket Tab



UUID	GUID	Agent	Client IP	Client ID	Server ...	Call Ti...	End Ti...	Respo...	CPU El...	SQL Ti...	Fetch ...	TX Time	Except...	Except...	Application
3d5e2de...	3d5e2de...	SB2	152.99.8...	-69ebce...	17:40:18...	17:40:20...	17:40:23...	2,533	3	10	16	0			/tseat/mGetSeatArea.jsp
6b1238c...	6b1238c...	WF1	211.233....	-7e8c12...	17:40:20...	17:40:15...	17:40:17...	2,733	3	2,661	11	0			/minigame/minigame5.jsp
157bd38...	157bd38...	WG1	211.233....	0	17:40:24...	17:40:19...	17:40:22...	2,726	2	2,393	14	0			/index.jsp
-371b60...	-371b60...	WF1	211.233....	-7e8c12...	17:40:33...	17:40:28...	17:40:31...	2,966	3	2,842	7	0			/minigame/minigame6.jsp
-513e6c...	-513e6c...	WJ1	211.233....	0	17:40:34...	17:40:29...	17:40:32...	2,681	3	2,126	19	0			/index.jsp
6c69df7...	6c69df7...	WH1	211.233....	-6eedd0...	17:40:43...	17:40:38...	17:40:40...	2,086	2	1,043	6	0			/reserve/category_now_list_l.jsp

NO	Execute Time	Gap Time	ResponseTime	Mode	IP	Port	Local Port
1	17:40:20 510	2	0	Input	127.0.0.1	9876	42857
2	17:40:20 510	0	0	Output	127.0.0.1	9876	42857
3	17:40:20 511	1	0	Input	127.0.0.1	9876	42857
4	17:40:21 191	680	0	Output	211.233.81.3	80	42858
5	17:40:21 192	1	0	Input	211.233.81.3	80	42858
7	17:40:22 468	1	0	Input	211.233.81.50	7020	42859
8	17:40:22 468	0	0	Output	211.233.81.50	7020	42859

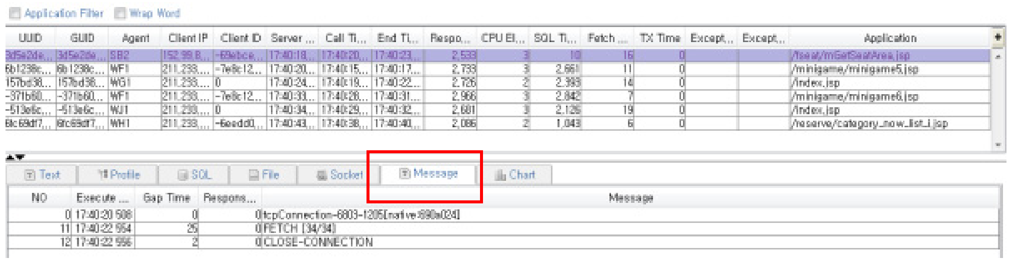
8.3.5.6. Message

In the message tab, the message type of the profile item for the transaction selected from the transaction list is displayed.

Notice: Profile items like SQL, files or sockets belong to the message type.However, these do not appear in the message tab.

The following table describes message list columns.

Figure 8-25:Message Tab



UUID	GUID	Agent	Client IP	Client ID	Server ...	Call Ti...	End Ti...	Respo...	CPU El...	SQL Ti...	Fetch ...	TX Time	Except...	Except...	Application
3d5e2de...	3d5e2de...	SB2	152.99.8...	-69ebce...	17:40:18...	17:40:20...	17:40:23...	2,533	3	10	16	0			/tseat/mGetSeatArea.jsp
6b1238c...	6b1238c...	WF1	211.233....	-7e8c12...	17:40:20...	17:40:15...	17:40:17...	2,733	3	2,661	11	0			/minigame/minigame5.jsp
157bd38...	157bd38...	WG1	211.233....	0	17:40:24...	17:40:19...	17:40:22...	2,726	2	2,393	14	0			/index.jsp
-371b60...	-371b60...	WF1	211.233....	-7e8c12...	17:40:33...	17:40:28...	17:40:31...	2,966	3	2,842	7	0			/minigame/minigame6.jsp
-513e6c...	-513e6c...	WJ1	211.233....	0	17:40:34...	17:40:29...	17:40:32...	2,681	3	2,126	19	0			/index.jsp
8c69df7...	8fc69df7...	WH1	211.233....	-6eedd0...	17:40:43...	17:40:38...	17:40:40...	2,086	2	1,043	6	0			/reserve/category_now_list_i.jsp

NO	Execute ...	Gap Time	Respons...	Message
0	17:40:20 508	0	0	tcpConnection-6803-1205[native:690a024]
11	17:40:22 954	25	0	FETCH [34/34]
12	17:40:22 956	2	0	CLOSE-CONNECTION

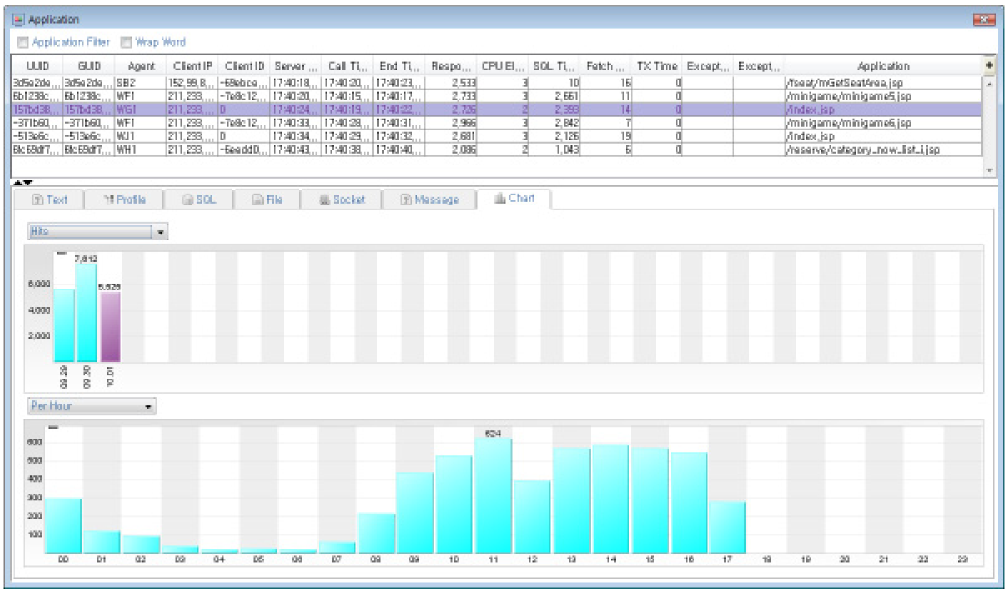
To check the detailed information of the message, proceed as follows.

- After selecting the message, right click on it.
- Select [Detailed info] from the contextual menu, and a new pop-up window will appear with detailed information regarding the message.

8.3.5.7. Chart

In the chart tab, the daily and hourly performance data for the application of the transaction selected from the transaction list are displayed.

Figure 8-26:Chart Tab



UUID	GUID	Agent	Client IP	Client ID	Server ...	Call Ti...	End Ti...	Respo...	CPU El...	SQL Ti...	Fetch ...	TX Time	Except...	Except...	Application
3d5e2de...	3d5e2de...	SB2	152.99.8...	-69ebce...	17:40:18...	17:40:20...	17:40:23...	2,533	3	10	16	0			/tseat/mGetSeatArea.jsp
6b1238c...	6b1238c...	WF1	211.233....	-7e8c12...	17:40:20...	17:40:15...	17:40:17...	2,733	3	2,661	11	0			/minigame/minigame5.jsp
157bd38...	157bd38...	WG1	211.233....	0	17:40:24...	17:40:19...	17:40:22...	2,726	2	2,393	14	0			/index.jsp
-371b60...	-371b60...	WF1	211.233....	-7e8c12...	17:40:33...	17:40:28...	17:40:31...	2,966	3	2,842	7	0			/minigame/minigame6.jsp
-513e6c...	-513e6c...	WJ1	211.233....	0	17:40:34...	17:40:29...	17:40:32...	2,681	3	2,126	19	0			/index.jsp
8c69df7...	8c69df7...	WH1	211.233....	-6eedd0...	17:40:43...	17:40:38...	17:40:40...	2,086	2	1,043	6	0			/reserve/category_now_list_i.jsp

- Daily chart - The upper chart displays the daily performance data. Using the drop-down list, you can select the desired performance data item.

Table 8-7: Performance Data Items

Item	Description
# of calls	The chart displays the number of calls.
# of failures	The chart displays the number of failures.
Sum of response times	The chart displays the sum of response times.
Average response time	The chart displays the average response time.
Minimum response	The chart displays the minimum response time.
Maximum response	The chart displays the maximum response time.
CPU time	The chart displays the CPU time.
CPU(tpmC)	The chart displays the CPU(tpmC) item.

- Daily chart - If you select a certain date from the upper daily chart, then the lower chart will display the daily performance data for the selected date. If you select [hour] from the drop-down list, then the hourly bar chart will be displayed, and if you select [unit time (10 min)], then the 10-minute line chart will be displayed.

8.4. Root-cause Isolation through X-View

The strong point of JENNIFER is that if any abnormality is discovered in the distribution pattern of responses, it is possible to selectively analyze the transactions. By comparing the performance information of the selected transactions, you can determine the origin of the problem. Here are some useful tips for analyzing the performance information of the listed transactions.

1. If the listed transactions have the same agent, then the problem is probably associated with the internal WAS/JAVA process. If they do not, then it is probably due to the external process.
2. If the listed transactions have the same application name, then the problem is probably associated with specific tasks.
3. If the SQL/Fetch time occupies significant portions of the Elapsed Time, then the problem is probably a database problem. If the Tx time occupies, significant portions of the Elasped Time, then the problem is probably due to an external system other than the DB.
4. If you have a DB connection problem, then the SQL/Fetch Time may not be high. A DB connection problem is either due to the DB or internal issues of the WAS. You can determine this by checking the agent in the list.
5. If the CPU Time is high, then it is likely that there exists some logical overhead (text string control, looping logics).

6. If the application names share the same JSP and either of them has high CPU time, then it is likely to be a JSP compile problem. In this case, you can check the FILE OPEN message in the profile content.

Notice: If the profile setting is not performed accurately, then some information in the profile may differ from the actual fact. For this reason, you should check for accuracy. For instance, if the installed jennifer20.so/dll file is not in compliance with the OS, then you may see some abnormal CPU time.

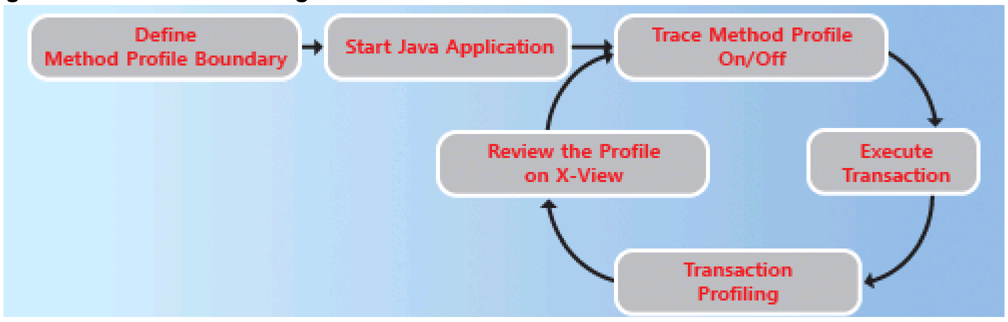
8.5. Java Method Profiling

JENNIFER makes a distinction between interconnection profiling and method profiling. Interconnection profiling implies that the JDBC/external transactions are tracked, while method profiling implies the tracking of response times or parameters. Method profiling can be monitored through the process described below.

Notice: Method profiling has a lower priority than interconnection profiling. For this reason, the class designated in the interconnection profile should not be included in the method profile.

Method profiling can be done in the following ways.

Figure 8-27:Method Profiling Procedures


Define Method Profile Boundary
Start Java Application
Trace Method Profile On/Off
Execute Transaction
Transaction Profiling
Review the Profile on X-View

- Setting the method profiling range – If you attempt to profile every method of every class, then it can cause degradations in the performance. Therefore, it is more efficient to profile the necessary methods of the classes only. The method profiling range setting phase is where you determined which method of which class should be profiled.
- Starting the Java application – If you want the newly changed method profiling range to be reflected in the system, you should restart the Java application. But if the JENNFER Agent is installed as javaagent then you can change the method profiling range setting without restarting the Java application.

- Controlling the method profile – You can specify whether to profile a certain method of the class included in the method profiling range. If this method profiling function is deactivated, then no method profiling is done.
- Executing and profiling a transaction – When a transaction is executed, profile data collection also takes place. The collected profile data is transmitted to the JENNIFER Server.
- Checking the profile data – Using the X-View chart, you can check the profile data.

8.5.1. Setting the Range of Method Profile

Using the profile starting option of your JENNIFER Agent, you can set the method profile range. To reflect the modified option in the system, you should restart the Java application where the JENNIFER Agent is installed..

Notice: java, org.jsn, org.apache.jennifer. com.javaservice . These packages can't be profiled. But the javax package can be profiled.

If you want to profile all the method of the example.BusinessManager class, change the setting as follows.

```
profile_class = example.BusinessManager
```

In addition, if you want to profile all the methods of the example.DomainManager class, then use a semi colon[;] as a separator to set the profile_class option. Beside the profile_class option, if you want to set more than one entity in every option related to method profiling ranges, then use a semi colon[;] as a separator.

```
profile_class = example.BusinessManager;example.DomainManager
```

The unit of method profiling is a specific method of a specific class. In other words, if you only set a class as shown earlier, then all the methods in the example.BusinessManager class will be included in the method profiling range. Therefore, if you want to include only a specific method in the method profiling range, then you should use the JENNIFER Agent's profile_target_method option to set the method.

```
profile_target_method = execute
```

By the way, if there are some methods with the same names among the classes set as the method profile range, and if you want to include a specific method of a specific class among them in the method profile range, then set it specifically as follows.

```
profile_target_method = example.BusinessManager.execute
```

To the contrary, if you want to exclude a specific method of a specific class from the method profiling range, then you use the JENNIFER Agent's profile_ignore_method option.

```
profile_ignore_method = example.BusinessManager.someMethod
```

You can also use a method accessor to set the method profile range. For instance, if you want to include a method having no public accessor and accessor in the method profiling range, then set it as follows.

```
profile_access_method = public;none
```

The following values can be set in the profile_access_method option.

- public - public accessor
- protected - protected accessor
- private - private accessor
- none – no accessor

And, if you want to include every class inheriting a specific class in the method profiling range, then use the JENNIFER Agent's profile_client_super option. For instance, if you want to include every class inheriting example.ejb.BaseSessionBean in the method profiling range, then set it as follows.

```
profile_super = example.ejb.BaseSessionBean
```

However, in this case, only the class directly inhering it will be included in the method profile range. If class A inherits the example.ejb.BaseSessionBean class and class B inherits class A , then class A is included in the method profiling range but class B is not. And, if you want to include every class that implements a specific interface in the method profiling range, then you can use the JENNIFER Agent's profile_interface option. For instance, if you want to include every class that implements the example.eai.IBusinessManager interface in the method profiling range, then set it as follows.

```
profile_interface = example.eai.IBusinessManager
```

However, in this case, only the class directly implementing it will be included in the method profiling range. If class A implements the example.eai.IBusinessManager interface and class B inherits class A , then class A is included in the method profiling range but class B is not.

And, using a class name, you can set the method profiling range. In this case, you can use the JENNIFER Agent's options such as profile_prefix, profile_postfix, profile_ignore_prefix, or profile_ignore_postfix. If you want to include every class whose name starts with example.biz, set it as follows.

```
profile_prefix = example.biz
```

In addition, if you want to include every class whose name ends with ‘Manager’ in the method profiling range, then set it as follows.

```
profile_postfix = Manager
```

If you want to exclude every class whose name starts with “OOO” from the method profiling range, then you can use the JENNIFER Agent’s profile_ignore_prefix option. This option can override all other options.

```
profile_ignore_prefix =
```

If you want to exclude every class whose name ends with “OOO” from the method profiling range, then you can use the JENNIFER Agent’s profile_ignore_postfix option. This option can override all other options.

```
profile_ignore_postfix =
```

You can set the JENNIFER Agent’s options whose names start with ‘profile’ in various ways. When setting the method profiling range, you can use the profile_target_method option to only set the methods to be included in the actual method profiling range. For instance, let’s consider the pkg.ClassA & pkg.ClassB class. Now, let’s assume that the run method of the ClassA class and the process method of the ClassB class should be included in the method profiling range. If the run method also exists in the ClassB class and you want to exclude it from the method profiling range, then set it as follows.

```
profile_class = pkg.ClassA;pkg.ClassB
profile_target_method = run;process
profile_ignore_method = pkg.ClassB.run
```

Or, you can set it as follows, too.

```
profile_class = pkg.ClassA;pkg.ClassB
profile_target_method = pkg.ClassA.run;pkg.ClassB.process
```

You can use the method size to set the method profiling range as well. Using the JENNIFER Agent’s profile_method_max_byte option, you can set the max method size to be included in the method profiling range. The default is 32000 and the unit is in bytes.

```
profile_method_max_byte = 32000
```

Since the Java method size can’t exceed 64 KB, an error will occur for a method with a

```
profile_class = example.BusinessManager
```

size greater than 64 KB. For method profiling, JENNIFER inserts a tracking code in the method. So, the method size tends to increase than before. Therefore, do not exceed 50KB when using this option.

Similarly, you can also set the minimum method size to be included in the method profiling range by using the JENNIFER Agent’s profile_method_min_byte option. The default is 0 and the unit is in bytes. .

```
profile_method_min_byte = 0
```

Notice: Based on the method size, you can exclude a method with simple logic such as getter or setter from the method profiling range.

8.5.2. Profiling Control

JENNIFER 4 class profiling is based on transactions. For this reason, as the data is collected from a single transaction, it is likely that too much data will be collected. To prevent such a problem, you should set profile_buffer_size. The default is 1000. .

```
profile_buffer_size = 1000
```

Setting the profile range does not mean that the response time information is automatically collected. In this case, you should set profile_default_on=true. The default is false.

```
profile_default_on = true
```

To prevent performance degradations caused by method profiling, you are not recommended to set this option to true.

Even after setting the profile_default_on option to false, you can either turn on/off the method profiling option for a specific class without restarting the Java application where the JENNIFER Agent is installed. For more details, please refer to [Dynamic Profiling(88 page)].

The profile_default_on option of the JENNIFER agent turn on/off the profile status about every class. However, you can also set the certain profile status seperately by setting a certain option. If you set the profile status of the class that is set by profile_class option of the JENNIFER agent, use the profile_class_on option of the JENNIFER agent.
profile_class_on = true

If you set the profile status of the class that is set by profile_super of the JENNIFER agent, use the profile_super_on of the JENNIFER agent.

```
profile_super_on = true
```

If you set the profile status of the class that is set by profile_interface option of the JENNIFER agent, use the profile_interface_on option of the JENNIFER agent.

```
profile_interface_on = true
```

8.5.3. Boot Class Profiling

If the JENNIFER Agent is installed as javaagent, then you can also profile a class set in -Xbootclasspath But, even in this case, the java package can't be profiled. For this purpose, you can set the JENNIFER Agent's enable_hooking_boot option to true. The default is false.

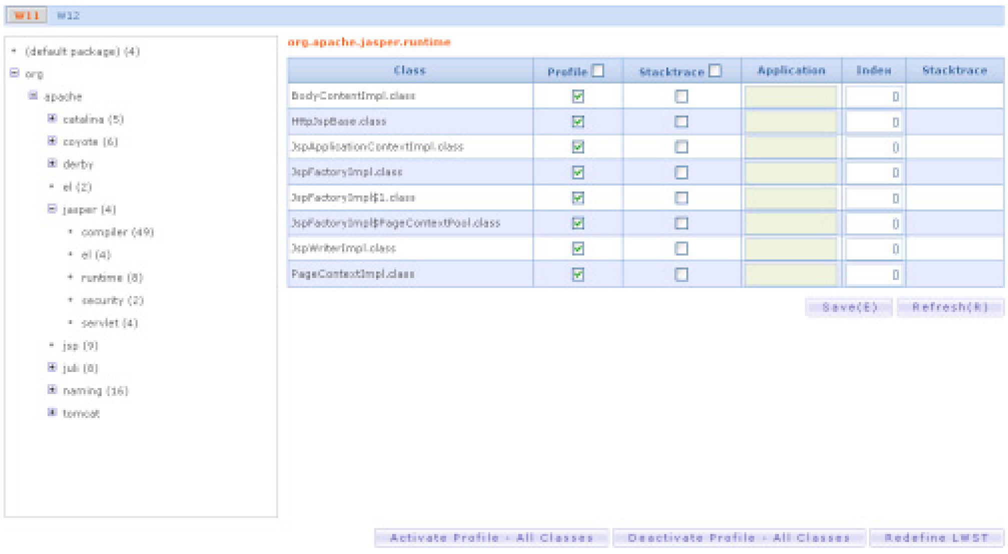
```
enable_hooking_boot = true
```

In general, you are supposed to set a class having system characteristics in the Boot Class path. Therefore, only if the business class is set in the Boot Class path, you are recommended to use it as an exception.

8.5.4. Dynamic Profiling

Dynamic profiling is when you dynamically set the method profiling range or activating/deactivating profiling of the classes included in the method profiling range without starting the Java application where the JENNIFER Agent is installed. But, you can change the method profiling range without restarting the Java application only if the JENNIFER Agent is installed as javaagent. In the [Real Time Monitoring | Profile] menu, you can deactivate or activate profiling of the classes included in the method profiling range.

Figure 8-28:Dynamic Profiling



org.apache.jasper.runtime

Class	Profile ☐	Stacktrace ☐	Application	Index	Stacktrace
BodyContentImpl.class	✓	☐		0	
HttpJspBase.class	✓	☐		0	
JspApplicationContextImpl.class	✓	☐		0	
JspFactoryImpl.class	✓	☐		0	
JspFactoryImpl$1.class	✓	☐		0	
JspFactoryImpl$PageContextPool.class	✓	☐		0	
JspWriterImpl.class	✓	☐		0	
PageContextImpl.class	✓	☐		0	

You can check whether a certain class is included in the method profiling range in the following ways. In the JENNIFER Agent selection area, select a JENNIFER Agent where you want to check the method profiling range. In the left side of the screen, the Java application package where the JENNIFER Agent is installed, is displayed as a tree. Select a specific package in the package tree, the list of the classes included in the method profiling range will appear in the right side of the screen. If the profile column for a class in the list is checked, then it means that it is being profiled.

Notice: Not all the classes in the package will appear. Only the classes included in the method profile range set by the option whose name starts with ‘profile’ will appear.

You can specify whether to activate or deactivate profiling of a method of a specific class in the following ways.

- Select a package form the package tree, where the class to be activated or deactivated is included.
- After activating or deactivating method profiling in the class list, press the [Save] button in the bottom.

In the package tree, select the default package to display the four classes. You can use them to control the operational method for the JENNIFER Agent.

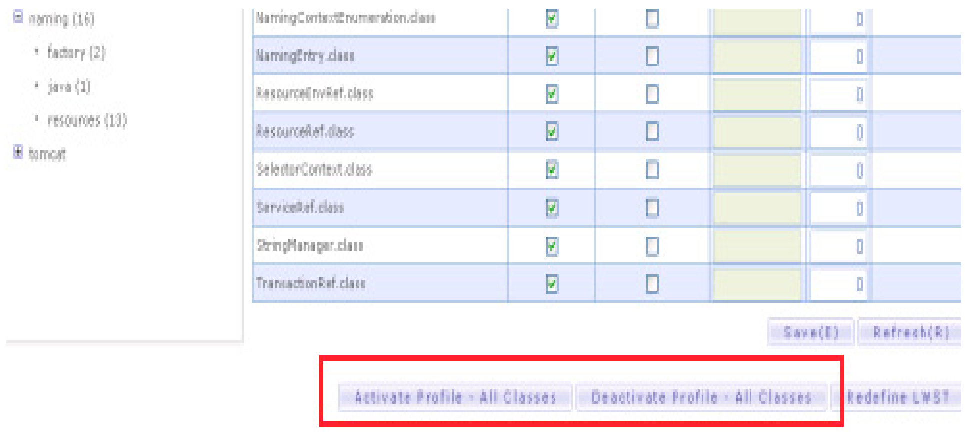
Table 8-8: Class Items

Basic Profile Status	Class	Description
true	PROFILE_MESSAGE	If you turn off profiling, then the message type profile items are not collected. But the messages related to getConnection are not affected.

Basic Profile Status	Class	Description
true	PROFILE_PARAM	The chart displays the number of failures.
true	PROFILE_SQL	If you turn off profiling, then the SQL type profile items are not collected.
false	PROFILE_USER_CONTROL	It is for functional expansions in the future, but the current functions are not affected.

You can use the [Entire Profile Setting] and [Entire Profile Cancel] buttons in the bottom of the screen to activate/deactivate profiling of every class included in the method profiling range.

Figure 8-29:Activation of Profiling



naming (16)
- factory (2)
- java (1)
- resources (13)

tomcat

NamingContextEnumeration.class	✓	☐		0
NamingEntry.class	✓	☐		0
ResourceEnvRef.class	✓	☐		0
ResourceRef.class	✓	☐		0
SelectorContext.class	✓	☐		0
ServiceRef.class	✓	☐		0
StringManager.class	✓	☐		0
TransactionRef.class	✓	☐		0

Save(E)
Refresh(R)
Activate Profile - All Classes
Deactivate Profile - All Classes
Redefine LWST

If the JENNIFER Agent is installed as javaagent, you can change the method profiling range without restarting the Java application. This is called LWST resetting. The following is the LWST resetting method. In the bottom of the [Real Time Monitoring | Profile] menu, press the [LWST Resetting] button to launch the pop up screen for LWST resetting. The pop up screen for LWST resetting is comprised of a loading class list tab and a configuration setting tab. In the configuration setting tab, use the options whose names start with ‘profile’ to change the method profiling range. In the loading class list tab of the pop up window for LWST resetting, select a class where you want to reset the profiling range and click on the [Apply] button in the bottom. You can perform searches by class name. Close the pop up window for LWST resetting, and check whether the changes are reflected in the system by using the [Real Time Monitoring | Profile] menu. LWST resetting is only applied to the selected classes, not all the classes. For instance, let’s consider the example.ClassA class and the example.ClassB class which are not included in the method profiling range. For LWST resetting, you can modify the profile_class option as follows.

```
profile_class = example.ClassA;example.ClassB
```

In the loading class list tab of the pop up window for LWST resetting, select the example.CassA class only and then click on the [Apply] button. In this case, the example.ClassA class is included in the method profiling range but the example.ClassB class is not. But if you restart the Java application where the JENNIFER Agent is installed, then the example.ClassB class will be also included in the method profiling range.

8.5.5. Dynamic Stack Trace

If you try to profile all the methods of all the classes, it will cause an overloading problem and not be efficient since only a small portion of the collected profile data is used for analysis. Therefore, it is more efficient to profile the classes directly related to the application with performance or functional problems. By the way, if you are about to use the source code not developed on your own or the external package, it is very difficult to examine the classes directly related to specific applications. The dynamic stack trace is a function to help you find classes directly related to specific applications. Among the classes included in the method profiling range, it tracks the Java stack based on an arbitrary class. It helps you to understand the transaction flows and easily select the profiling target classes.

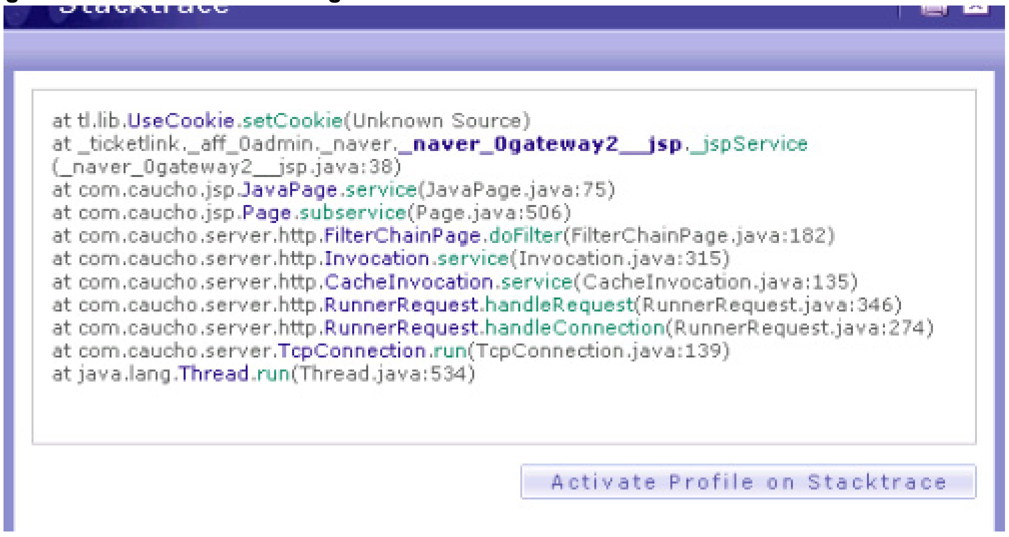
Notice: To use the dynamic stack trace function, you need to understand one or more class used by a specific application.

The dynamic stack trace function is used as follows.

- Activate profiling of the method of an arbitrary class used by the application having performance or functional problems.
- After checking the stack trace check box for the class, click on the [Save] button.
- Click on the [Refresh] button repeatedly until the Java stack trace is displayed.
- When an arbitrary transaction uses the class, then the stack trace is recorded and the stack trace status is changed to off.

In the following way, you can activate profiling of the method of every class included in the stack trace. Clicking on the collected stack trace, the pop up window will appear. Click on the [Profile Setting] button in the bottom. The profile status of every class displayed in the stack trace among all the classes included in the method profiling range will be changed to ‘on’.

Figure 8-30:Stack Trace Setting



```
at tl.lib.UseCookie.setCookie(Unknown Source)
at _ticketlink._aff_0admin._naver._naver_0gateway2__jsp._jspService
(_naver_0gateway2__jsp.java:38)
at com.caucho.jsp.JavaPage.service(JavaPage.java:75)
at com.caucho.jsp.Page.subservice(Page.java:506)
at com.caucho.server.http.FilterChainPage.doFilter(FilterChainPage.java:182)
at com.caucho.server.http.Invocation.service(Invocation.java:315)
at com.caucho.server.http.CacheInvocation.service(CacheInvocation.java:135)
at com.caucho.server.http.RunnerRequest.handleRequest(RunnerRequest.java:346)
at com.caucho.server.http.RunnerRequest.handleConnection(RunnerRequest.java:274)
at com.caucho.server.TcpConnection.run(TcpConnection.java:139)
at java.lang.Thread.run(Thread.java:534)
```

Basically, the stack trace for the first transaction that used the class will be recorded. You can also record the stack trace related to the transaction executed by a specific application. You can do so as follows. Click on the application field for stack trace collections in the class list. The pop up window with an application list will appear. In the pop up window with an allocation list, click on the application name for stack trace recording. Then the application name will be automatically entered in the application field. Click on the [Save] button in the bottom of the class list. You can also record the stack trace for an arbitrary class used by a background thread which is not a Java threat that executes a transaction. Enter '-1' in the index field in for the class in the class list where you want to record to stack trace. Click on the [Save] button in the bottom of the class list.

8.5.6. Parameter/Return Tracking

When performing method profiling, you can also collect the parameter values or returned values for a specific method. But, the type of parameter or returned value should be java.lang.String. For instance, you can trace the method parameter or returned value for the class as follows.

```
package pkg;

public class ClassC {

    public void ptrace1(String value) {
        System.out.println("ClassC.process:" + value);
    }

    public void ptrace2(int x, String value) {
       System.out.println("ClassC.process:" + value);
    }

    public String rtrace() {
        return "It's the return value";
    }
}
```

If you want to trace the method parameter value for ptrace1 of the ClassC class, then set the JENNIFER Agent's lwst_profile_method_using_param option as follows.

```
lwst_profile_method_using_param = pkg.ClassC.ptrace1(String)
```

If the method has multiple java.lang.String type parameters, only the first string type parameter is collected.

A class name and a method name are separated by a dot[.] and a class name should include a package name. A method's parameter type should be defined. If you want to set more than one method, use semicolon[;] as a separator. For instance, if you want to trace the ptrace2 method parameter, change the setting as follows.

```
lwst_profile_method_using_param =
pkg.ClassC.ptrace1(String);pkg.ClassC.ptrace2(int,String)
```

If the method has multiple java.lang.String type parameters, only the first string type parameter is collected.

A class name and a method name are separated by a dot[.] and a class name should include a package name. A method's parameter type should be defined. If you want to set more than one method, use semicolon[;] as a separator. For instance, if you want to trace the ptrace2 method parameter, change the setting as follows.

```
lwst_profile_method_using_param =
pkg.ClassC.ptrace1(String);pkg.ClassC.ptrace2(int,String)
```

If you want to set more than one method, then use a semicolon [;] as a separator. Once the setting is changed as above, you can now check the following details in the profile data.

```
PARAM[test param] ClassC.ptrace1
PARAM[test param] ClassC.ptrace2
RETURN[It's the return value] ClassC.rtrace
```

8.5.7. Tracking the Called Method

JENNIFER allows you to determine whether a specific method in the method logic is called. This function is very useful to analyze the method content in details. For instance, let's say we have a class/method called ClassD.process.

Table 8-9: ClassD Source

```
package pkg;

public class ClassD {
   public static void process() {
      IClass c = new ClassE();
      //some logic
      c.proc();
      //some logic
      c.proc();
      //some logic
   }
}
```

The process method of the ClassD class is comprised as follows.

1. Execute step 1.
2. After creating the ClassE object, call the run method.
3. Execute step 2.

4. After creating the ClassF object, call the run method.

In steps 1 and 2, various logics are executed and many methods can be called. At this time, if you want to measure the execution time of step 1/2, not the individual method response time, then you need to user the JENNIFER Agent’s profile_call option. First of all, using the profile_class option, include ClassD in the method profiling range.

```
profile_class = pkg.ClassD
```

And, set the profile_call option as follows. If more than one, then use a semicolon [;] as a separator.

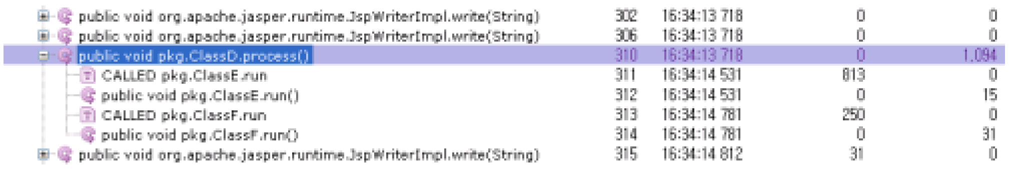
```
profile_call = pkg.ClassE.run;pkg.ClassF.run
```

If the same method name is used, then use a star mark[*] to set it.

```
profile_call = *.run
```

Once the setting is changes as above, you can check the following details in the profile data.

Figure 8-31:Profile Data



```
public void org.apache.jasper.runtime.JspWriterImpl.write(String)   302  16:34:13 718    0      0
public void org.apache.jasper.runtime.JspWriterImpl.write(String)   306  16:34:13 718    0      0
public void pkg.ClassD.process()                                    310  16:34:13 718    0  1,094
    CALLED pkg.ClassE.run                                           311  16:34:14 531  813      0
    public void pkg.ClassE.run()                                    312  16:34:14 531    0     15
    CALLED pkg.ClassF.run                                           313  16:34:14 781  250      0
    public void pkg.ClassF.run()                                    314  16:34:14 781    0     31
public void org.apache.jasper.runtime.JspWriterImpl.write(String)   315  16:34:14 812   31      0
```

As shown in the figure above, it took 504ms before calling the first proc and 101ms before calling the second proc.

The tracking function for the called method can be used to partition the logic area of the method into many segments and analyze them. However, after modifying the profile_call option, you need to restart the WAS/Java process. For this reason, it is more useful for a test system than for an operating system.

When tracking each region of the JSP file, you can directly use the profile API rather than profile_call.[Refer to high-level monitoring]

8.5.8. Extracting the User ID

You can extract the user ID using parameter/return value of the certain method.

If you want to extract the user ID using parameter value of the certain method, set the userid_param option of the JENNIFER agent. Extract the first parameter of java.lang.String type as a user ID.

```
userid_param = pkg.UserManager.setId(String)
```

If you want to extract the user ID using return value of the certain method, set the userid_return option of the JENNIFER agent. In this case, the return type should be java.lang.String.

```
userid_return = pkg.UserManager.getId()
```

You can check the extracted user ID in the X-View.

8.6. Checking the Loading Class

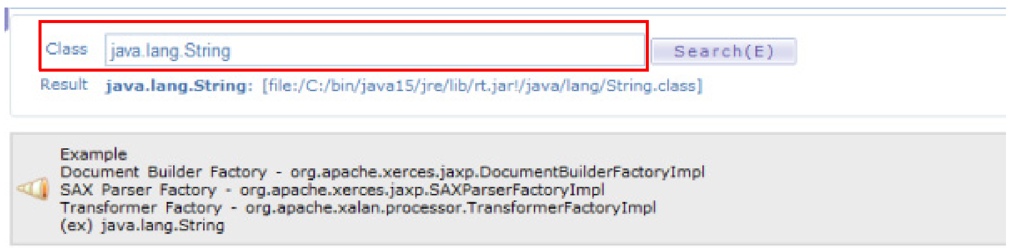
For optimal tracking of the transactions, it is necessary to find out in which JAT the class has been loaded, or what the subordinate relationship is.

8.6.1. Class/JAR Checking

Using the Utility | Class/Jar menu of the upper troubleshooting menu, you can check the location of a certain class in the monitoring target server and its shape. A class can be located in the JAR file or in a certain directory.

Due to the operating method of the class pass and loader, a single class can be located in many locations. Using this function, you can easily determine the location of the class used by the monitoring target server.

Figure 8-32:Searching for the JAR File of the Class



Class: java.lang.String

Result: java.lang.String: [file:/C:/bin/java15/jre/lib/rt.jar!/java/lang/String.class]

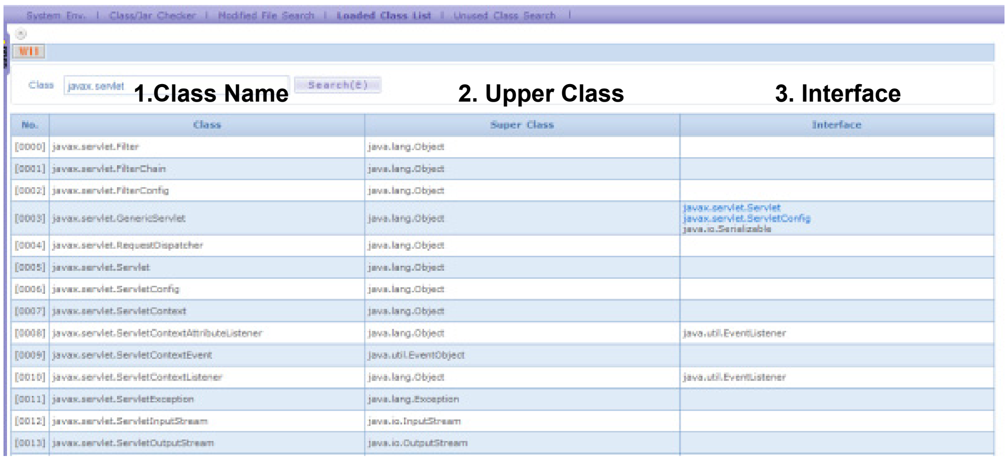
```
Example
Document Builder Factory - org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
SAX Parser Factory - org.apache.xerces.jaxp.SAXParserFactoryImpl
Transformer Factory - org.apache.xalan.processor.TransformerFactoryImpl
(ex) java.lang.String
```

After entering the full name of the class in the class name field, including the Java package name, press the search button. The location of the class file in the monitoring target server will then be displayed, as shown in the figure. In the example, the java.lang.String class is located in the core.jar file in the /WebSphere/AppServer/java/jre/lib directory of the monitoring target server.

8.6.2. Loading Class List

Using the **[troubleshooting | utility | loading class list]** menu, you can check every class list loaded by the class loader.

Figure 8-33:Loading Class List

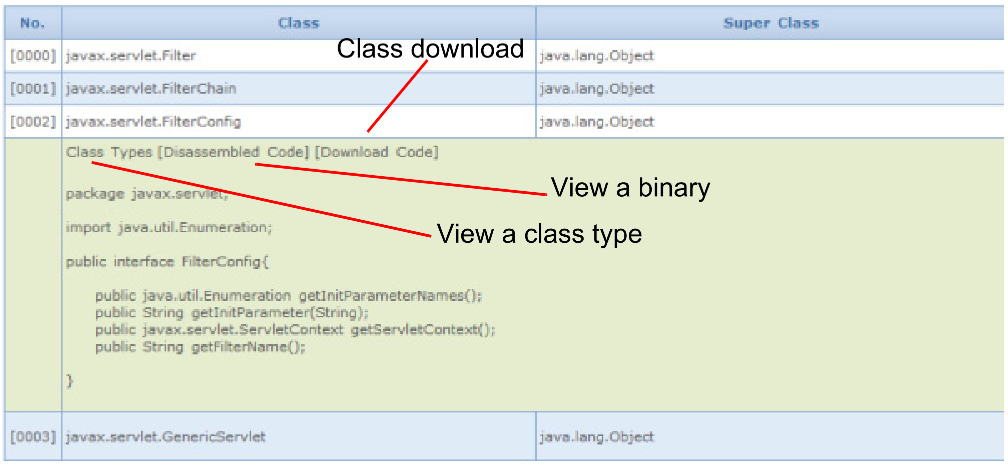

System Env. | Class/Jar Checker | Modified File Search | Loaded Class List | Unused Class Search |
WII
Class
javax.servlet
Search(E)
1.Class Name
2. Upper Class
3. Interface

No.	Class	Super Class	Interface
[0000]	javax.servlet.Filter	java.lang.Object	
[0001]	javax.servlet.FilterChain	java.lang.Object	
[0002]	javax.servlet.FilterConfig	java.lang.Object	
[0003]	javax.servlet.GenericServlet	java.lang.Object	javax.servlet.Servlet javax.servlet.ServletConfig java.io.Serializable
[0004]	javax.servlet.RequestDispatcher	java.lang.Object	
[0005]	javax.servlet.Servlet	java.lang.Object	
[0006]	javax.servlet.ServletConfig	java.lang.Object	
[0007]	javax.servlet.ServletContext	java.lang.Object	
[0008]	javax.servlet.ServletContextAttributeListener	java.lang.Object	java.util.EventListener
[0009]	javax.servlet.ServletContextEvent	java.util.EventObject	
[0010]	javax.servlet.ServletContextListener	java.lang.Object	java.util.EventListener
[0011]	javax.servlet.ServletException	java.lang.Exception	
[0012]	javax.servlet.ServletInputStream	java.io.InputStream	
[0013]	javax.servlet.ServletOutputStream	java.io.OutputStream	

- Class name - The name of the class that has been downloaded by the class loader. If you click on the name, you can see the detailed information of the class.
- Upper class - Shows the upper class of the class. If you click on the upper class name, then you will be moved to the class name.
- Interface - Shows the interface name of the class. If you click on the interface name, you will be moved to the interface name.

By clicking on the class name, you can check the detailed information of the class.

Figure 8-34:Detailed Class Data.



No.	Class	Super Class
[0000]	javax.servlet.Filter	java.lang.Object
[0001]	javax.servlet.FilterChain	java.lang.Object
[0002]	javax.servlet.FilterConfig	java.lang.Object
	Class Types [Disassembled Code] [Download Code]	
[0003]	javax.servlet.GenericServlet	java.lang.Object

```
package javax.servlet;

import java.util.Enumeration;

public interface FilterConfig{

    public java.util.Enumeration getInitParameterNames();
    public String getInitParameter(String);
    public javax.servlet.ServletContext getServletContext();
    public String getFilterName();

}
```

- View a class type - If you click on the link, the basic structure of the class will appear.
- View a binary - If you click on the link, then you can view the binary format of the class.
- Class download - If you click on the link, you can download the class file.

9 Resources, External Transactions and JDBC Monitoring in Java System

This chapter describes the method for monitoring H/W resources such as CPU and memory utilization, and external transactions and JDBC in interconnection with external applications.

9.1. CPU Monitoring

Using the JENNIFER agent, you can monitor the system resources such as CPU utilization and the Java process CPU utilization operated by the Java application and you can also monitor the CPU occupancy time by specific transaction.

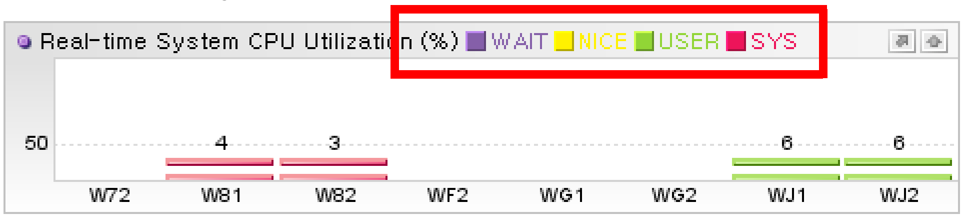
In addition, by using the WMOND module, you can monitor the arbitrary H/W CPU utilization per each CPU number, regardless of whether or not the JENNIFER agent is installed.

9.1.1. System CPU Utilization and Java Process CPU Utilization

The system CPU utilization rate is the ratio of CPU use by the system, while the Java process CPU utilization rate is the ratio of CPU use by the Java application in which the JENNIFER agent is installed.

The system CPU utilization and the Java process CPU utilization are the general performance data collected by the JENNIFER agent. The 30-second averages can be viewed on equalizer and runtime line charts.

Figure 9-1:Real Time System CPU Utilization Chart



However, the CPU utilization is displayed in each region of CPU use and an equalizer chart distinguishes the regions of CPU use with different colors. The following is a description of the regions of CPU use.

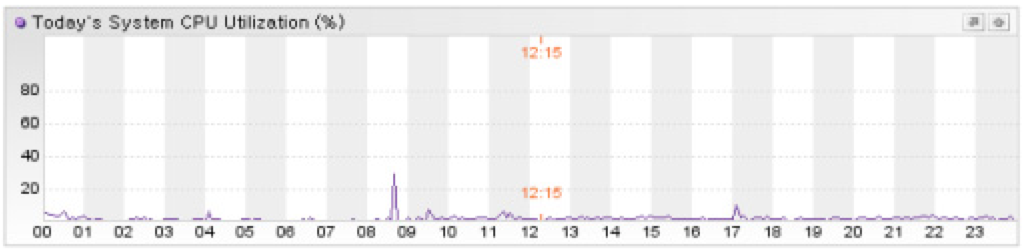
Table 9-1: Region of CPU Use

Regions of CPU use	Description
WAIT	CPU utilization related to I/O standby
NICE	CPU utilization by the user (application) with the NICE priority
USER	CPU utilization by the user (application)
SYS	CPU utilization by the system (kernel)

Notice: The equalizer chart distinguishes the regions of Java CPU utilization and individual CPU utilization collected by WMOND by using different colors.

In addition, the five-minute averages of system CPU utilization and Java process CPU utilization are stored in the SYS_CPU and JVM_CPU columns of the PERF_X_01~31 table. The CPU utilization trend for a certain date can be viewed on a line chart.

Figure 9-2:Today's System CPU Utilization



9.1.2. Transaction CPU Time

The CPU occupancy time by a transaction is the amount of CPU use time while a transaction is executed. It can be collected from the application processing statistical data and the X-view profile data.

Using tpmC, you can objectively compare the CPU occupancy times by transaction for each H/W. tpmC represents the maximum number of the TPC-C benchmark scenario of TPC (Transaction Processing Performance Council, http://www.tpc.org) processed in one minute, and it is the most popular way to measure the H/W (mainly, CPU) performance.

Since CPU processing capacity varies greatly between computers, the CPU occupancy time by a transaction alone may not objectively illustrate the extent to which the transaction affects the performance. For example, even if the CPU occupancy time by transaction A processed by the H/W with a large CPU processing capacity is less than the CPU occupancy time by transaction B processed by the H/W with a small CPU processing capacity, transaction A may affect the performance more severely than transaction B. In other words, if transaction A is processed by H/W with a lower capacity, then it could occupy more CPU time than transaction B.

Therefore, by using tpmC, you should understand the manner in which the transaction affects the performance when the CPU processing capacity of the H/W is fixed.

To achieve this, use the approximate_tpmc_on_this_system option of the JENNIFER agent to set the tpmC value for the H/W in which the JENNIFER agent is installed.

```
approximate_tpmc_on_this_system = 30000
```

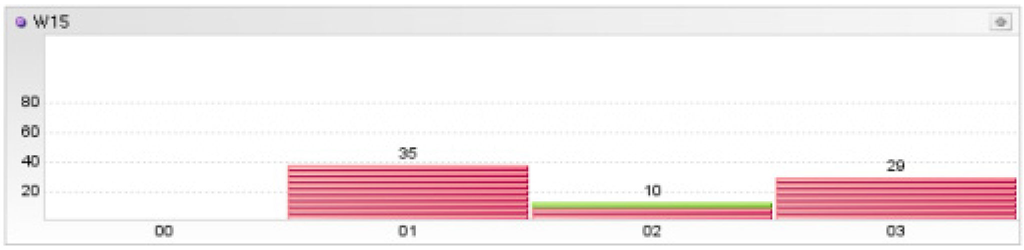
9.1.3. Individual CPU Utilization Monitoring with WMOND

WMOND is a module used to collect the individual CPU utilization data from arbitrary H/W. This module is developed in the C language and is very simple to use. As it is not related to installation of the JENNIFER agent. it is suitable for monitoring the CPU utilization by the H/W operated by a web server or database server.

Notice: Since WMOND collects the H/W CPU utilization data for each CPU, it is sometimes necessary to use WMOND in the H/W in which the JENNIFER agent is installed.

The individual CPU utilization data can be viewed on an equalizer chart. The following figure indicates that four CPUs are installed in the H/W monitored by WMOND.

Figure 9-3:Individual CPU Utilization



In addition, the individual CPU utilization is classified by region of CPU use, and is stored in the PERF_HOST table. By default, the storage period is five minutes. You can modify this using the perf_host_update_interval option of the JENNIFER server. It is expressed in milliseconds.

```
perf_host_update_interval = 300000
```

Notice: The permissible range is from 60000(1 minute) to 300000(5 minutes).

9.1.3.1. Installing and executing WMOND

Since the WMOND module is developed in the C language, install a version that is suitable for your OS. Currently, it can be used in Microsoft Windows, Sun Solaris, HP HP-UX, IBM AIX and various Linux environments.

If you want to use WMOND, first select the wmond file that is suitable for your OS from the JENNIFER_HOME/agent/wmond directory, and copy it to the H/W on which you want to monitor the CPU utilization for.

If your OS is unix or linux, grant an execution privilege to the copied wmond file. For example, if the wmond file is already copied to the /usr/bin directory, grant an execution privilege as follows:

```
chmod 755 /usr/bin/wmond
```

You can execute the WMOND module as follows:

```
wmond [JENNIFER server IP] [Port number] [WMOND ID]
```

- JENNIFER server IP - Assign the JENNIFER server IP address, to which the CPU utilization data collected by WMOND is sent.
- Port number - Use the UDP port number set in the server_udp_listen_port option of the JENNIFER server. The default is 6902.
- WMOND ID - A unique ID to distinguish the WMOND module. Use a unique ID that is different from all other WMOND IDs or JENNIFER agent IDs. It is comprised of alphanumeric characters, and has a maximum length of 3.

For example, you can execute WMOND as follows:

```
nohup wmond 127.0.0.1 6902 DB1 &
```

If you want WMOND to be automatically launched when the linux or unix system is started, or if you want the system to be automatically restarted when WMOND is stopped, include the following content in the /etc/inittab file.

```
wmond:2:respawn:/usr/bin/wmond 127.0.0.1 6902 DB1 > /dev/null 2>&1
```

For Solaris, add the /etc/rc2.d/S97wmond file. The filename must be unique.

```
/usr/bin/wmond  192.168.0.100 6902 DB1 > /dev/null 2>&1 &
```

Do not omit the [&] symbol.

9.1.3.2. Stopping WMOND and Issuing Alerts

If the CPU utilization data is not sent from WMOND within the time set in the agent_death_detection_time option of the JENNIFER server, the JENNIFER server assumes that the system is halted and issues the ERROR_SYSTEM_DOWN alert. It is expressed in milliseconds and default is 8000.

```
agent_death_detection_time = 8000
```

9.1.4. CPU Monitoring and Issuing Alerts

When the system CPU utilization and the Java process CPU utilization exceed the threshold, the JENNIFER server issues an alert. Please refer to Alert and Exception Monitoring for more details.

9.1.4.1. When the actual CPU utilization differs significantly from the CPU utilization collected by JENNIFER

JENNIFER uses the kernel API provided by the OS for CPU monitoring. However, when a multi core CPU is used in IBM AIX, sometimes the actual CPU utilization differs from the CPU utilization data collected by JENNIFER by more than 50%. In this case, request technical support by contacting tech@jennifersoft.com.

9.1.4.2. Native Module Tests for the JENNIFER Agent

The native module of the JENNIFER agent collects CPU utilization and memory utilization data. If the CPU utilization and memory utilization data is not displayed in the JENNIFER client, this means that the native module has not been installed correctly.

If the native module has been installed correctly, the following message will be recorded in the JENNIFER agent log file and the Java application output file.

```
libjennifer20.so(sl) shared library loaded successfully.
```

If the above message is not in the log file, then the native module configuration is incorrect, or an incorrect native module is being used. In this case, select an appropriate native module for your system.

To find an appropriate native module for your system, you can test the native module separately from the JENNIFER agent.

1. The native module for each type of OS can be found in the JENNIFER_HOME/agent/jni directory. Select the directory corresponding to your OS.
2. In this directory, you will find various native modules for your OS version and CPU. Find the correct module and change the name to libjennifer20.so(sl). In Microsoft Windows, change it to jennifer20.dll.
3. If you are using unix or linux, grant execution privileges to the native module.
4. Using the test.sh utility, you can test the native module. This utility uses Java, which means that you must register the JVM used by the Java application monitored by the JENNIFER agent in the path. If the native module and the test.sh utility are not in the same directory, open the test.sh file and change the content related to Djava.library.path.

5. Run the test.sh utility. If the following message appears, the native module is operating properly.

```
jennifer@jennifer1:~/agent/jni/linux$ ./test.sh

JENNIFER: pid=3553, ppid=3551, ncpu=4, cpucluck=100
JENNIFER SysProf libjennifer20.so(sl) shared library loaded
successfully.
JENNIFER4.1.0(2010-02-10) libjennifer20.so(sl) shared library loaded
successfully.
Process Id = 3553
Parent Process Id = 3551
Number of CPU = 4
Cpu Cluck Speed = 100
System Total Mem = 2026 MB
System Free Mem = 52 MB
Process Mem = 646 MB
Native Thread Id = -1210348656
Current Thread Cpu Time = 0 ms

          SYSTEM                       PROCESS
 USER   SYS  NICE  WAIT  IDLE  USER   SYS  NICE  WAIT  IDLE
  0.0   0.0   0.0   0.0 100.0   0.0   0.0   0.0   0.0 100.0
System Cpu Time = 8412973, 4834772, 9914, 0, 633993961,
Process Cpu Time = 2, 0, 1875530323, 0, -1,
Current Thread Cpu Time = 0 ms

          SYSTEM                       PROCESS
 USER   SYS  NICE  WAIT  IDLE  USER   SYS  NICE  WAIT  IDLE
  5.7   3.6   0.0   0.0  90.7   0.0   0.0   0.0   0.0 100.0
System Cpu Time = 8412998, 4834788, 9914, 0, 633994363,
Process Cpu Time = 2, 0, 1875530423, 0, -1,
Current Thread Cpu Time = 0 ms

          SYSTEM                       PROCESS
 USER   SYS  NICE  WAIT  IDLE  USER   SYS  NICE  WAIT  IDLE
  5.2   3.6   0.0   0.0  91.2   0.0   0.0   0.0   0.0 100.0
System Cpu Time = 8413021, 4834804, 9914, 0, 633994767,
Process Cpu Time = 2, 0, 1875530523, 0, -1,
Current Thread Cpu Time = 0 ms

System.exit(0)
```

If it does not operate properly after you have compiled all the files, compile the native module in the appropriate manner for your system. In this case, request technical support at tech@jennifersoft.com.

If it does not operate properly after you have compiled all the files, compile the native module in the appropriate manner for your system. In this case, request technical support at tech@jennifersoft.com.

9.2. Memory Monitoring

Using the JENNIFER agent, you can monitor the memory utilization of the H/W system operated by the Java application and the memory utilization of the Java process operated by the Java application. In addition, you can monitor the total Java heap memory and the Java heap memory utilization by the Java application.

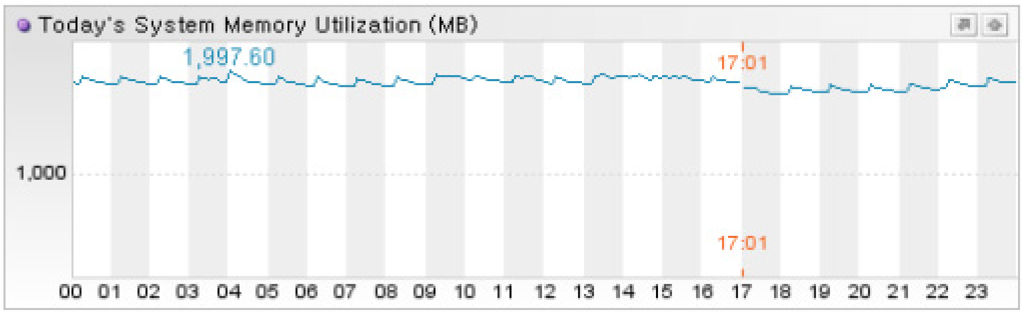
9.2.1. System Memory Utilization and Java Process Memory Utilization

The system memory utilization is the memory utilized by the system, while the Java process memory utilization is the memory utilization by the Java application in which the JENNIFER agent is installed. It is expressed in MB.

The system memory utilization and the Java process memory utilization is the general performance data collected by the JENNIFER agent. The thirty-second averages can be viewed on a runtime line chart.

In addition, the five-minute average of the system memory utilization and the Java process memory utilization is saved in the SYS_MEM_USED and JVM_NAT_MEM columns of the PERF_X_01~31 table. The memory utilization trend on a certain date can be viewed on a line chart.

Figure 9-4:Today's System Memory Utilization



Notice: Like the system CPU utilization, the system memory utilization and the Java process memory utilization require the correct installation of the native module.

9.2.2. Total Java Heap Memory and Java Heap Memory Utilization

The total Java heap memory is obtained from the following Java code.

```
Runtime runtime = Runtime.getRuntime();
long totalMemory = runtime.totalMemory();
```

And, the Java heap memory utilization is can be obtained from the following Java code

```
Runtime runtime = Runtime.getRuntime();
long usedMemory = runtime.totalMemory() - runtime.freeMemory();
```

The value to be obtained from the above code is expressed in bytes, but JENNIFER converts it into Mbytes.

The total Java heap memory and the Java heap memory utilization are general performance data collected by the JENNIFER agent. The thirty-second averages can be viewed on a runtime line chart.

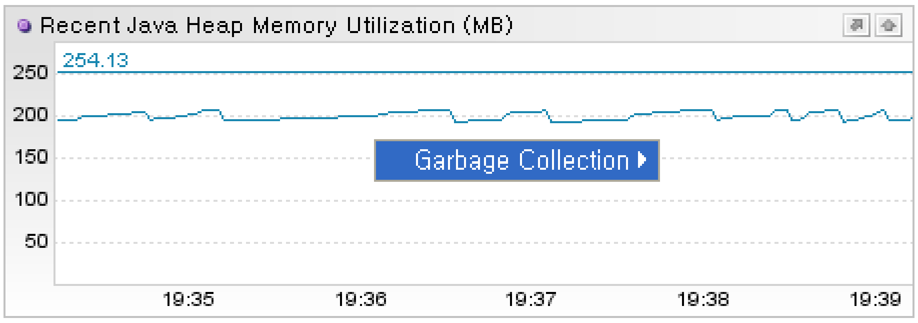
In addition, the five-minute average of the total Java heap memory and the Java heap memory utilization is saved in the HEAP_TOTAL and HEAP_USED columns of the PERF_X_01~31 table. The Java heap memory utilization trend on a certain date can be viewed on a line chart.

Notice: The Java heap memory utilization is the ratio of the Java heap memory being used to the total Java heap memory. It is expressed as a %. However, since the total Java heap memory is unlikely to change, the Java heap memory utilization and the Java heap memory utilization rate have similar shapes on a line chart and a runtime line chart.

On a runtime line chart illustrating the total Java heap memory, the Java heap memory utilization and the Java heap memory utilization rate, you can perform garbage collection on a specific JENNIFER agent.

- Right-click on a runtime line chart. A contextual menu will appear.
- From the contextual menu, select **[Garbage collection]** and select a JENNIFER agent from which you want to collect garbages. Garbage collection will now be performed.

Figure 9-5:Java Heap Memory Garbage Collection



Notice: The contextual menu will appear only if the user’s group has a gc privilege.

9.2.3. Memory Monitoring and Issuing Alerts

If the Java heap memory utilization rate exceeds the threshold for a certain period of time, the WARNING_JVM_HEAP_MEM_HIGH alert will be issued.

Please refer to Alert and Exception Monitoring for more details.

9.2.3.1. Memory Collection Monitoring

Java heap memory leakage in the Java application is the main cause of performance degradation. To solve this problem, JENNIFER provides memory-collection monitoring tool.

Understanding Memory-Collection Monitoring

Memory-collection monitoring is a function used to track the Java collection object that exceeds the threshold out of all the Java collection objects that implement the java.util.Collection and java.util.Map interface, and to record the result in stacktrace. The threshold can be set using the lwst_collection_minimum_monitoring_size option of the JENNIFER agent. The default is 3000.

For example, let's assume that the following Java code is executed by a certain transaction.

```
java.util.List books = new java.util.ArrayList();
for (int i = 0; i < 4000; i++) {
    books.add(new Book(i));
}
```

In the above code, 4000 objects are included in the books Java collection object. So, the JENNIFER agent tracks the books Java collection object.

The user can check the list of the tracked objects using the **[Problem Determination | Memory-Collection]** menu.

9.2.4. Activating Memory-Collection Monitoring

By default, memory-collection monitoring is deactivated. To activate this function, you must set the build_collection patch option to true and restart LWST build. For more details, refer to [LWIS build and installation].

One reason for not using memory-collection monitoring is that it can cause severe performance degradation compared to other types of monitoring. For this reason, use it only when you need to resolve the issues related to Java heap memory leakages. After use, set the build_collection patch option to false and restart LWST build.

9.2.5. Using Memory-Collection Monitoring

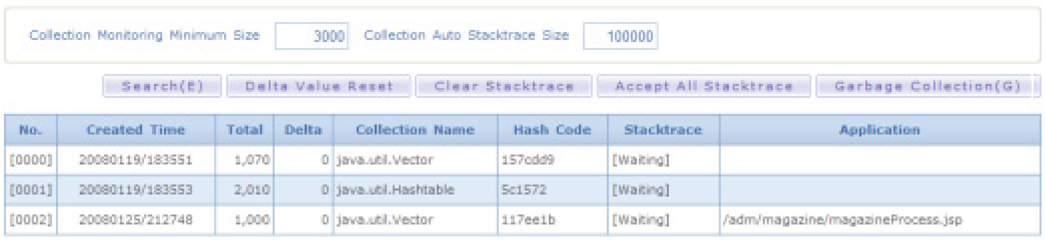
All of the memory-collection monitoring functions can be accessed via the **[Problem Determination | Memory-Collection]** menu. From this menu, you can check the change in the number of objects in the Java collection object and extract the stracktrace of the transaction that uses the Java collection object.

First, if you select a certain JENNIFER agent from the JENNIFER agent selection area, a list of Java collection objects with more objects than the threshold set in the lwst_collection_minimum_monitoring_size option of the JENNIFER agent will appear. The default is 3000.

```
lwst_collection_minimum_monitoring_size = 3000
```

Without modifying the lwst_collection_minimum_monitoring_size option, you can temporarily change the threshold using the [Min Collection Monitoring Size] menu. After entering an arbitrary value, click the [Search] button.

Figure 9-6:Memory-Collection Monitoring Screen



Collection Monitoring Minimum Size: 3000
Collection Auto Stacktrace Size: 100000

No.	Created Time	Total	Delta	Collection Name	Hash Code	Stacktrace	Application
[0000]	20080119/183551	1,070	0	java.util.Vector	157cdd9	[Waiting]	
[0001]	20080119/183553	2,010	0	java.util.Hashtable	5c1572	[Waiting]	
[0002]	20080125/212748	1,000	0	java.util.Vector	117ee1b	[Waiting]	/adm/magazine/magazineProcess.jsp

The following table describes the memory-collection monitoring items.

Table 9-2: Memory-Collection Monitoring Items

Item	Description
Number	Serial number
Created time	The time when the Java collection object is created
All	The number of objects in the Java collection object. It is the value returned by the java.util.Collection or java.util.Map interface.
Delta	The change in the number of objects in the Java collection object
Collection name	Name of the Java collection
Hash code	Hash code for the Java collection object
Stacktrace	Stacktrace extracted from the transaction that uses the Java collection object
Application	Application name of the transaction that uses the Java collection object

To check the change in the number of objects in the Java collection object, follow the steps described below.

- Click the **[Delta Value Reset]** button.
- Click the **[Search]** button repeatedly. While doing this, check the change in the number of objects in the Java collection object in [Delta].

You can extract a stacktrace of the transaction that uses the Java collection object. If you want to extract a stacktrace for a single Java collection object, proceed as follows:

- The **[Trace]** link will appear for the stacktrace item of the Java collection object.
- If you want to record a stacktrace, then click on the **[Trace]** link. The **[Trace]** link will then become the **[Waiting]** link.
- If the **[Waiting]** link appears, the agent is waiting to extract a stacktrace. The reason for the wait is that the transaction has to use the Java collection object before a stacktrace can be recorded.

- If the transaction does not use the Java collection object, then no stacktrace will be extracted. For this reason, continue clicking the **[Search]** button until the transaction uses the Java collection object.
- When a stacktrace is extracted, the stacktrace menu will show the **[Delete]** and the **[Trace Again]** links. The stacktrace information will appear in the bottom.
- If you click the **[Delete]** link, the extracted stacktrace will be deleted and the **[Trace]** link will appear.
- If you click on the **[Trace Again]** link, the extracted stacktrace will be deleted and the **[Waiting]** link will appear. In other words, the agent is waiting for a stacktrace to be extracted from a new transaction.

Notice: Once a stacktrace is extracted from the transaction, no stacktrace will be extracted even if another transaction uses the Java collection object. Therefore, if you want to record a new stacktrace, click he **[Trace Again]** link.

If you want to extract stacktraces from all Java collection objects, proceed as follows:

- If you want to extract stacktraces from all Java collection objects, click the **[Accept All Stacktraces]** button at the top, and keep pressing the **[Search]** button until a stacktrace is extracted.
- If you want to delete all stacktraces, click the **[Clear Stacktrace]** button..

Notice: The extracted stacktrace will be located in the Java heap memory of the Java application in which the JENNIFER agent is installed. For this reason, you are recommended to delete all stacktraces after completing the analysis.

If the number of objects in the Java collection object exceeds the threshold, a stacktrace will automatically be extracted. The threshold can be set using the lwst_collection_auto_stacktrace_size option of the JENNIFER agent. The default is 100000.

```
lwst_collection_auto_stacktrace_size = 100000
```

Without modifying the lwst_collection_auto_stacktrace_size option, you can temporarily change the threshold in the [Collection Auto Stacktrace Size] menu. After entering an arbitrary value in the [Collection Auto Stacktrace Size] menu, click the **[Search]** button.

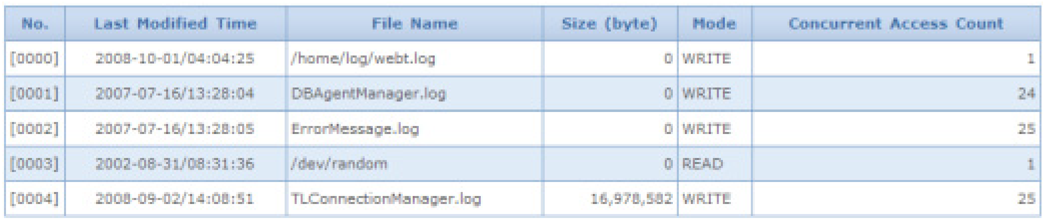
9.3. File/Socket Monitoring

The Java application can cause degradation in the performance of the I/O. To solve this problem, JENNIFER provides file/socket monitoring as a tool.

9.3.1. File Monitoring

In the **[Trouble Determination | File/Socket]** menu, you can check the IO status of the active Java application files. If you want to monitor the file IO status, set the build_file patch option to true and perform LWST build. By default, it is set to true. For more details, refer to [LWST build and installation].

Figure 9-7:File IO List



No.	Last Modified Time	File Name	Size (byte)	Mode	Concurrent Access Count
[0000]	2008-10-01/04:04:25	/home/log/webt.log	0	WRITE	1
[0001]	2007-07-16/13:28:04	DBAgentManager.log	0	WRITE	24
[0002]	2007-07-16/13:28:05	ErrorMessage.log	0	WRITE	25
[0003]	2002-08-31/08:31:36	/dev/random	0	READ	1
[0004]	2008-09-02/14:08:51	TLConnectionManager.log	16,978,582	WRITE	25

The following table describes the file IO items.

Table 9-3: File IO Items

Column name	Description
Number	Serial number
Last modified time	The most recent time when the file was modified
Filename	Absolute directory for the file
Size	File size in bytes
Mode	A code to represent the working status. Set to Read or Write.
Concurrent access counts	The number of simultaneous accesses to the file

Using the file IO monitoring methods, you can check whether or not the Java application is executing unnecessary file IO.

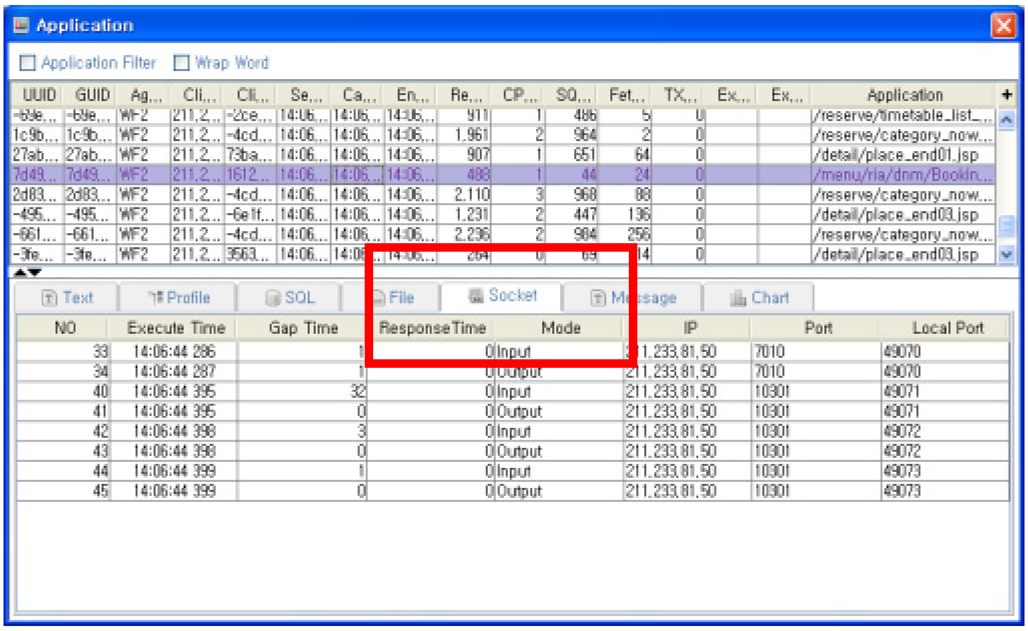
Notice: File IO monitoring only detects files accessed with the java.io.FileInputStream or java.io.FileOutputStream class. Therefore, if you open a C program file using the JNI, the file will not appear in the IO list.

9.3.2. Socket Monitoring

Socket monitoring shows the status of socket IO through the java.net.Socket class of the Java application. If you want to monitor the status of socket IO, set the build_socket patch option to true and perform LWST build. By default, it is set to true. For more details, refer to [LWST build and installation].

Basically, the socket IO works done by a certain transaction are displayed in the X-view profile data.

Figure 9-8:X-View Profile - Socket



Application

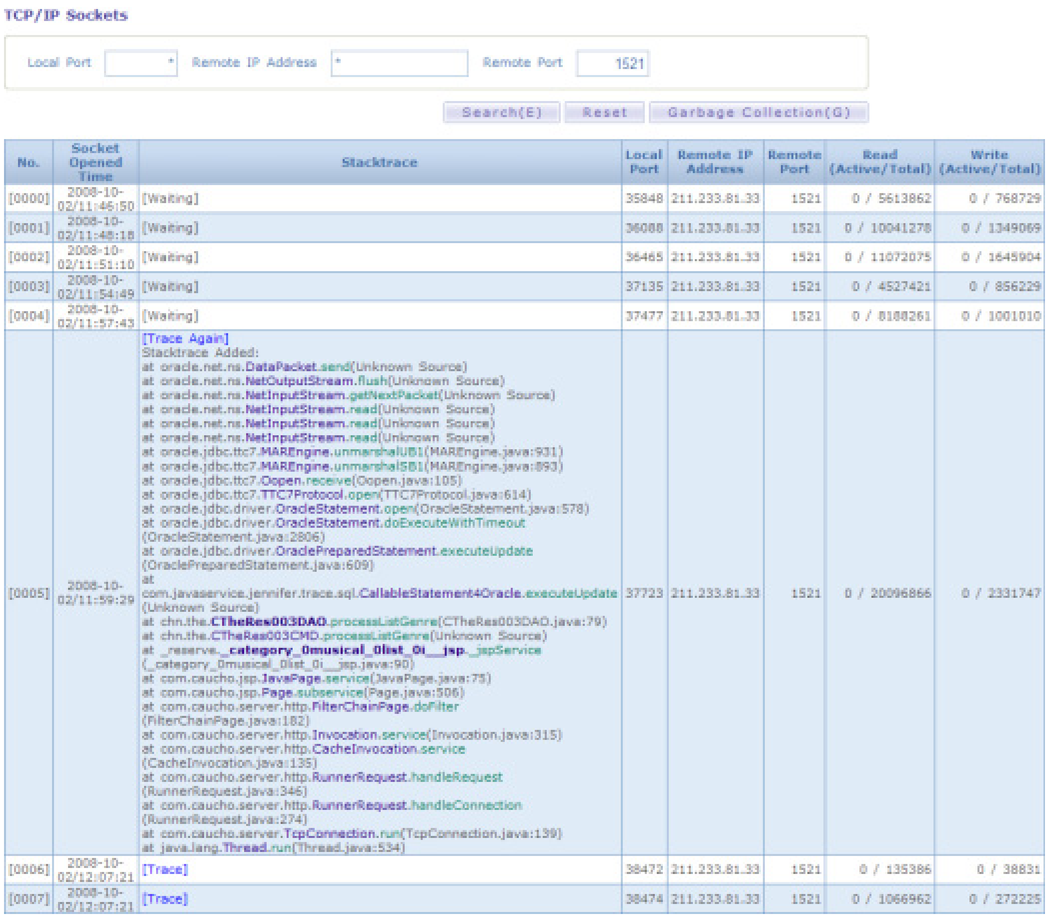
UUID	GUID	Ag...	Cli...	Cli...	Se...	Ca...	En...	Re...	CP...	SQ...	Fet...	TX...	Ex...	Ex...	Application
-69e...	-69e...	WF2	211.2...	-2ce...	14:06...	14:06...	14:06...	911	1	486	5	0			/reserve/timetable_list_...
1c9b...	1c9b...	WF2	211.2...	-4cd...	14:06...	14:06...	14:06...	1,961	2	964	2	0			/reserve/category_now...
27ab...	27ab...	WF2	211.2...	73ba...	14:06...	14:06...	14:06...	907	1	651	64	0			/detail/place_end01.jsp
7d49...	7d49...	WF2	211.2...	1612...	14:06...	14:06...	14:06...	488	1	44	24	0			/menu/ria/dnm/Bookin...
2d83...	2d83...	WF2	211.2...	-4cd...	14:06...	14:06...	14:06...	2,110	3	968	88	0			/reserve/category_now...
-495...	-495...	WF2	211.2...	-6e1f...	14:06...	14:06...	14:06...	1,231	2	447	136	0			/detail/place_end03.jsp
-661...	-661...	WF2	211.2...	-4cd...	14:06...	14:06...	14:06...	2,236	2	904	256	0			/reserve/category_now...
-3fe...	-3fe...	WF2	211.2...	3563...	14:06...	14:06	14:06...	264	0	69	14	0			/detail/place_end03.jsp

NO	Execute Time	Gap Time	Response Time	Mode	IP	Port	Local Port
33	14:06:44 286	1	0	Input	211.233.81.50	7010	49070
34	14:06:44 287	1	0	Output	211.233.81.50	7010	49070
40	14:06:44 395	32	0	Input	211.233.81.50	10301	49071
41	14:06:44 395	0	0	Output	211.233.81.50	10301	49071
42	14:06:44 398	3	0	Input	211.233.81.50	10301	49072
43	14:06:44 398	0	0	Output	211.233.81.50	10301	49072
44	14:06:44 399	1	0	Input	211.233.81.50	10301	49073
45	14:06:44 399	0	0	Output	211.233.81.50	10301	49073

If you set the socket_simple_trace option of the JENNIFER agent to false, you can monitor the status of socket IO of the active Java application in the **[Trouble Determination | File/Socket]** menu. By default, it is set to true.

```
socket_simple_trace = false
```

Figure 9-9:Socket IO List


TCP/IP Sockets

Local Port [*] Remote IP Address [*] Remote Port [1521]

No.	Socket Opened Time	Stacktrace	Local Port	Remote IP Address	Remote Port	Read (Active/Total)	Write (Active/Total)
[0000]	2008-10-02/11:46:50	[Waiting]	35848	211.233.81.33	1521	0 / 5613862	0 / 768729
[0001]	2008-10-02/11:48:18	[Waiting]	36088	211.233.81.33	1521	0 / 10041278	0 / 1349069
[0002]	2008-10-02/11:51:10	[Waiting]	36465	211.233.81.33	1521	0 / 11072075	0 / 1645904
[0003]	2008-10-02/11:54:49	[Waiting]	37135	211.233.81.33	1521	0 / 4527421	0 / 856229
[0004]	2008-10-02/11:57:43	[Waiting]	37477	211.233.81.33	1521	0 / 8188261	0 / 1001010
[0005]	2008-10-02/11:59:29	[Trace Again] Stacktrace Added: at oracle.net.ns.DataPacket.send(Unknown Source) at oracle.net.ns.NetOutputStream.flush(Unknown Source) at oracle.net.ns.NetInputStream.getNextPacket(Unknown Source) at oracle.net.ns.NetInputStream.read(Unknown Source) at oracle.net.ns.NetInputStream.read(Unknown Source) at oracle.net.ns.NetInputStream.read(Unknown Source) at oracle.jdbc.ttc7.MAREngine.unmarshalUB1(MAREngine.java:931) at oracle.jdbc.ttc7.MAREngine.unmarshalSB1(MAREngine.java:893) at oracle.jdbc.ttc7.Oopen.receive(Oopen.java:105) at oracle.jdbc.ttc7.TTC7Protocol.open(TTC7Protocol.java:614) at oracle.jdbc.driver.OracleStatement.open(OracleStatement.java:578) at oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout (OracleStatement.java:2806) at oracle.jdbc.driver.OraclePreparedStatement.executeUpdate (OraclePreparedStatement.java:609) at com.javaservice.jennifer.trace.sql.CallableStatement4Oracle.executeUpdate (Unknown Source) at chn.the.CTheRes003DAO.processListGenre(CTheRes003DAO.java:79) at chn.the.CTheRes003CMD.processListGenre(Unknown Source) at _reserve._category_0musical_0list_0i__jsp._jspService (_category_0musical_0list_0i__jsp.java:90) at com.caucho.jsp.JavaPage.service(JavaPage.java:75) at com.caucho.jsp.Page.subservice(Page.java:506) at com.caucho.server.http.FilterChainPage.doFilter (FilterChainPage.java:182) at com.caucho.server.http.Invocation.service(Invocation.java:315) at com.caucho.server.http.CacheInvocation.service (CacheInvocation.java:135) at com.caucho.server.http.RunnerRequest.handleRequest (RunnerRequest.java:346) at com.caucho.server.http.RunnerRequest.handleConnection (RunnerRequest.java:274) at com.caucho.server.TcpConnection.run(TcpConnection.java:139) at java.lang.Thread.run(Thread.java:534)	37723	211.233.81.33	1521	0 / 20096866	0 / 2331747
[0006]	2008-10-02/12:07:21	[Trace]	38472	211.233.81.33	1521	0 / 135386	0 / 38831
[0007]	2008-10-02/12:07:21	[Trace]	38474	211.233.81.33	1521	0 / 1066962	0 / 272225

The following table describes the Socket IO items.

Table 9-4: Socket IO Items

Items	Description
Number	Serial number
Socket opened time	The time when the socket is opened
Stacktrace	Stacktrace extracted from the transaction that uses the socket
Local port	Socket's local port number
Remote IP address	Socket's remote server IP address
Remote port	Socket's remote server port number
Read(Active/Total)	Displays the size of the data read from the socket. It is expressed in bytes. Active means the data size that is read currently, while Total means the size of the accumulated data read from the socket.

Items	Description
Write(Active/Total)	Displays the size of the data read from the socket. It is expressed in bytes. Active means the data size that is read currently, while Total means the size of the accumulated data read from the socket.

If there is too much socket IO information displayed in the socket IO list, you can sort the information by [Local Port], [Remote IP address] and [Remote Port]. If you click the **[Reset]** button beside **[Search]**, the searching condition is initialized.

You can extract a stacktrace of the transaction that uses the socket. If you want to extract a stacktrace for a single socket, proceed as follows:

- The **[Trace]** link will appear for the stacktrace item of the Socket.
- If you want to record a stacktrace, then click on the **[Trace]** link. The **[Trace]** link will then become the **[Waiting]** link.
- If the **[Waiting]** link appears, the agent is waiting to extract a stacktrace. The reason for the wait is that the transaction has to use the socket before a stacktrace can be recorded.
- If the transaction does not use the socket, then no stacktrace will be extracted. For this reason, continue clicking the **[Search]** button until the transaction uses the socket.
- When a stacktrace is extracted, the stacktrace menu will show the **[Delete]** and the **[Trace Again]** links. The stacktrace information will appear in the bottom.
- If you click the **[Delete]** link, the extracted stacktrace will be deleted and the **[Trace]** link will appear.
- If you click on the **[Trace Again]** link, the extracted stacktrace will be deleted and the **[Waiting]** link will appear. In other words, the agent is waiting for a stacktrace to be extracted from a new transaction

Notice: Once a stacktrace is extracted from the transaction, no stacktrace will be extracted even if another transaction uses the socket. Therefore, if you want to record a new stacktrace, click he **[Trace Again]** link.

Socket monitoring can be used as a means of acquiring the necessary configuration information for JDBC connecting tracking of the JENNIFER agent. For JDBC monitoring, find the class that provides the java.sql.Connection object. However, if it is not possible to find it, find the socket for database connections by using the remote IP address and port number of the database server, and extract a stacktrace from the transaction that uses the socket. After doing this, you can determine which class returns the java.sql.Connection object.

A stacktrace is automatically extracted from the remote port that has been set using the lwst_trace_remote_port option of the JENNIFER agent. For example, if you want to automatically record a stacktrace from the Java application socket that communicates with port 2300 of an external EAI solution, set it as follows:

```
lwst_trace_remote_port = 2300
```

If a certain remote port number is set using this option, a stacktrace is automatically extracted from the socket that uses the remote port, even when the socket_simple_trace option is set to true.

Even if you change the lwst_trace_remote_port option of the JENNIFER agent, you do not need to restart the Java application. If a socket object that uses the remote port number set by this option is already created and pulled, you cannott extract a Stacktrace from the socket object. In this case, restart the Java application.

9.4. Live Object Monitoring

Live object monitoring is a function used to obtain the number of objects in the class.

9.4.1. Live Object Monitoring Configuration

To prevent an unnecessary degradation in performance, do not attempt to obtain the number of all objects. Instead, perform live object monitoring on the classes set with the liveobject option of the JENNIFER agent. If you start the option set with the corrected liveobject, you have to restart the Java applications installed 로 시작the JENNIFER agent.

For example, to obtain the number of objects in the java.util.ArrayList class, set it as follows:

```
liveobject_class = java.util.ArrayList
```

Multiple classes are separated by semicolons [;]. Multiple entities in the liveobject option are also separated by semicolons. To obtain the number of objects in the java.util.HashMap class, set it as follows:

```
liveobject_class = java.util.ArrayList;java.util.HashMap
```

To obtain the number of all classes inheriting from a certain class, use the liveobject_super option of the JENNIFER agent. For example, if you want to obtain the number of child classes of the java.lang.Thread class, set it as follows:

```
liveobject_super = java.lang.Thread
```

However, in this case, only the number of objects in the classes that are directly inherited is monitored. For expample, if class A inherits from the java.lang.Thread class and class B inherits from class A, the number of objects in class A is monitored, but the number of objects in class B is not.

To obtain the number of objects in all classes that implement a certain interface, use the liveobject_interface option of the JENNIFER agent. For example, if you want to obtain the number of objects in the classes that implement the java.lang.Runnable and java.sql.Connection interface, set it as follows:

```
liveobject_interface = java.lang.Runnable;java.sql.Connection
```

However, in this case, only the number of objects in the classes that directly implement the interface will be monitored. For example, if class A inherits from the java.lang.Runnable interface and class B inherits from class A, the number of objects in class A is monitored but the number of objects in class B is not.

You can monitor the number of objects in classes with certain names. For example, to obtain the number of objects in all classes in the java.util package, set it as follows:

```
liveobject_prefix = java.util
```

And, you can monitor the number of objects in classes with certain names. For example, to obtain the number of objects in all classes with ‘Factory’ postfix, set it as follows.

```
liveobject_postfix = Factory
```

9.5. HTTP Session Monitoring

An HTTP session is used to maintain the user status in the web application. An HTTP session on the Java application implements the javax.servlet.http.HttpSession interface, and it varies depending on the Java application server.

9.5.1. Dummy HTTP Session Tracking

If an HTTP session object is made when the JSP set to true is executed or the page attribute session is not created, then the JENNIFER agent will issue a WARNING_DUMMY_HTTPSESSION_CREATED exception. In most environments, this exception is not generated because it is not suitable. To generate this exception, set the enable_dummy_httpsession_trace option of the JENNIFER agent to true. By default, it is set to false.

```
enable_dummy_httpsession_trace = true
```

9.5.2. HTTP Session Dump

In the **[Problem Determination | HTTP Session]** menu, you can check the active HTTP session objects of the Java application server. This is called an HTTP session dump. If you want to record an HTTP session dump, additional configuration is required for each Java application server, because each Java application server requires different implementation of the javax.servlet.http.HttpSession interface.

For this reason, an HTTP session dump is not available for every type of Java application server. The following Java application server support an HTTP session dump.

- Apache tomcat 4.x, 5.x, 6.x
- BEA web logic 8.x, 9.x, 10.x
- IBM websphere 5.x, 6.0, 6.1, 7.0

The HTTP session dump method for each Java application server is as follows. After configuration is completed, restart the Java application server.

First, for apache tomcat 4.x, 5.x, set it as follows:

- Copy the JENNIFER_HOME/agent/jennifer_session.jar file into the TOMCAT_HOME/server/lib directory.
- Set the session_class option of the JENNIFER agent to tomcat.

```
session_class = tomcat
```

- If the configuration is successful, the following message will appear in the JENNIFER agent log file.

```
Catalina session-trace ok
```

For apache tomcat 6.x, set it as follows:

- Copy the JENNIFER_HOME/agent/jennifer_session.jar file into the TOMCAT_HOME/lib directory.
- Set the session_class option of the JENNIFER agent to tomcat.

```
session_class = tomcat
```

- If the configuration is successful, the following message will appear in the JENNIFER agent log file..

```
Catalina session-trace ok
```

For oracle weblogic 8.x, 9,x, 10,x, set it as follows:

- Put the JENNIFER_HOME/agent/jennifer_session.jar file behind the jennifer.jar file.
- Set the session_class option of the JENNIFER agent to weblogic.

```
session_class = weblogic
```

- If the configuration is successful, the following message will appear in the JENNIFER agent log file.

```
Weblogic session-trace ok
```

For IBM websphere 5.x, 6.0 set it as follows:

- Copy the JENNIFER_HOME/agent/jennifer_session.jar file into the WEBSPHERE_HOME/lib directory.
- Set the session_class option of the JENNIFER agent to websphere.

```
session_class = websphere
```

- If the configuration is successful, the following message will appear in the JENNIFER agent log file.

```
WebSphere session-trace ok
```

For IBM websphere 6.1 set it as follows:.

Warning: If you want to execute HTTP sesstion dump in the IBM WebSphere 6.1, you have to change the library. This is not recommended except in the special case.

- Backup the WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer_2.0.0.jar file into the temp directory.
- Copy the classes in the JENNIFER_HOME/agent/jennifer_session.jar file into the WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer_2.0.0.jar file.
- Set the session_class option of the JENNIFER agent to websphere.

```
session_class = websphere
```

If the configuration is successful, the following message will appear in the JENNIFER agent log file.

```
WebSphere session-trace ok
```

Finally, for IBM websphere 7.x, set it as follows:

Warning: If you want to execute HTTP sesstion dump in the IBM WebSphere 7.x, you have to change the library. This is not recommended except fexcept in the special case.

- Backup the WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer.jar file in the temp directory.
- Let the classes in the JENNIFER_HOME/agent/jennifer_session.jar file include in the WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer.jar file.
- Set the session_class option of the JENNIFER agent to websphere7.

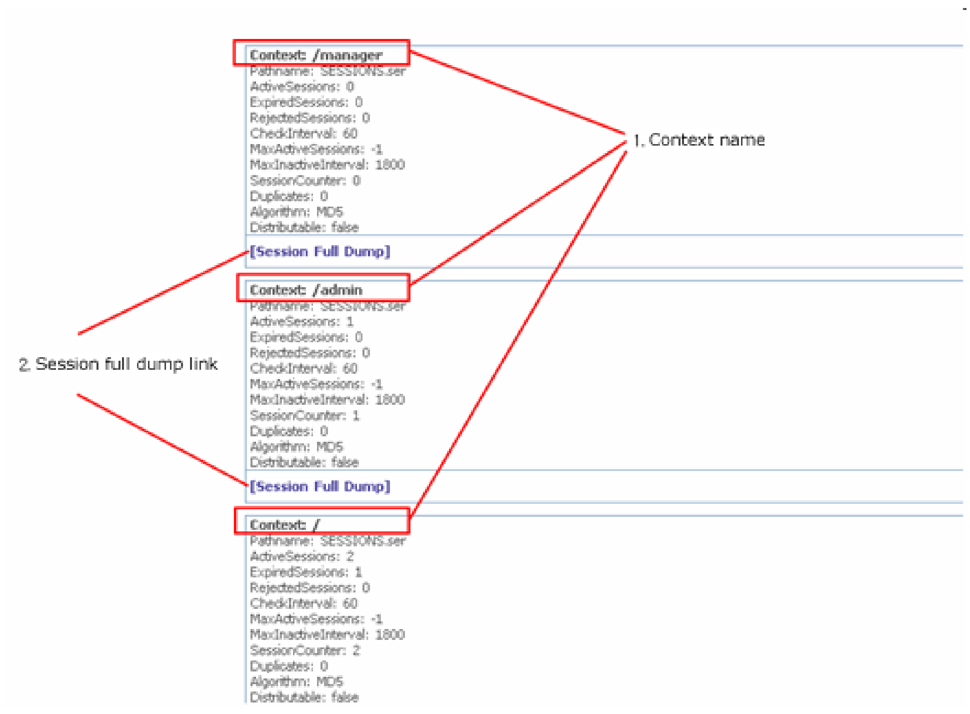
```
session_class = websphere7
```

If the configuration is successful, the following message will appear in the JENNIFER agent log file.

```
WebSphere session-trace ok
```

Once you set the HTTP session dump, you can check the status of HTTP session objects using the **[Problem Determination | HTTP Session]** menu.

Figure 9-10:HTTP Session List



```
Context: /manager
Pathname: SESSIONS.ser
ActiveSessions: 0
ExpiredSessions: 0
RejectedSessions: 0
CheckInterval: 60
MaxActiveSessions: -1
MaxInactiveInterval: 1800
SessionCounter: 0
Duplicates: 0
Algorithm: MD5
Distributable: false
[Session Full Dump]
Context: /admin
Pathname: SESSIONS.ser
ActiveSessions: 1
ExpiredSessions: 0
RejectedSessions: 0
CheckInterval: 60
MaxActiveSessions: -1
MaxInactiveInterval: 1800
SessionCounter: 1
Duplicates: 0
Algorithm: MD5
Distributable: false
[Session Full Dump]
Context: /
Pathname: SESSIONS.ser
ActiveSessions: 2
ExpiredSessions: 1
RejectedSessions: 0
CheckInterval: 60
MaxActiveSessions: -1
MaxInactiveInterval: 1800
SessionCounter: 2
Duplicates: 0
Algorithm: MD5
Distributable: false
```

- Context name - JENNIFER classifies HTTP session related statistical information according to the context of the web application. For this reason, the context name can help you to understand the status of HTTP sessions in the Java application server.
- Session Full Dump Link - If you click on the line, you can check the basic information of all HTTP session objects related to a certain application.

The following table describes the main information of the HTTP session dump for apache tomcat as an example. The HTTP session dump information varies according to the Java application server.

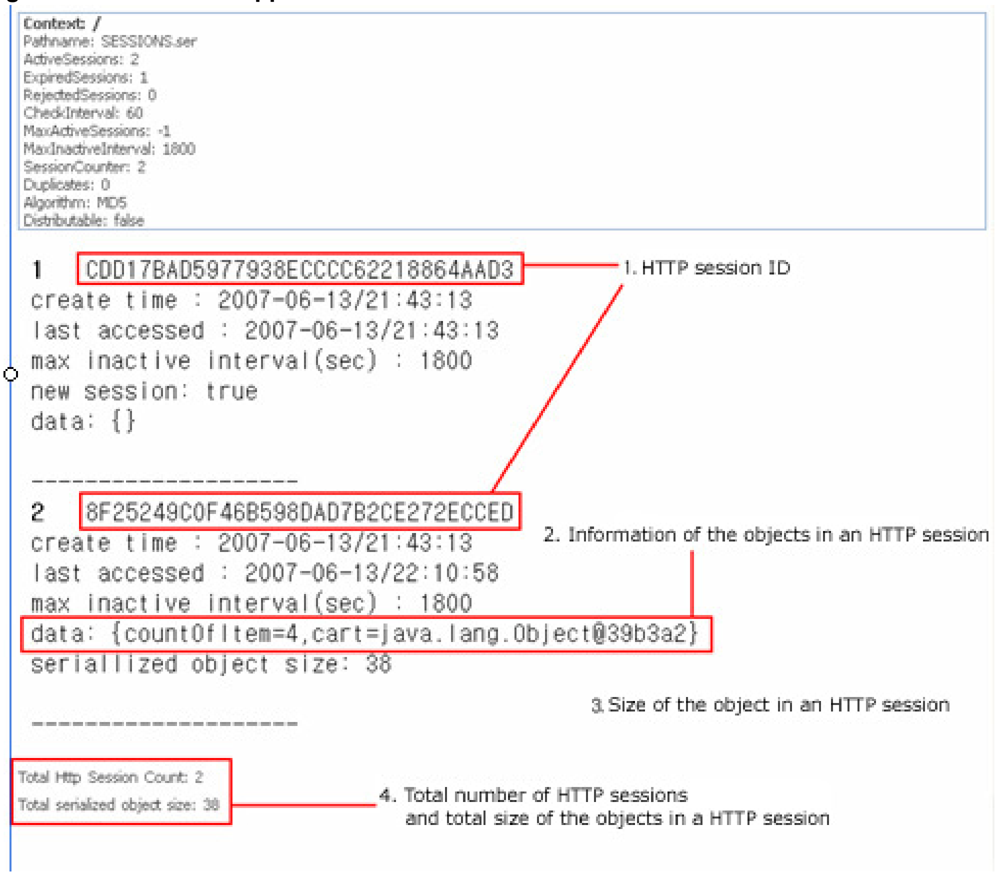
Table 9-5: HTTP Session Information (Tomcat)

Item	Description
Pathname	The active session objects are saved in a file when the Java application server is stopped, and are loaded from the file when the Java application server is restarted. Pathname refers to the path for this file. For relative paths, the temporary directory set by the javax.servlet.context.tempdir attribute is used.
ActiveSessions	Number of active HTTP session objects
ExpiredSessions	Number of expired HTTP session objects

Table 9-5: HTTP Session Information (Tomcat)

Item	Description
RejectedSessions	Number of rejected HTTP sessions because the maximum number of active HTTP session objects was exceeded.
CheckInterval	Period for checking whether the HTTP session object is expired. Expressed in seconds.
MaxActiveSessions	Maximum number of active HTTP session objects that can be created by the Java application server. -1 means that it is unlimited.
MaxInactiveInterval	Maximum time period for clearing the HTTP session objects when a user does not request it. Expressed in seconds. A negative number means that the HTTP session object is not cleared.
SessionCounter	Number of HTTP session objects created by the Java application server
Duplicates	Number of duplicate HTTP session IDs. A positive number means that it is duplicated.
Algorithm	An algorithm to create the HTTP session ID. It is supported by the java.security.MessageDigest class.
Distributable	Determines whether the HTTP session object is copied in a distributed environment. If set to true, all objects added in the HTTP session must be implemented by the java.io.Serializable interface.

Figure 9-11:Individual Application's HTTP Session Information



```
Context: /
Pathname: SESSIONS.ser
ActiveSessions: 2
ExpiredSessions: 1
RejectedSessions: 0
CheckInterval: 60
MaxActiveSessions: -1
MaxInactiveInterval: 1800
SessionCounter: 2
Duplicates: 0
Algorithm: MD5
Distributable: false

1   CDD17BAD5977938ECCCC62218864AAD3
create time : 2007-06-13/21:43:13
last accessed : 2007-06-13/21:43:13
max inactive interval(sec) : 1800
new session: true
data: {}

--------------------
2   8F25249C0F46B598DAD7B2CE272ECCED
create time : 2007-06-13/21:43:13
last accessed : 2007-06-13/22:10:58
max inactive interval(sec) : 1800
data: {countOfItem=4,cart=java.lang.Object@39b3a2}
seriallized object size: 38

--------------------

Total Http Session Count: 2
Total serialized object size: 38
```

1. HTTP session ID
2. Information of the objects in an HTTP session
3. Size of the object in an HTTP session
4. Total number of HTTP sessions and total size of the objects in a HTTP session

- HTTP session ID - The value returned by the getid method of the javax.servlet.http.HttpSession object.
- Information of the objects in an HTTP session - Shows the information of every object in an HTTP session. It is comprised of the name used to save the object in the HTTP session and the value returned by the toString method of the object.
- Size of the object in an HTTP session - Shows the total size of the objects included in an HTTP session. However, any object or variable assigned to the field declared as transient will be excluded.
- Total number of HTTP sessions and total size of the objects in a HTTP session - Shows the number of HTTP sessions in a specific application, and the total size of all the objects in a HTTP session.

Create time, last accessed and maximum inactive interval are the values returned by the getCreationTime, getLastAccessedTime and getMaxInactiveInterval methods of the HttpSession object, respectively.

9.6. External Transaction Monitoring

An external transaction implies that the Java application communicates with other applications. The status of external transaction processing and the response time greatly affects the the performance of the entire Java application.

Therefore, by monitoring external transactions, JENNIFER can collect the X-view profile data, including external transaction processing and external transaction processing statistic data.

An external transaction implies an interconnection with external applications such as the database, the TP(Transaction Processing) monitors, and the LDAP (Lightweight Directory Access Protocol) servers. However, due to its importance and special characteristics, the database is handled as additional JDBC monitoring. Therefore, the purpose of external transaction monitoring for JENNIFER is to monitor interconnections with every external application, excluding the database.

9.6.1. Configuration of the External Transaction Start Point

The purpose of external transaction monitoring is to monitor an internal Java application module (a certain method of the class) for interconnections with other applications. For example, if the example.MailManager class interconnects with the mail server and the sendMail method sends files, external transaction monitoring icollects the hits and response time from the sendMail method of the example.MailManager class.

To achieve this, first determine which class of which method is the external transaction start point. This is called external transaction start point configuration.

The following types of interconnection with external applications are automatically recognized as external transaction start points, without the need for additional configuration:

- BEA JOLT 2.0, 2.1, 2.5, 3.0
- BEA WTC (Weblogic Tuxedo Connector)
- 티Max soft WebT 3.x, 5.x

However, the IBM CICS (Customer Information Control System) is not automatically recognized as an external transaction start point. But when using CTG 5.x, if the JENNIFER_HOME/agent/3rdparty/ctg51_jennifer40.jar file is put ahead of the JAR file for the CTG (CICS Transaction Gateway), it is automatically recognized as an external transaction start point.

Notice: For versions other than CTG 5.x, use the tx_client option to set the external transaction start point.

The rest external transactions except above examples are set as external transaction start points using the tx_client option of the JENNIFER agent. If you adjust the modified transactions, you have to restart the Java application installed the JENNIFER agent.

The following is an example that the the example.MailManager class interconnects with the mail server. If you want to set it as an external transaction start point, use the tx_client_class option of the JENNIFER agent.

```
tx_client_class = example.MailManager
```

Next, let us assume that the example.LdapManager class interconnects with the LDAP server. If you want to set this class as an external transaction start point as well, add it to the tx_client_class option, using a semicolon as a separator. In addition to the tx_client_class option, if you want to enter multiple entities in the options related to external transaction start points, use a semicolon [;] as a separator.

```
tx_client_class = example.MailManager;example.LdapManager
```

An external transaction start point refers to a certain method of the class. Therefore, according to the above configuration, all the methods of the example.MailManager class are recognized as external transaction start points.

Thus, if you want to set specific methods as external transaction start points, use the tx_client_target_method option of the JENNIFER agent.

```
tx_client_target_method = sendMail
```

If the classes set as external transaction start points have the same methods, and you want to set specific methods of the classes as external transaction start points, set it as follows:

```
tx_client_target_method = example.MailManager.sendMail
```

On the other hand, if you want to exclude specific methods from the external transaction start points, set the tx_client_ignore_method option of the JENNIFER agent.

```
tx_client_ignore_method = example.MailManager.someMethod
```

Using accessors of the method, you can set the external transaction start points. For example, if you want to set the public accessor and the method without an accessor as external transaction start points, set it as follows:

```
tx_client_access_method = public;none
```

The following parameters can be set in the tx_client_access_method option.

- public - public accessor
- protected - protected accessor
- private - private accessor
- none - no accessors

If all classes inheriting from a specific class are the external transaction start points, use the tx_client_super option of the JENNIFER agent. For example, if you want to set all the classes inheriting from example.ejb.BaseSessionBean as external transaction start points, set it as follows:

```
tx_client_super = example.ejb.BaseSessionBean
```

However, in this case, only the classes directly inheriting from it become external transaction start points. For example, if class A inherits from the example.ejb.BaseSessionBean class and class B inherits from class A, class A is recognized as an external transaction start point, but class B is not.

If all classes implementing a specific interface are external transaction start points, use the tx_client_interface option of the JENNIFER agent. If you want to set all classes implementing the example.eai.ITransactionManager interfaces as the external transaction start points, set it as follows:

```
tx_client_interface = example.eai.ITransactionManager
```

In this case, only the classes directly implementing example.eai.ITransactionManager become external transaction start points. For example, if class A implements the example.eai.ITransactionManager interface and class B inherits from class A, class is recognized as an external transaction start point, but class B is not.

Using the class name, you can set the external transaction start point. In this case, use the tx_client_prefix, tx_client_postfix and tx_client_ignore_prefix option of the JENNIFER agent. For example, if you want to set all the classes whose names begin with example.eai as external transaction start points, set it as follows:

```
tx_client_prefix = example.eai
```

In addition, if you want to set all the classes whose names end with TpMonitor as external transaction start points, set it as follows:

```
tx_client_postfix = TpMonitor
```

If you want to exclude certain classes from the external transaction start points, use the tx_client_ignore_prefix option of the JENNIFER agent. This option can override all other options.

```
tx_client_ignore_prefix =
```

Using the tx_client option of the JENNIFER agent, you can perform the configuration in various ways. Using the tx_client_target_method option, you can set the method as the external transaction start points.

For example, let us assume that we have the pkg.ClassA and pkg.ClassB class, and that the run method of ClassA and the process method of ClassB are the external transaction start points. If another run method exists in ClassB and it is not related to external transactions, set it as follows:

```
tx_client_class = pkg.ClassA;pkg.ClassB
tx_client_target_method = run;process
tx_client_ignore_method = pkg.ClassB.run
```

Or you can set it as follows:

```
tx_client_class = pkg.ClassA;pkg.ClassB
tx_client_target_method = pkg.ClassA.run;pkg.ClassB.process
```

9.6.2. External Transaction Naming

The external transaction names are stored in the TXNAMES table. In essence, the external transaction name consists of the class name and the method name. When calling the BEA tuxedo, the tuxedo name must be included in the external transaction name. Also, when calling the IBM websphere MQ, the queue name is included in the external transaction name. Otherwise, it is impossible to perform accurate monitoring.

To achieve this, we can set the external transaction names in various ways. This is called external transaction naming. If you want to adjust the modified options related to the external transaction naming, you have to restart the Java application installed the JENNIFER agent.

Essentially, the external transaction name is comprised of the class name and the method name. Using the tx_client_ntype option of the JENNIFER agent, you can set them in various ways. The default is FULL, and it includes other values such as SIMPLE, CLASS and METHOD .

```
tx_client_ntype = FULL
```

For example, if we assume that the process method of the pkg.ClassB class is an external transaction start point, the external transaction name is determined according to the tx_client_ntype option as follows:

- FULL - public void pkg.ClassB.process()
- SIMPLE - ClassB.process
- CLASS - ClassB
- METHOD - process

You can use some portions of the parameter value of the method that is an external transaction start point as the external transaction name. To accomplish this, set the tx_client_using_param option of the JENNIFER agent to true.

```
tx_client_using_param = true
```

If the tx_client_using_param option is set to true, the first java.lang.String type of parameter becomes the external transaction name. If there is no such parameter, the tx_client_ntype option determines the external transaction name.

You can use the parameter or the returned value of a specific method of the class called within the method that is an external transaction start point as the external transaction name.

If you want to use the parameter of a specific method of the class as the external transaction name, set it as follows:

```
lwst_txclient_method_using_param = pkg.ClassC.f1(String)
```

In this case, among the various parameters, the first java.lang.String type of parameter becomes the external transaction name.

If you want to use the returned value of a specific method as the external transaction name, set it as follows:

```
lwst_txclient_method_using_return = pkg.ClassC.f2(String)
```

However, in this case, the returned object type is java.lang.String.

Multiple entries in the lwst_txclient_method_using_param and lwst_txclient_method_using_return options are separated by semicolons [;].

The following priority must apply when multiple options are set.

- lwst_txclient_method_using_param, or lwst_txclient_method_using_return
- tx_client_using_param
- tx_client_ntype

When the lwst_txclient_method_using_param or lwst_txclient_method_using_return options is used, the parameter or returned value of the method that is executed last becomes the external transaction name.

9.6.3. External Transaction Monitoring and Exceptions

If the response time of the external transaction exceeds the threshold, the WARNING_TX_BAD_RESPONSE exception is issued. The threshold can be set in the tx_bad_responsetime option of the JENNIFER agent. The default is 10000, and it is expressed in milliseconds.

```
tx_bad_responsetime = 10000
```

9.7. JDBC Monitoring

Database interconnection using the JDBC greatly affects the Java application performance. For this reason, JENNIFER provides JDBC monitoring for performance analysis and troubleshooting. The following data is collected from JDBC monitoring.

- SQL and parameters executed by the Java application
- SQL response time and fetch counts
- JDBC connections
- Non-returned JDBC objects and exceptions

9.7.1. Basic Configuration for DB Monitoring

For DB Connection monitoring, set the build_jdbc patch option to true and perform LWST build. By default, it is set to true. For more details, refer to [LWST Build and Installation].

In addition, set the enable_jdbc_sql_trace option of the JENNIFER agent to true. By default, it is set to true. If this option is set to false, no DB Connection monitoring will be performed. .

```
enable_jdbc_sql_trace = true
```

In addition, you can deactivate DB Connection monitoring for each individual item. By default, it is set to true. .

```
enable_jdbc_callablestatement_trace = true
enable_jdbc_preparedstatement_trace = true
enable_jdbc_statement_trace = true
enable_jdbc_resultset_trace = true
enable_jdbc_databasemetadata_trace = true
```

If DB Connection monitoring is deactivated for each individual item, no data will be collected. For example, if you set the enable_jdbc_resultset_trace option to false, then the fetch count is not collected. If the enable_jdbc_preparedstatement_trace option is set to false, then the SQL response time executed by the java.sql.PreparedStatement object is not collected.

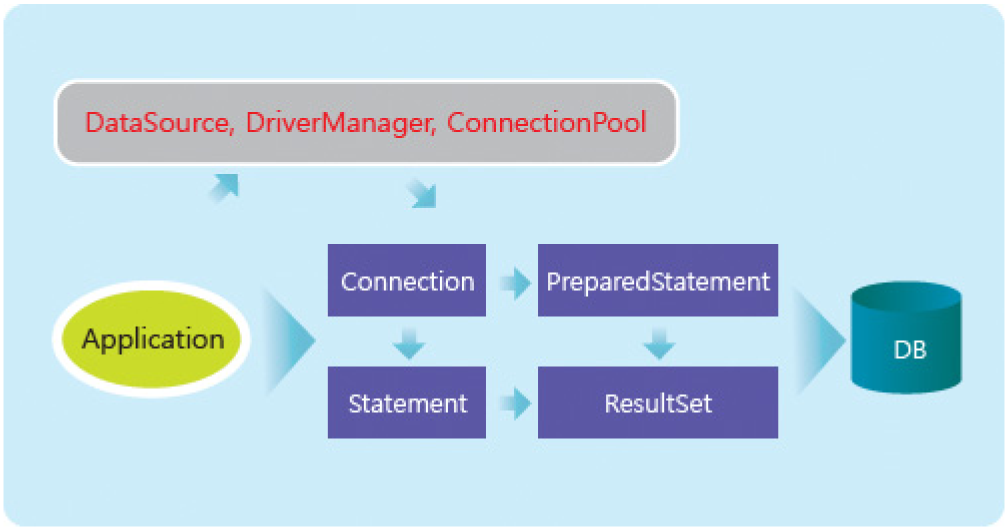
When installing JENNIFER in a Java application that interconnects with the database in a non-standard way, conflicts can occur. In this case, deactivate DB Connection monitoring for each individual item to investigate the cause of such conflict.

9.7.2. DB Connection Tracking Configuration

The Java application interconnects with the database through the JDBC API as follows:

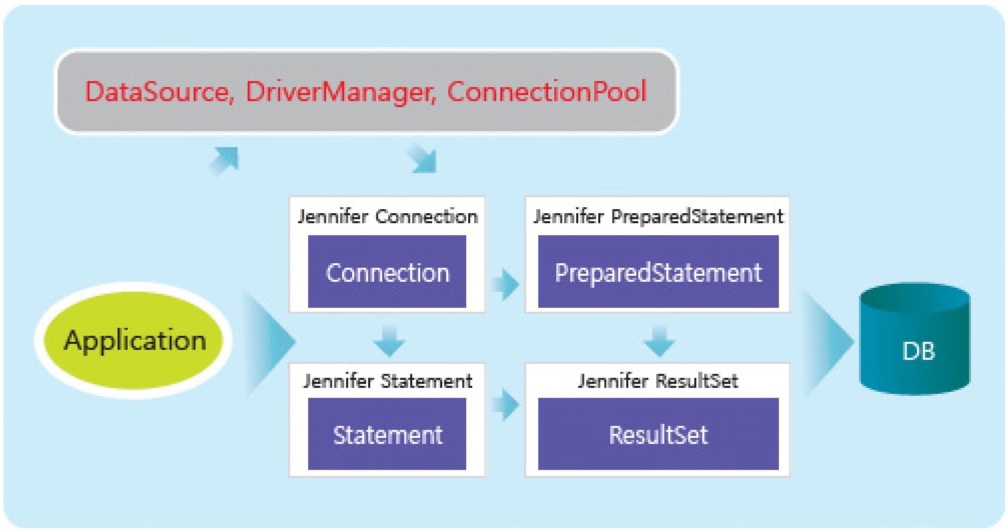
- It obtains the java.sql.Connection object through the java.sql.DriverManager or javax.sql.DataSource class.
- It obtains the java.sql.Statement, java.sql.PreparedStatement and java.sql.ResultSet objects from the java.sql.Connection object and performs database works.

Figure 9-12:JDBC API Architecture



The class wrapping method is used for DB connection monitoring.

Figure 9-13:JDBC Monitoring Architecture



To achieve this, when the Java application obtains the java.sql.Connection object, it must be wrapped to the JenniferConnection object. The java.sql.Statement and java.sql.ResultSet objects are automatically wrapped by the JenniferConnection object.

DB connection tracking configuration refers to wrapping the java.sql.Connection object to the JenniferConnection object. In general, there are three different ways for the application to obtain the java.sql.Connection object. JENNIFER provides suitable methods for these types in DB Connection Tracing chapter..

- Type 1 - Using the JNDI and javax.sql.DataSource objects

- Type 2 - Using the java.sql.DriverManager class
- Type 3 - Using an arbitrary class

However, for the stacktrace that is created while java.sql.Connection object was creating , you are able to save the stacktrace in a JENNIFER agent log file. Set the debug_connection_open option of the JENNIFER agent as true. The default value is false.

```
debug_connection_open = true
```

Notice: Restart is needed.

You can use this stacktrace data for configuring JDBC connection trace. However, JENNIFER builds the stacktrace using printStackTrace method of java.lang.Throwable object as java.sql.Connection is created. This may cause overhead problem. Therefore, use this function just for debugging purpose and set the option generally as false.

9.7.2.1. Type 1 - JNDI & DataSource

The Java application obtains the javax.sql.DataSource object by using the JNDI(Java Naming and Directory Interface) and obtains the java.sql.Connection object from this object.

```
javax.naming.Context jndiContext = new javax.naming.InitialContext();
javax.sql.DataSource ds = (javax.sql.DataSource)
             jndiContext.lookup("java:comp/env/jdbc/AppDS"
);
java.sql.Connection con = ds.getConnection();
```

Type 1 supports JDBC monitoring without any additional configuration. However, set the enable_jdbc_datasource_trace option of the JENNIFER agent to true. By default, it is set to true.

```
enable_jdbc_datasource_trace = true
```

If the javax.sql.DataSource object is not directly obtained from the javax.naming.InitialContext object, but the javax.sql.DataSource object is obtained from the javax.naming.Context object, additional configuration will be required.

```
javax.naming.Context jndiContext = new javax.naming.InitialContext();
javax.naming.Context jdbcContext = (javax.naming.Context)
                jndiContext.lookup("java:comp/env/jdbc"
);
javax.sql.DataSource ds = (javax.sql.DataSource)
                jdbcContext.lookup("AppDS");
java.sql.Connection con = ds.getConnection();
```

In this case, set the enable_wrap_context_jdbc_trace option of the JENNIFER agent to true.

```
enable_wrap_context_jdbc_trace = true
```

Notice: Restart is needed.

However, if the enable_wrap_context_jdbc_trace option is set to true, a java.lang.ClassCastException may occur when using the EJB. In this case, use Type 3 for JDBC connection tracking.

When tracking the JDBC connection in Type 1, if the Java thread that processes the transaction creates another Java thread to process the JDBC, then the JDBC processed by the Java thread cannot be monitored. In this case, set the enable_non_servlet_thread_jdbc_trace option of the JENNIFER agent to true.

```
enable_non_servlet_thread_jdbc_trace = true
```

Notice: Restart is needed.

When tracking the JDBC connection in Type 1, if the Java application initializes the data source in the background Java thread, no JDBC monitoring will be performed. In this case, set the enable_non_servlet_thread_jdbc_trace option of the JENNIFER agent to true.

A failure can occur when using the EJB entity bean. In this case, use Type 3, instead of Type 1 for JDBC connection tracking.

9.7.2.2. Type 2 - DriverManager

Type 2 refers to when the Java application obtains the vava.sql.Connection object from the java.sql.DriverManager class. In this case, JDBC connections are not pooled, so you cannot check the number of JDBC connections.

Warning: If a single transaction processes too many SQLs, or if it is necessary to obtain and return a connection too frequently, the rate of resource use is likely to be very high as JENNIFER is installed.

In Type 2, the JENNIFER agent uses the user_defined_jdbc_connectionpool_prefixes option to track the JDBC connections. The configuration may vary depending on the database and the JDBC driver to be used.

The following code is used for the Oracle database.

```
Class.forName("oracle.jdbc.driver.OracleDriver");
java.sql.Connection con = java.sql.DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:ORACL","scott", "tiget");
```

Include the beginning of the first parameter of the getConnection method of the java.sql.DriverManager class in the user_defined_jdbc_connectionpool_prefixes option of the JENNIFER agent.

```
user_defined_jdbc_connectionpool_prefixes = jdbc
```

If multiple databases are used and you want to monitor a specific database, set the user_defined_jdbc_connectionpool_prefixes option in a more specific manner..

```
user_defined_jdbc_connectionpool_prefixes = jdbc:oracle:thin:@local
```

If you want to check whether or not the JDBC connection object is returned when this option is used, set the user_defined_jdbc_ignore_close option of the JENNIFER agent to false.

```
user_defined_jdbc_ignore_close = false
```

9.7.2.3. Type 3 - Arbitrary class

Type 3 refers to when the Java application obtains the java.sql.Connection object from a class that operates as a connection pool. This type is used for a connection pool library such as Apache DBCP, or a framework such as Redhat Hibernate or Apache BATIS.

Notice: Strictly speaking, regardless of the JDBC connection pooling, if the java.sql.Connection object is obtained and returned to and from a specific method of the class, it is Type 3. Therefore, if you want to use Type 3, you must first understand the application source code.

For example, if the Java application obtains the java.sql.Connection objects from the get-Connection method of the example.ConnectionPool class, set it as follows.

```
jdbc_connection_get = example.ConnectionPool.getConnection()
```

Notice: Restart is needed.

If the method has parameters, they must be described as well.

```
jdbc_connection_get = example.ConnectionPool.getConnection(String)
```

If the releaseConnection method of the example.ConnectionPool class returns the java.sql.Connection object, set it as follows:

```
jdbc_connection_close =
example.ConnectionPool.releaseConnection(Connection)
```

Notice: Restart is needed.

In this case, the parameters of the method must include a java.sql.Connection type.

However, if the close method of the java.sql.Connnection object returns a value, do not need to set the jdbc_connection_close option of the JENNIFER agent.

If the method for returning the java.sql.Connection object is unknown, set the jdbc_connection_justget option of the JENNIFER agent.

```
jdbc_connection_juestget =
example.ConnectionPool.getConnection(String)
```

Notice: Restart is needed.

If the jdbc_connection_juestget option of the JENNIFER agent is used, you are not able to check the number of JDBC connections, or whether the JDBC connection object has been returned.

Cautions regarding the use of Type 3.

- Use a class name that includes a package name.
- However, the parameter of the method can use a class name that excludes a package name.
- The returned value for the method set by the jdbc_connection_get or jdbc_connection_justget option must be a java.sql.Connection type.
- At least one of the parameter of the method set by the jdbc_connection_close option must be a java.sql.Connection type. But the parameter location is irrelevant to this.
- Every option can include multiple entities separated by semicolons [;].

For Redhat Hibernate, set it as follows:

```
jdbc_connection_justget =
org.hibernate.jdbc.ConnectionManager.getConnection()
```

For Apache iBATIS, set it as follows:

```
jdbc_connection_get =
com.ibatis.sqlmap.engine.transaction.jdbc.JdbcTransaction.getConnectio
n()
```

For Apache DBCP, set it as follows:

```
jdbc_connection_get =
org.apache.commons.dbcp.PoolingDataSource.getConnection()
```

9.7.3. DB Connection Monitoring

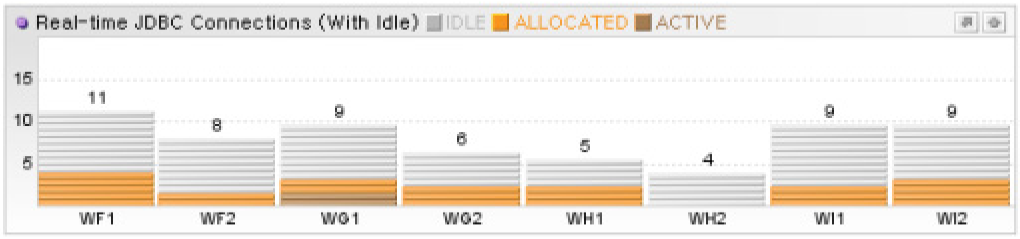
When Type 1 or 3 is used for JDBC monitoring, you can check the number of JDBC connections. The available types of JDBC connections are as follows:

Table 9-6: JDBC Connection Type

Type	Description
Idle DB connection	The number of DB connections waiting in the DB connection pool
Allocated DB connection	The number of DB connections that are not currently performing SQL queries among all DB connections assigned to the Java application thread by the DB connection pool
Active DB connection	The number of DB connections that are currently performing SQL queries among all DB connections assigned to the Java application thread by the DB connection pool

The number of DB connections is the general performance data collected by the JENNIFER agent. The data on the time can be viewed on an equalizer and runtime line chart.

Figure 9-14:Real-Time DB Connections



In addition, the five-minute average of JDBC connections is saved in the JDBC_IDLE, JDBC_ALLOC and JDBC_ACTIVE columns of the PERF_X_01~31 tables.

• In case that the number of JDBC connection is not monitored

Key connections with each DB and JDBC driver are listed up in the JENNIFER. If the unregistered JDBC driver is used, the number of JDBC connections is not monitored.

In this case, find the unregistered key connection of the JDBC driver and set it into the connection_trace_class option of the JENNIFER agent. If the connections are more than two, define as a semicolons [;].

```
connection_trace_class = com.ibm.db2.jcc.b.bb;com.ibm.db2.jcc.b.o
```

Notice: Restart is needed.

Especially, set this option for the the ket connection class of the DB2 JDBC driver because it is changed base on the version. You can find the key connection by tracing stactrace related to TCP port number which is used to connect the DB.

9.7.4. SQL Data Collection Method

JDBC monitoring involves the collection of an excessive amount of data compared to the monitoring of other types of resources. It has to collect all of the SQL and the parameters executed by the Java application, so that it can tune the problematic SQL.

Therefore, to minimize the amount of data transmitted, the JENNIFER agent uses the SQL hash values, not the SQL on the data sent to the JENNIFER server. It collects the SQL corresponding to the hash value and stores it in the SQLS table of the JENNIFER server.

To reduce the amount of SQL stored in the SQLS table, the JENNIFER agent processes the constants in the SQL as parameters and saves them. For example, the Java application processes the following SQL as additional SQL.

```
SELECT * FROM TABLE WHERE ID = ? AND AGE = 32 AND NAME = 'KIM'
SELECT * FROM TABLE WHERE ID = ? AND AGE = 27 AND NAME = 'LEE'
```

However, excluding the constants in the above code, the basic structure is the same. If the SQL is saved in the table, the data size would be too large. For example, if the following code is executed, 10,000 SQLs would need to be saved in the SQLS table.

```
Statement stmt = ...;
ResultSet rs = null;
for (int i = 0; i < 10000; i++) {
   rs = st.executeQuery("SELECT * FROM USERS WHERE ID = " + i);
   ...
}
```

To resolve this problem, the characters are converted to [$] and the numbers are converted to [#] before being saved in the SQLS table.

Therefore, only the following SQL is only saved in the SQLS table.

```
SELECT * FROM TABLE WHERE ID = ? AND AGE = # AND NAME = '$'
```

In the X-view profile data, the constant parameters are expressed as 1 and the binding parameters are expressed as 2. The binding parameters refer to the parameters expressed as [?] in the SQL executed by the java.sql.PreparedStatement object.

However, sometimes, the string of binding parameters can not be expressed properly. In this case, set the encoding for the SQL binding parameter of java.sql.PreparedStatement using the sqlparam_encoding option of the JENNIFER agent.

```
sqlparam_encoding = 8859_1
```

For instance, if DB encoding and application encoding are different each other, only SQL binding parameter does not represent properly even though other profile data is shown. This problem may occur when you change the encoding before configuring application SQL parameter. Use the option to solve this problem.

9.7.5. Oracle Dependency

If Oracle-dependent codes are used when monitoring interconnections with the Java applications using Oracle database, a java.lang.ClassCastException may occur. This could be manifested in a case in which the CLOB and the BLOB are processed as follows:

```
Statement stmt = ...;
ResultSet rs =
stmt.executeQuery("SELECT TEXT FROM TEST_CLOB WHERE ID =1 FOR UPDATE");
if (rs. next()) {
    oracle.sql.CLOB cl =
           ((oracle.jdbc.OracleResultSet) rs).getCLOB("TEXT");
    ....
}
```

In this case, set the enable_jdbc_oracle_dependency_used option of the JENNIFER agent to true.

```
enable_jdbc_oracle_dependency_used = true
```

Notice: Restart is needed.

If you set this option to true when using other types of database, it will not cause any problems.

After setting the enable_jdbc_oracle_dependency_used option to true, JDBC monitoring may not be achieved, but the following message may appear in the X-view profile data.

```
enable_jdbc_oracle_dependency_used = true, but Oracle Driver not found
```

In this case, it means that the option is set to true even if the Oracle JDBC driver does not exist in the Java application class path, or that the JDBC file cannot be referred to by the jennifer.jar file.

After setting the enable_jdbc_oracle_dependency_used option as true, you can unwrap the interior object when the object is not a Oracle JDBC object..

Notice: Application server uses Wrapper class when it implements a JDBC connection pool. Therefore, unwrap means that the server take the original JDBC object from the wrapper ones.

Set the name of unwraped method by using the JENNIFER agent option ending the unwrap_method.

```
jdbc_connection_unwrap_method = getVendorObj
jdbc_statement_unwrap_method = getVendorObj
jdbc_preparedstatement_unwrap_method = getVendorObj
jdbc_callablestatement_unwrap_method = getVendorObj
jdbc_resultset_unwrap_method = getVendorObj
```

Notice: Restart is needed.

You can set the unwrap method on each JDBC object. If the method names for the JDBC object are same, just set the jdbc_unwrap_method option of the JENNIFER agent.

```
jdbc_unwrap_method = getVendorObj
```

Notice: Restart is needed.

Unwrap the method depending on the circumstances.

Notice: The wrapper class that has the WebLogic JDBC connection pool implements the Oracle JDBC API. Therefore, if you are using WebLogic, you do not need to unwrap the class.

9.7.6. JDBC Vendor Dependency

If you install the JENNIFER on the Java application which is programmed with the dependent code for the Java application server, an error would occur. It is because that the wrapper class for JDBC monitoring is not compatible with the class for implementing the JDBC connection pool.

To solve this problem, you have to make the the wrapper class for JDBC monitoring compatible with the class for implementing the JDBC connection pool.

At first, set the enable_jdbc_vendor_wrap option of the JENNIFER agent to true.

```
enable_jdbc_vendor_wrap = true
```

Notice: Restart is needed.

If you set this option, JENNIFER ignores the enable_jdbc_oracle_dependency_used option.

Each wrapper class for implementing the JDBC connection pool is different depending on the kinds/version of the Java application server and database. That means you have to define each wrapper class. The wrapper class generated by the JENNIFER has JWP after the class name. This is an example showing how to set the properties.

• WebLogic 9.x + DataSource + Oracle

```
weblogic.jdbc.wrapper.PoolConnection_oracle_jdbc_driver_T4CConnectionJWP(S) =
    weblogic.jdbc.wrapper.PoolConnection_oracle_jdbc_driver_T4CConnection
weblogic.jdbc.wrapper.Statement_oracle_jdbc_driver_T4CStatementJWP(S) =
    weblogic.jdbc.wrapper.Statement_oracle_jdbc_driver_T4CStatement
weblogic.jdbc.wrapper.PreparedStatement_oracle_jdbc_driver_T4CPreparedStatementJWP(S) =
    weblogic.jdbc.wrapper.PreparedStatement_oracle_jdbc_driver_T4CPreparedStatement
weblogic.jdbc.wrapper.CallableStatement_oracle_jdbc_driver_T4CCallableStatementJWP(S) =
    weblogic.jdbc.wrapper.CallableStatement_oracle_jdbc_driver_T4CCallableStatement
weblogic.jdbc.wrapper.ResultSet_oracle_jdbc_driver_OracleResultSetImplJWP(S) =
    weblogic.jdbc.wrapper.ResultSet_oracle_jdbc_driver_OracleResultSetImpl
```

Notice: Restart is needed.

• WebLogic 6.x + DataSource

```
weblogic.jdbc.pool.ConnectionJWP(S) = weblogic.jdbc.pool.Connection
weblogic.jdbc.pool.StatementJWP(S) = weblogic.jdbc.pool.Statement
weblogic.jdbc.pool.PreparedStatementJWP(S) = weblogic.jdbc.pool.PreparedStatement
weblogic.jdbc.pool.CallableStatementJWP(S) = weblogic.jdbc.pool.CallableStatement
weblogic.jdbc.pool.ResultSetJWP(S) = weblogic.jdbc.pool.ResultSet
```

Notice: Restart is needed.

• DriverManager + Oracle

```
oracle.jdbc.driver.T4CConnectionJWP(S) = oracle.jdbc.driver.OracleConnection
oracle.jdbc.driver.T4CStatementJWP(S) = oracle.jdbc.driver.T4CStatement
oracle.jdbc.driver.T4CPreparedStatementJWP(S) = oracle.jdbc.driver.T4CPreparedStatement
oracle.jdbc.driver.T4CCallableStatementJWP(S) = oracle.jdbc.driver.T4CCallableStatement
oracle.jdbc.driver.OracleResultSetImplJWP(S) = oracle.jdbc.driver.OracleResultSetImpl
```

Notice: Restart is needed.

• Tomcat 6.x + DataSource

```
org.apache.tomcat.dbcp.dbcp.PoolingDataSource$PoolGuardConnectionWrapperJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingConnection
org.apache.tomcat.dbcp.dbcp.DelegatingStatementJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingStatement
org.apache.tomcat.dbcp.dbcp.DelegatingPreparedStatementJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingPreparedStatement
org.apache.tomcat.dbcp.dbcp.DelegatingCallableStatementJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingCallableStatement
org.apache.tomcat.dbcp.dbcp.DelegatingResultSetJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingResultSet
```

Notice: Restart is needed.

• WebSphere 5.x + DataSource

```
com.ibm.ws.rsadapter.jdbc.WSJdbcConnectionJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcConnection
com.ibm.ws.rsadapter.jdbc.WSJdbcStatementJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcStatement
com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatementJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatement
com.ibm.ws.rsadapter.jdbc.WSJdbcCallableStatementJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcCallableStatement
com.ibm.ws.rsadapter.jdbc.WSJdbcResultSetJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcResultSet
```

Notice: Restart is needed.

• Only Oracle JDBC

By the way, if the dependency of the application on the JDBC driver is clear, then you can simply set it as follows.

```
java.sql.Connection(S) = oracle.jdbc.driver.OracleConnection
java.sql.Statement(S) = oracle.jdbc.driver.OracleStatement
java.sql.PreparedStatement(S) =
oracle.jdbc.driver.OraclePreparedStatement
java.sql.CallableStatement(S) =
oracle.jdbc.driver.OracleCallableStatement
java.sql.ResultSet(S) = oracle.jdbc.driver.OracleResultSet
```

9.7.7. Oracle SID Checking Functions

In the JDBC tab of the **[Real Time Monitoring | Application]** menu, you can check the Oracle SID. It requires some additional configurations, which vary significantly depending on the database version.

For the Oracle 9i database, set the dbsession_query option of the JENNIFER agent as follows:

```
dbsession_query = \
oracle.jdbc.driver.OracleConnection: \
select SYS_CONTEXT ('USERENV','SESSIONID') sid from dual;
```

For the Oracle 10g database, set the dbsession_query option of the JENNIFER agent as follows:

```
dbsession_query = \
oracle.jdbc.driver.PhysicalConnection: \
select SYS_CONTEXT ('USERENV','SESSIONID') sid from dual;
```

9.7.8. SQL Exception Log Recording

When a java.sql.SQLException occurs in the Java application, you can record it in the JENNIFER agent log file. To record it, set the enable_sql_error_trace option of the JENNIFER agent to true. By default, it is set to false.

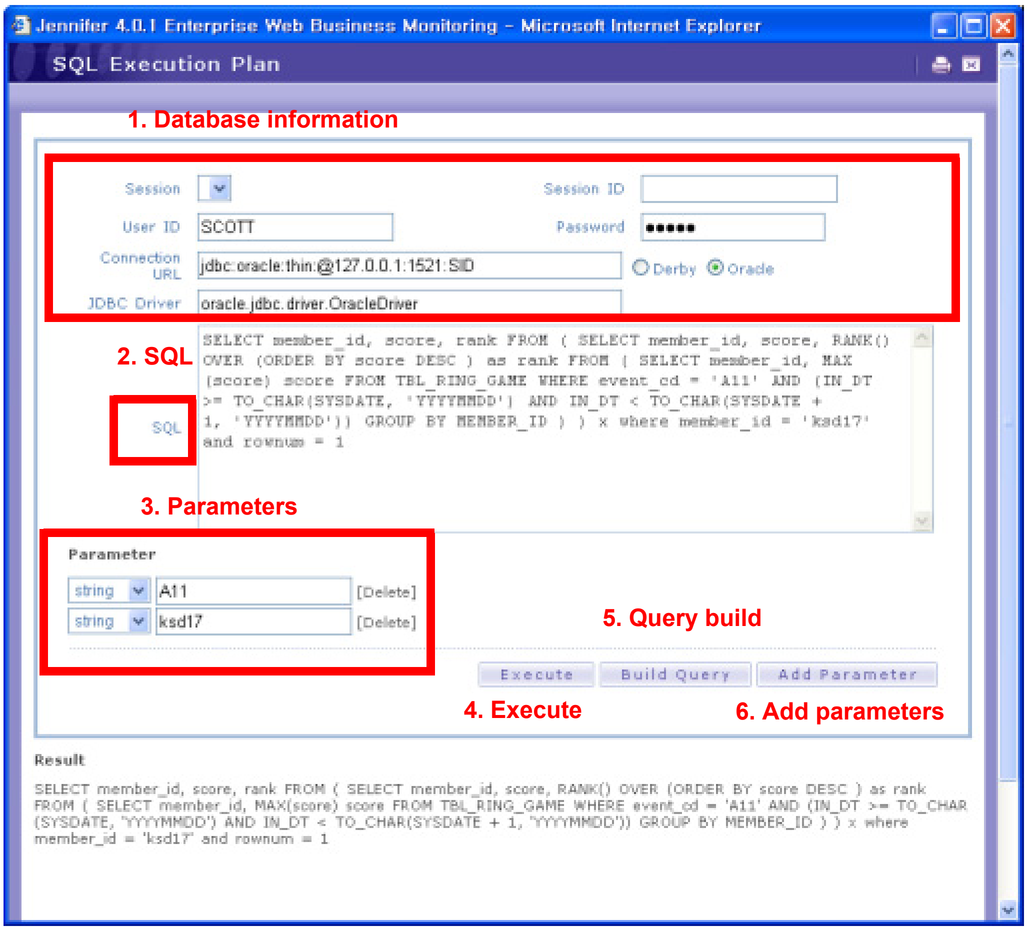
```
enable_sql_error_trace = true
```

However, in a normal situation, a java.sql.SQLException may occur, dependently on the business logic. Therefore, a java.sql.SQLException must not be interpreted as an error or failure. In addition, if you record to the log too often due to the frequent occurrence of java.sql.SQLException that occurs too frequently, it may lead to performance degradation. Set this option to true only if necessary, and in normal situations, set it to false.

9.7.9. SQL Execution Plan

If you right-click on the SQL tab of the X-View profile, a contextual menu will appear.Select the **[Query Build]** menu to launch a pop-up window, in which you can check the SQL execution plan.

Figure 9-15:SQL Execution Plan



Session:
Session ID:
User ID: SCOTT
Password: ●●●●●
Connection URL: jdbc:oracle:thin:@127.0.0.1:1521:SID
Derby [✓ Oracle]
JDBC Driver: oracle.jdbc.driver.OracleDriver

SQL

```
SELECT member_id, score, rank FROM ( SELECT member_id, score, RANK()
OVER (ORDER BY score DESC ) as rank FROM ( SELECT member_id, MAX
(score) score FROM TBL_RING_GAME WHERE event_cd = 'A11' AND (IN_DT
>= TO_CHAR(SYSDATE, 'YYYYMMDD') AND IN_DT < TO_CHAR(SYSDATE +
1, 'YYYYMMDD')) GROUP BY MEMBER_ID ) ) x where member_id = 'ksd17'
and rownum = 1
```

Parameter

string	A11	[Delete]
string	ksd17	[Delete]

Result

```
SELECT member_id, score, rank FROM ( SELECT member_id, score, RANK() OVER (ORDER BY score DESC ) as rank
FROM ( SELECT member_id, MAX(score) score FROM TBL_RING_GAME WHERE event_cd = 'A11' AND (IN_DT >= TO_CHAR
(SYSDATE, 'YYYYMMDD') AND IN_DT < TO_CHAR(SYSDATE + 1, 'YYYYMMDD')) GROUP BY MEMBER_ID ) ) x where
member_id = 'ksd17' and rownum = 1
```

1. Database information - Enter the database information to check the SQL execution plan.
2. SQL - The SQL selected in the X-View profile tab is automatically entered. Parameter 1 is automatically replaced, and the binding parameter shown as parameter 2 is displayed in the bottom.
3. Parameter - The binding parameter or parameter 2 is displayed. You can modify the parameter value on your own.
4. Execute - If you press this button, the SQL execution plan will be displayed at the bottom.
5. Query build - If you press this button, the SQL that binds the binding parameter will be displayed at the bottom.
6. Add parameters - If you click this button, a parameter will be added.

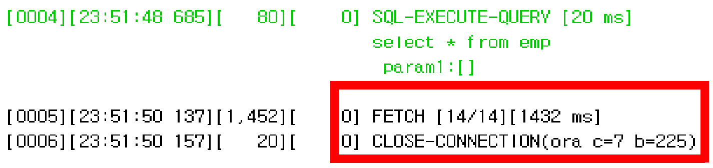
Notice: If you want to check the SQL execution plan, copy the JDBC driver for the database into the JENNIFER_HOME/server/common/lib directory and restart the JENNIFER server.

9.7.10.Checking the Database Resource Utilization by the Session for the SQL

If you set the enable_dbstat option of the JENNIFER agent to true, you can check the database resource utilization by the session for the SQL in the CLOSE-CONNECTION message of the X-view profile data.

```
enable_dbstat = true
```

Figure 9-16:Database Utilization by the Session for the SQL



```
[0004][23:51:48 685][   80][    0] SQL-EXECUTE-QUERY [20 ms]
                                   select * from emp
                                    param1:[]

[0005][23:51:50 137][1,452][    0] FETCH [14/14][1432 ms]
[0006][23:51:50 157][   20][    0] CLOSE-CONNECTION(ora c=7 b=225)
```

Here, ora stands for oracle, c stands for CPU utilization(1/100 seconds) and b stands for Oracle's read block(Logical/Physical IO Block).

9.7.11.JDBC Monitoring and Exceptions

The following exceptions related to JDBC monitoring may occur.

9.7.11.1. Delayed SQL Response Time

When the SQL response time exceeds the threshold, the WARNING_JDBC_BAD_RESPONSE exception will occur. The threshold can be set using the sql_bad_responsetime option of the JENNIFER agent. The default is 20000, and it is expressed in milliseconds.

```
sql_bad_responsetime = 20000
```

9.7.11.2. Excessive Fetch Counts

To "Fetch" is to call the next method of the java.sql.ResultSet object. This refers to the number of hits for the next method. If the fetch count for the same java.sql.ResultSet object exceeds the threshold while executing a transaction, then the WARNING_JDBC_TOOMANY_RS_NEXT exception will occur. The threshold can be set using the jdbc_resultset_warning_fetch_count option of the JENNIFER agent. The default is 10000.

```
jdbc_resultset_warning_fetch_count = 10000
```

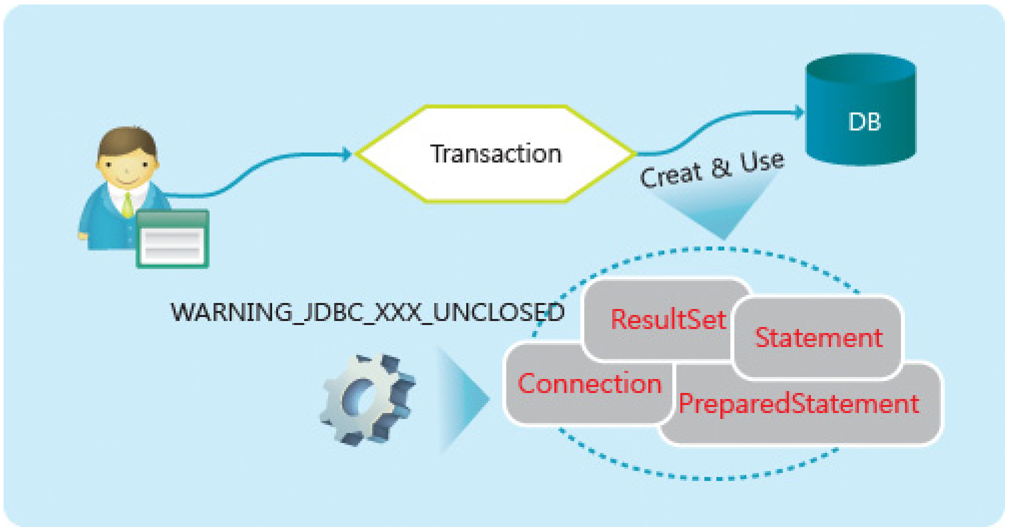
9.7.11.3. Non-returned JDBC Resource Tracking

The java.sql.Connection, java.sql.Statement and java.sql.ResultSet objects are called the JDBC resource. If the close method of the object is not called after the object is used, a non-returned JDBC resource exception will occur.

The problem is the question of when the non-returned JDBC resource must be detected. It is desirable to detect the non-returned JDBC resource when the transaction is terminated, but if the JDBC resource is used and returned by the background Java thread, then an incorrect exception may occur. To resolve this problem, when garbage collection is performed on the JDBC resource, JENNIFER checks whether or not the close method is executed, and determines whether the JDBC resource is returned or not.

Therefore, if no garbage collection is performed, then the non-returned JDBC resource exception might not occur. Even if garbage collection is performed, there could be a time difference between the time of the terminated transaction and the time of the non-returned JDBC resource exception. In other words, an exception can be delayed.

Figure 9-17:Non-returned JDBC Resource Tracking



Eventually, it is impossible to determine whether or not the JDBC resource is returned when the transaction is terminated. Therefore, the X-view transaction data or the application processing statistic data will show that the non-returned JDBC resource exception has not occurred.

The following exceptions are related to the non-returned JDBC resource.

• WARNING_DB_CONN_UNCLOSED

If the close method is not called after the Java application uses the java.sql.Connection object, then the WARNING_JDBC_CONN_UNCLOSED exception will occur.

• WARNING_JDBC_STMT_UNCLOSED

If the close method is not called after the Java application uses the java.sql.Statement object, then the WARNING_JDBC_STMT_UNCLOSED exception will occur.

• WARNING_JDBC_PSTMT_UNCLOSED

If the close method is not called after the Java application uses the java.sql.Prepared-Statement object, then the WARNING_JDBC_PSTMT_UNCLOSED exception will occur.

• WARNING_JDBC_CSTMT_UNCLOSED

If the close method is not called after the Java application uses the java.sql.CallableStatement object, then the WARNING_JDBC_CSTMT_UNCLOSED exception will occur.

• WARNING_JDBC_RS_UNCLOSED

If the close method is not called after the Java application uses the java.sql.ResultSet object, then the WARNING_JDBC_RS_UNCLOSED exception will occur.

To resolve the problem with the non-returned JDBC resource, JENNIFER provides stacktrace. In general, this is sufficient, but sometimes a more detailed stacktrace is required. In this case, set the enable_jdbc_XXXX_fullstack_trace option of the JENNIFER agent to true. By default, it is set to false.

```
enable_jdbc_connection_fullstack_trace = true
enable_jdbc_callablestatement_fullstack_trace = true
enable_jdbc_preparedstatement_fullstack_trace = true
enable_jdbc_statement_fullstack_trace = true
enable_jdbc_resultset_fullstack_trace = true
```

Warning: The detailed stacktrace may overload the system. For this reason, use it only when it is necessary.

You can adjust the configuration, so that the non-returned JDBC resource can be arbitrarily returned by the JENNIFER agent. Set the enable_auto_XXXX_close option of the JENNIFER agent. By default, it is set to true.

```
enable_auto_connection_close = true
enable_auto_statement_close = true
enable_auto_preparedstatement_close = true
enable_auto_callablestatement_close = true
enable_auto_resultset_close = true
```

However, it is very dangerous to automatically return the java.sql.Connection object while it is being managed by the Java application server or connection pool library.

9.7.11.4. Oracle XML DB XMLType Leakage Tracking

If the close method is not called after using the oracle.xdb.XMLType object when the Oracle XML DB is used, the WARNING_ORACLE_XMLTYPE_UNCLOSED exception will occur. However, this may occur only if the enable_oracle_xdb_xmltype_trace option of the JENNIFER agent is set to true. By default, it is set to false.

```
enable_oracle_xdb_xmltype_trace = true
```

9.7.11.5. Uncommit/Rollback Statement Tracking

If the JDBC transaction is explicitly terminated (by calling the rollback or commit method of the java.sql.Connection object) after the Java application starts the JDBC transaction (the parameter is set to false and thesetAutoCommit method of the java.sql.Connection object is called), and it calls the execute, executeBatch and executeUpdate methods of the java.sql.Statement object, the WARNING_JDBC_UN_COMMIT_ROLLBACK exception will occur. In this case, you can automatically rollback the java.sql.Connection objects using the enable_rollback_uncommited_close option of the JENNIFER agent. Default is false.

```
enable_rollback_uncommited_close = true
```

Warning: You are recommended to use this option only for a system that has already been verified. If the rollback method of another Java thread is called when the close method of the java.sql.Connection object is called, then it could lead to a failure.

However, if the SELECT text is executed by the execute method of the java.sql.Statement object, then this exception does not accurately describe the actual situation. Therefore, you can deactivate this exception by using the ignore_rollback_uncommited_error option of the JENNIFER agent.

```
ignore_rollback_uncommited_error = true
```

Notice: In the long run, modifying the source code is recommeded so that the SELECT text is processed by the executeQuery method.

9.7.11.6. Duplication assignment exception caused by the DB duplication pooling structure

- Although it is not duplicate assignments, due to the internal application structure, this alarm can be issued. Such an internal application structure that causes this phenomenon is called a duplicate pooling structure.
- For instance, if JENNIFER performs DataSource(type 1) or DriverManager(type 2) JDBC monitoring in the Java application and another connection pool exists inside the application, then JENNIFER recognizes that one connection is not returned but it is used by multiple threads and thus it causes a WARNING_DB_CONN_ILLEGAL_ACCESS exception.
- Therefore, if a WARNING_DB_CONN_ILLEGAL_ACCESS exception occurs in the Java application, then you must first check the whether it is a duplicate pooling structure and then change the DB connection trace setting. When the JENNIFER configuration is incorrect, a WARNING_DB_CONN_ILLEGAL_ACCESS exception may occur. For example, if Type 1 and 3 are set simultaneously, a WARNING_DB_CONN_ILLEGAL_ACCESS exception can occur. In this case, change the DB connection tracking configuration.

9.8. User Defined Resource Monitoring

In addition to the JDBC resource, you can monitor leakages of arbitrary resources. For example, you can pool the java.net.Socket object to improve the network performance.

However, not every type of resource leakage can be monitored. This function can be used only when the method that obtains and returns the resource is clearly defined. If the hits for the method that obtains the resource are not the same as the hits for the method that returns it, then the WARNING_RESOURCE_LEAK exception can occur.

User defined resource monitoring can be set in the leakcheck_XXXX_get and leakcheck_XXXX_close option of the JENNIFER agent.

For example, if you have the mypool and yourpool resource pools, then you can set the method that obtains the resource by using the leakcheck_XXXX_get option of the Jen-nier agent. Enter the resource pool name in XXXX.

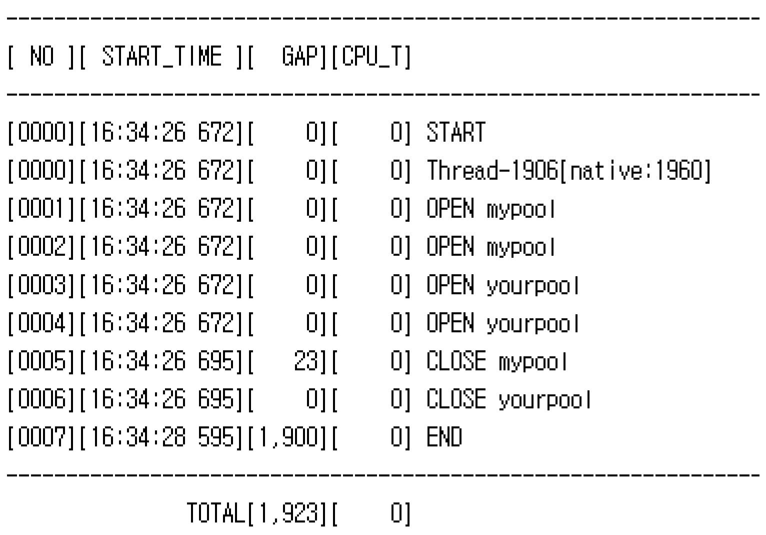
```
leakcheck_mypool_get = example.Leak.get();
leakcheck_yourpool_get = example.Leak2.get();
```

By using the leakcheck_XXXX_close option of the JENNIFER agent, you can also set the method that returns the resource. Enter the resource pool name in XXXX.

```
leakcheck_mypool_close = example.Leak.release(Object);
leakcheck_yourpool_close = example.Leak2.release(Object);
```

The profile of the user defined resource pools can be viewed in the X-view profile data.

Figure 9-18:X-view Profile for the User Defined Resource Pool



```
-----------------------------------------------------------------
[ NO ][ START_TIME ][  GAP][CPU_T]
-----------------------------------------------------------------
[0000][16:34:26 672][    0][    0] START
[0000][16:34:26 672][    0][    0] Thread-1906[native:1960]
[0001][16:34:26 672][    0][    0] OPEN mypool
[0002][16:34:26 672][    0][    0] OPEN mypool
[0003][16:34:26 672][    0][    0] OPEN yourpool
[0004][16:34:26 672][    0][    0] OPEN yourpool
[0005][16:34:26 695][   23][    0] CLOSE mypool
[0006][16:34:26 695][    0][    0] CLOSE yourpool
[0007][16:34:28 595][1,900][    0] END
-----------------------------------------------------------------
              TOTAL[1,923][    0]
```

9.9. Thread Monitoring(How to Perform CPU Tuning)

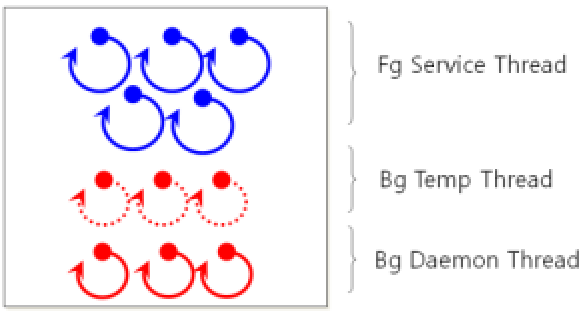
Jennifer provides a detailed thread monitoring function using the JMX in the Java1.5 environment. Jennifer is a service oriented monitoring solution. It tracks the service performance based on the threads processing the user requests.

By the way, sometimes, it is necessary to tune the CPU usage by a process. In this case, you must perform thread oriented monitoring. In other words, you must identify which thread is using excessive CPU and tune the thread. The threads are classified into service threads and background threads.

At a web application server, a service thread generally has a standardized thread name. Therefore, from the thread name, we can differentiate a service thread and a background thread..

Then, we can determine which service or background thread is using excessive CPU and perform tuning on the indentified thread.

Figure 9-19:Thread Monitoring



Jennifer distinguishes the threads inside a process by their names and monitors the CPU usage, so that it can provide the basic information for CPU tuning.

9.9.1. Configuration and Environment

JENNIFER AGENT Configuration

Detailed thread monitoring can be used only in a Jennifer Agent environment of Java 1.5 or higher.

```
extra_enable=true
extra_agent_class=jennifer.extra.ThreadMon
extra_agent_classpath=/jennifer/agent/lwst40.custom.jar
extra_data_interval=3000
xtmon_fg_name=http-8080
xtmon_topn=20
xtmon_file_save=true
```

The Jennifer Agent’s monitoring expansion function, ExtraAgent is used.

- extra_enable : Activates the Extra Agent function.
- extra_agent_class : Extra Agent adaptor that will collect the thread information.
- extra_agent_classpath : Jar files including the Extra Agent module.
- extra_data_interval :Data collection period for the Extra Agent.
- xtmon_file_save : Stores the important data in a file among the thread monitoring data.
- xtmon_topn : Shows a list of top 20 most CPU usages by the threads.

- xtmon_fg_name : Can set multiple entities by using the service thread name rules, ';'.

JENNIFER SERVER Configuration

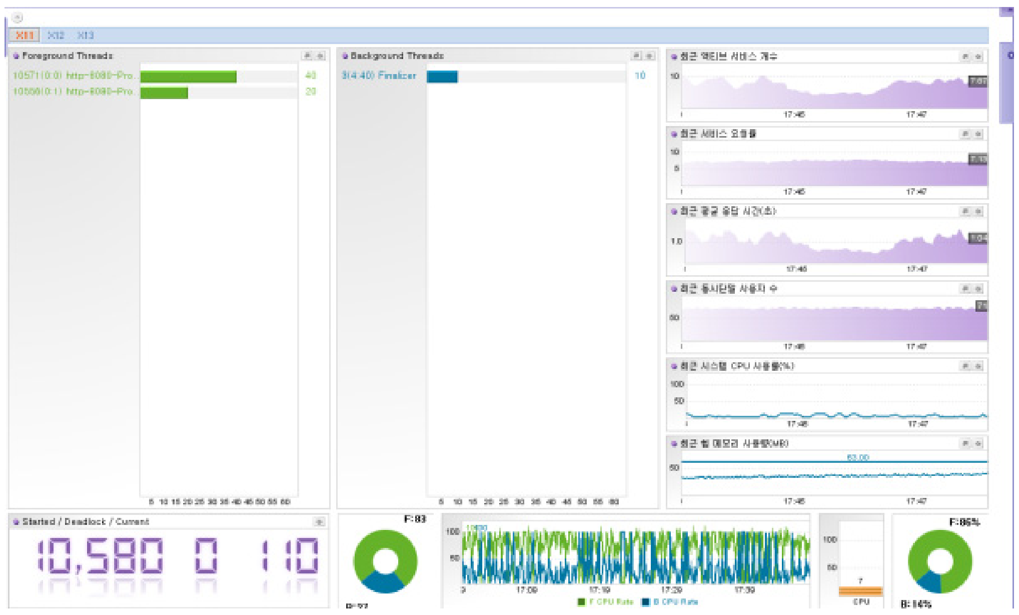
At the Jennifer Server, you must register the screen for detailed memory information monitoring. If the Jennifer 4.2 server is first started, the it will be automatically registered in [Real Time Monitoring| Thread] but only the privilege is omitted. But if it is a different version or the menu does not exist, then register the following URL in the menu.

```
real_thread.jsp
```

9.9.1.1. Interpretations of the Thread Dashboard

The following dashboard for detailed thread monitoring is used.

Figure 9-20:Thread Dashboard



Warning: You must select an agent in the upper area of the screen to show the graph.

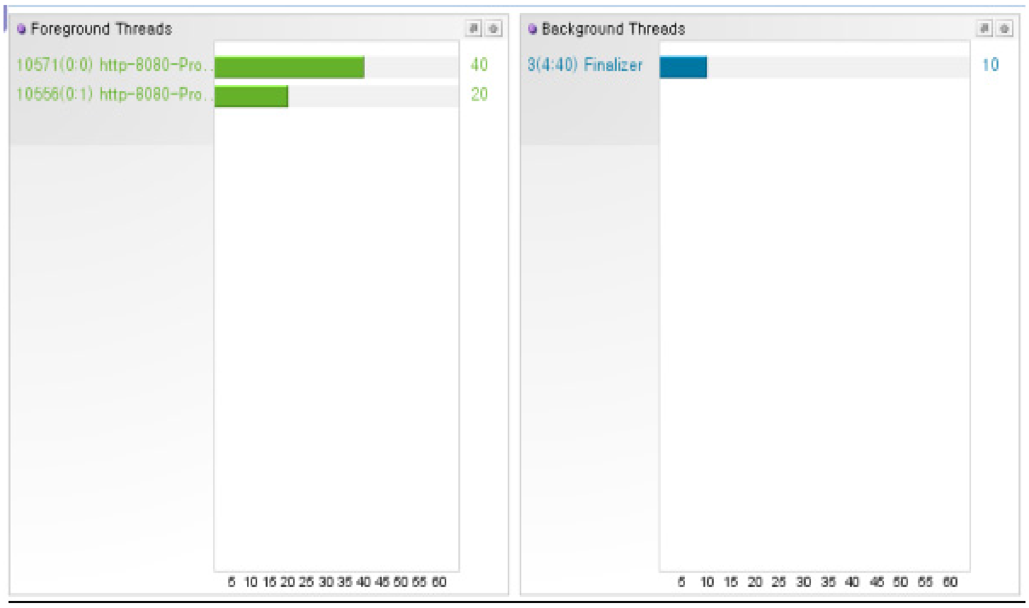
Important Graphs

The important graphs have the following contents.

Forground/Background Threads

Among the service(Forground) and background threads, it shows the list of longest CPU time in the unit time(basic 3000ms). If a specific thread has excessive CPU time consistently, then you must find the cause.

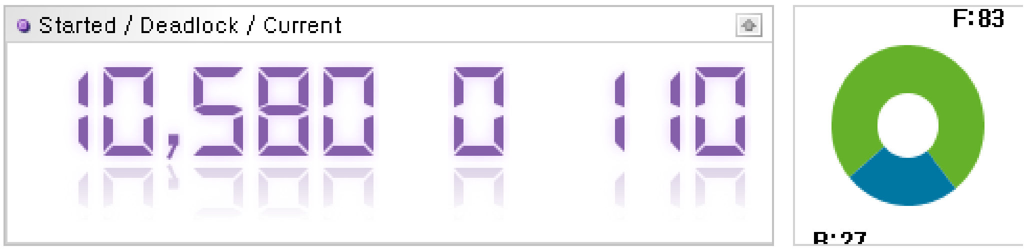
Figure 9-21:Forground/Background Threads



The left area of the bottom graphs in the dashboard shows how many threads are executed.

Thread Count

Figure 9-22:Thread Count



The left area of the bottom graphs in the dashboard shows how many threads are executed.

- Started : total number of threads executed after the Java process is started.
- Deadlock : total number of threads in deadlock (if it is not 0, there is a problem.)
- Current : thread being executed currently.

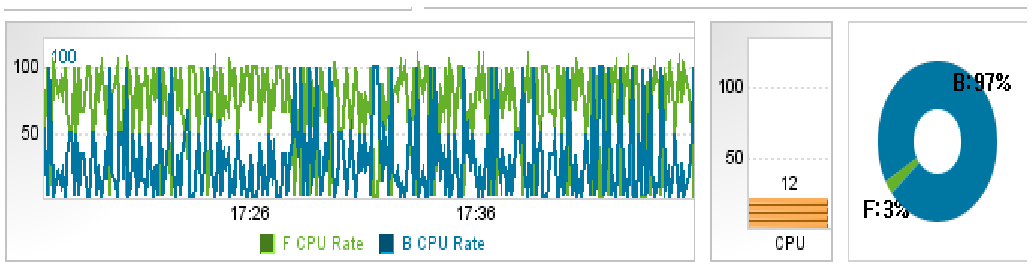
The graph showing F:99, B:99 indicates the ratio between the forground and background threads among the threads being currently executed.

CPU Usage Ratio

This graph indicates whether the background thread uses more CPU or the forground thread uses more CPU.

By referring to the system CPU usage, check the CPU usage ratio and study how to perform tuning on the CPU usage.

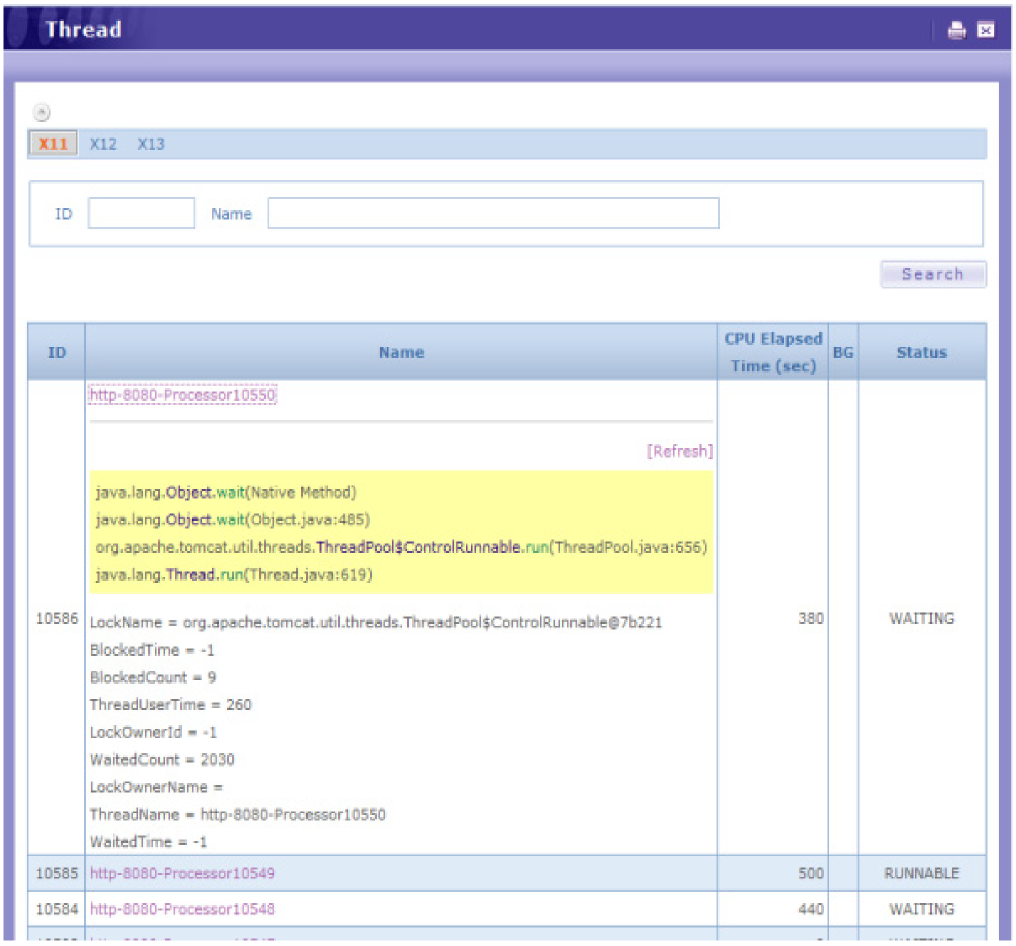
Figure 9-23:CPU Usage Ratio



Detailed Real Time Inquiry

You can inquire about the detailed stack information of the thread being executed.

Figure 9-24:Detailed Real Time Inquiry



ID	Name	CPU Elapsed Time (sec)	BG	Status

10586	http-8080-Processor10550
	[Refresh]
	java.lang.Object.wait(Native Method)
	java.lang.Object.wait(Object.java:485)
	org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
	java.lang.Thread.run(Thread.java:619)
	LockName = org.apache.tomcat.util.threads.ThreadPool$ControlRunnable@7b221
	BlockedTime = -1
	BlockedCount = 9
	ThreadUserTime = 260
	LockOwnerId = -1
	WaitedCount = 2030
	LockOwnerName =
	ThreadName = http-8080-Processor10550
	WaitedTime = -1
		380		WAITING

10585	http-8080-Processor10549	500		RUNNABLE
10584	http-8080-Processor10548	440		WAITING

You can inquire about the detailed stack information of the thread being executed.

You can find out when it started and how much CPU has been used. In addition, you can inquire about the call stack of the thread.

Notice: The CPU time for each thread indicates the total of CPU occupation time since execution of the thread. Looking at this value, you can determine which thread is using how much CPU.

9.10. Detailed Java Memory Monitoring

When you tune the Java process performance, the one thing you can't omit is memory tuning. By suggesting a strong heap memory management model, you can now enjoy more convenient and stable programming, but a memory still has the no. 1 priority among the management areas. Engine level management using the heap memory has stabilized a memory but caused excessive use of the CPU. The memory shortage is not due to a memory problem but it is due to excessive use of the CPU. So memory management and tuning are not regarded as an easy management area.

But, since Java5, we have had a much stronger means to monitor our memory. Using this means, you can monitor the heap memory or non heap memory status or make real time observations on the GC movement to release the memory or the status of overheads.

Jennifer uses this tool to help you monitor the memory status more intuitively from the perspectives of an administrator.

9.10.1.figuration and Environment

9.10.1.1. IFER AGENT Configuration

IFER AGENT Configuration

```
extra_enable=true
extra_agent_class=jennifer.extra.MemoryMon
extra_agent_classpath=/jennifer/agent/lwst40.custom.jar
extra_data_interval=3000
xmmon_file_save=true
```

The Jennifer Agent's monitoring expansion function, ExtraAgent is used.

- extra_enable : Activates the Extra Agent function.
- extra_agent_class : Extra Agent adaptor that will collect the thread information.
- extra_agent_classpath : Jar files including the Extra Agent module.
- extra_data_interval :Data collection period for the Extra Agent.
- xtmon_file_save : Stores the important data in a file among the thread monitoring data.

9.10.1.2. JENNIFER SERVER Configuration

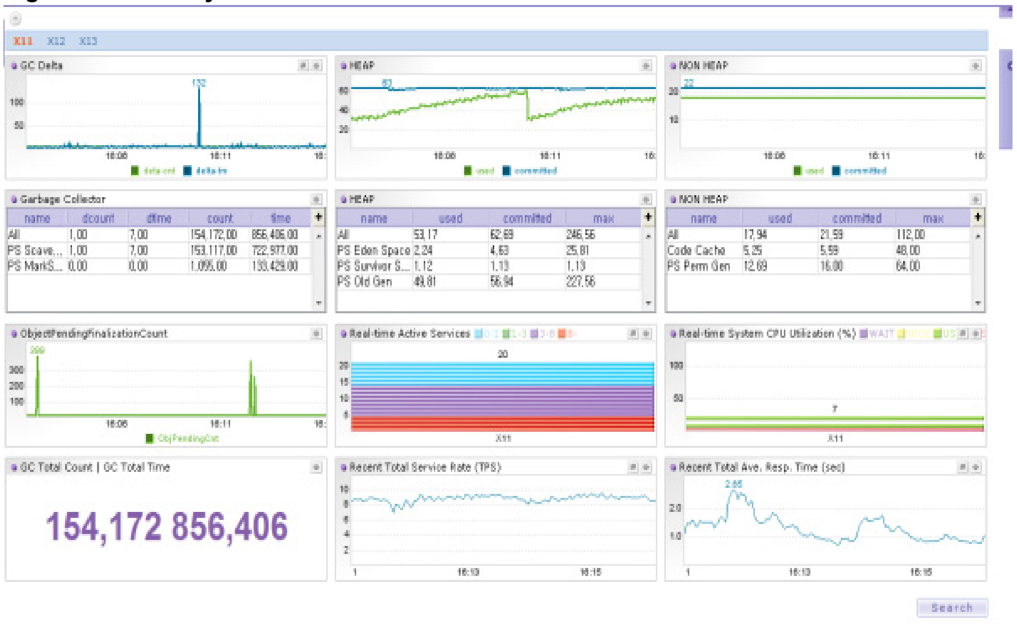
At the Jennifer Server, you must register the screen for detailed memory information monitoring. If the Jennifer 4.2 server is first started, the it will be automatically registered in [Real Time Monitoring| Thread] but only the privilege is omitted. But if it is a different version or the menu does not exist, then register the following URL in the menu.

```
real_memory.jsp
```

9.10.2.Interpretations of the Memory Dashboard

The following dashboard for detailed memory monitoring is used.

Figure 9-1:Memory Dashboard



X11 X12 X13
GC Delta
delta-cnt
delta-tm
HEAP
used
committed
NON HEAP
ObjectPendingFinalizationCount
ObjPendingCnt
Real-time Active Services
Real-time System CPU Utilization (%)
GC Total Count | GC Total Time
Recent Total Service Rate (TPS)
Recent Total Ave. Resp. Time (sec)
Search

Garbage Collector

name	dcount	dtime	count	time
All	1,00	7,00	154,172,00	856,406,00
PS Scave...	1,00	7,00	153,117,00	722,977,00
PS MarkS...	0,00	0,00	1,055,00	133,429,00

HEAP

name	used	commited	max
All	53,17	62,69	246,56
PS Eden Space	2,24	4,63	25,81
PS Survivor S...	1,12	1,13	1,13
PS Old Gen	49,81	56,94	227,56

NON HEAP

name	used	commited	max
All	17,94	21,59	112,00
Code Cache	5,25	5,59	48,00
PS Perm Gen	12,69	16,00	64,00

154,172 856,406

Notice: You must select an agent in the upper area of the screen to show the graph.

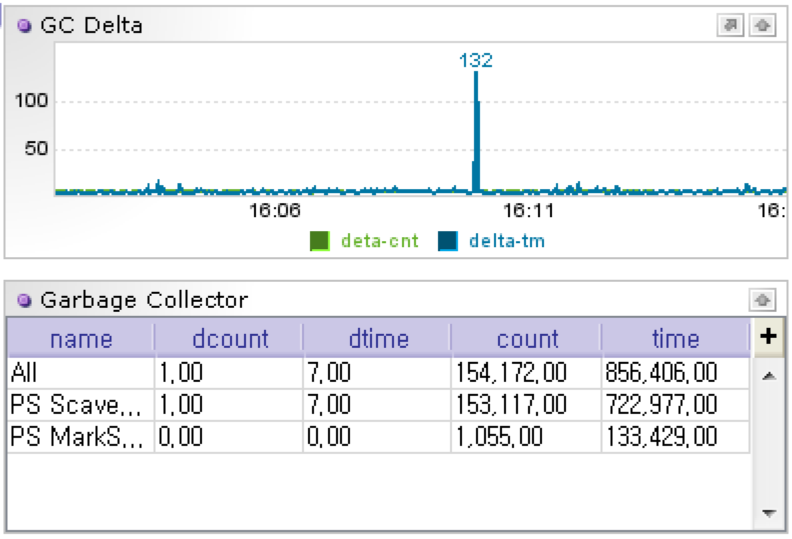
9.10.3.Important Graphs

The important graphs have the following contents.

9.10.3.1. GC Delta

The number of GC occurrences and time consumed for them in the unit time(basic 3000ms) displayed in a line graph. The elapsed time for GC occurrences can be monitored in real time.

Figure 9-2:GC Delta

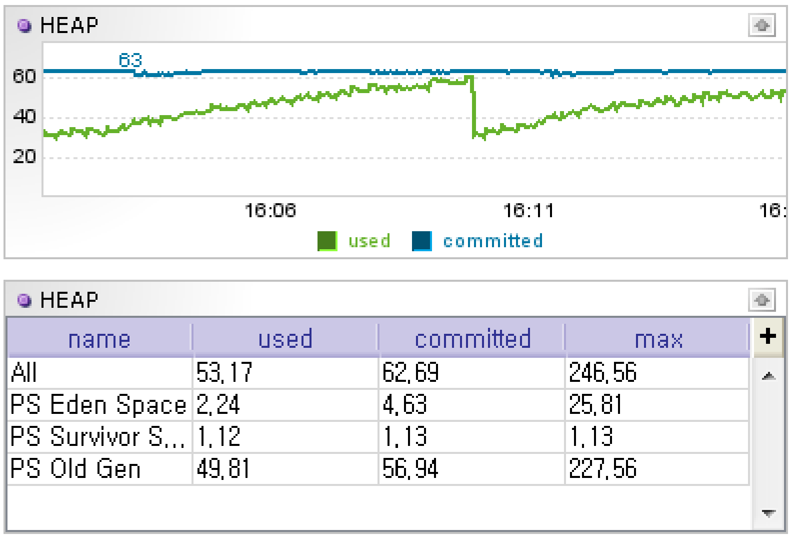


name	dcount	dtime	count	time
All	1,00	7,00	154,172,00	856,406,00
PS Scave,,,	1,00	7,00	153,117,00	722,977,00
PS MarkS,,,	0,00	0,00	1,055,00	133,429,00

9.10.3.2. Heap Usage in Each Area

The changes in the heap memory usage are monitored.

Figure 9-3:Heap Usage in Each Area

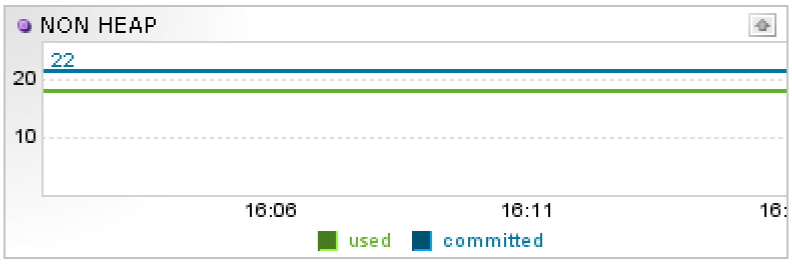


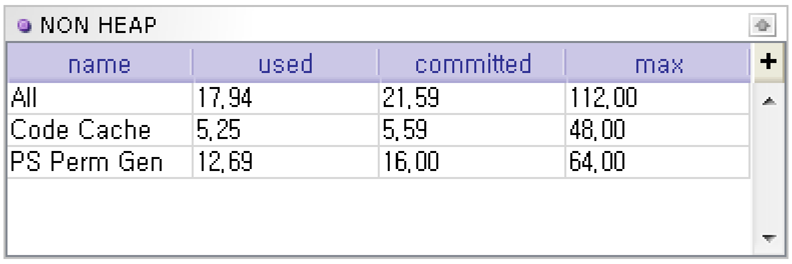
name	used	committed	max
All	53,17	62,69	246,56
PS Eden Space	2,24	4,63	25,81
PS Survivor S,,,	1,12	1,13	1,13
PS Old Gen	49,81	56,94	227,56

9.10.3.3. Non Heal Usage in Each Area

The changes in the NON-HEAP memory usage are monitored.

Figure 9-4:Non Heal Usage in Each Area



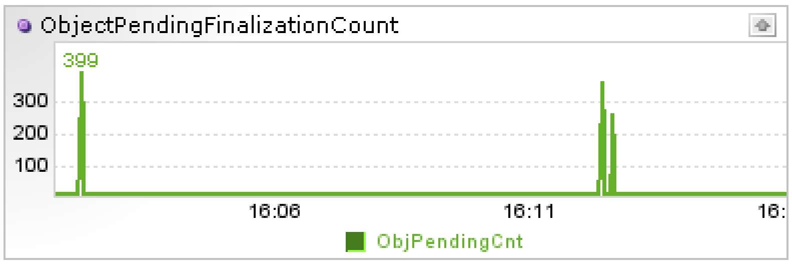

NON HEAP

name	used	committed	max
All	17,94	21,59	112,00
Code Cache	5,25	5,59	48,00
PS Perm Gen	12,69	16,00	64,00

9.10.3.4. Object Pending Finalization Count

If the number of object waiting for a call to the finalize() method. If this value is large, then it means that there can be many logics in the finalze() method. In this case, it requires some analysis.

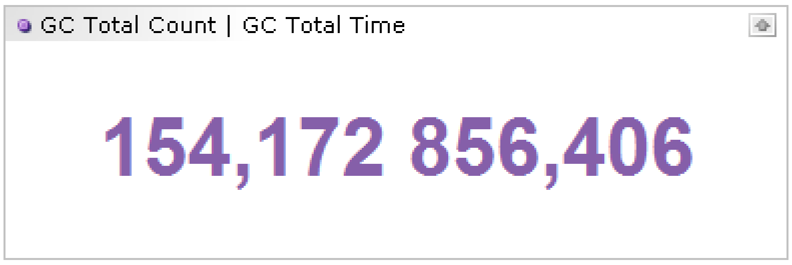
Figure 9-5:Object Pending Finalization Count



9.10.3.5. GC Total

This screen displays the information of the time used for and the total number of GC executed since the process was started.

Figure 9-6:GC Total



Other System Graph

The monitored agent's active service., system CPU, TPS and response time are included to draw comparisons on the agent status affected by the GC.

9.10.4.Data Inquiry

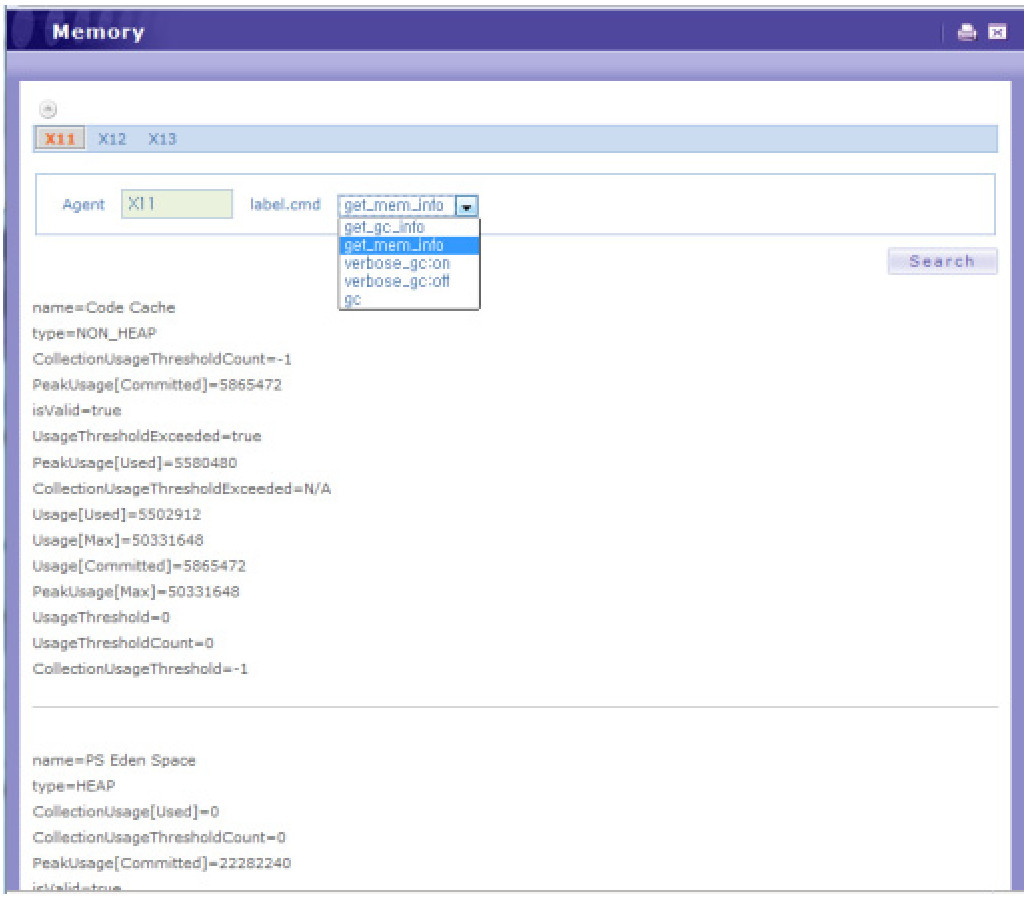
Detailed memory monitoring aims at monitoring and tuning the memory status in order to make the most optimal execution environment. Therefore, there is no need to store or manage them for a long time like other performance data. Therefore, Jennifer provides a function to inquire about the current process's memory status or to perform simple controls. Also, it provides a function to process the detailed memory data in a relatively short period of time by downloading it into an excel file.

9.10.4.1. Real Time Data Inquiry

Click the left button in the GC Delta(upper left) graph to inquire about the detailed memory status of the current agent. The following main functions can be used.

- Detailed memory status inquiry
- VERBOSE:GC on/off
- Forced GC

Figure 9-7:Real Time Data Inquiry



```
name=Code Cache
type=NON_HEAP
CollectionUsageThresholdCount=-1
PeakUsage[Committed]=5865472
isValid=true
UsageThresholdExceeded=true
PeakUsage[Used]=5580480
CollectionUsageThresholdExceeded=N/A
Usage[Used]=5502912
Usage[Max]=50331648
Usage[Committed]=5865472
PeakUsage[Max]=50331648
UsageThreshold=0
UsageThresholdCount=0
CollectionUsageThreshold=-1

name=PS Eden Space
type=HEAP
CollectionUsage[Used]=0
CollectionUsageThresholdCount=0
PeakUsage[Committed]=22282240
isValid=true
```

Notice: The meaning of each attribute for detailed memory inquiry should be found in the JVM manual.

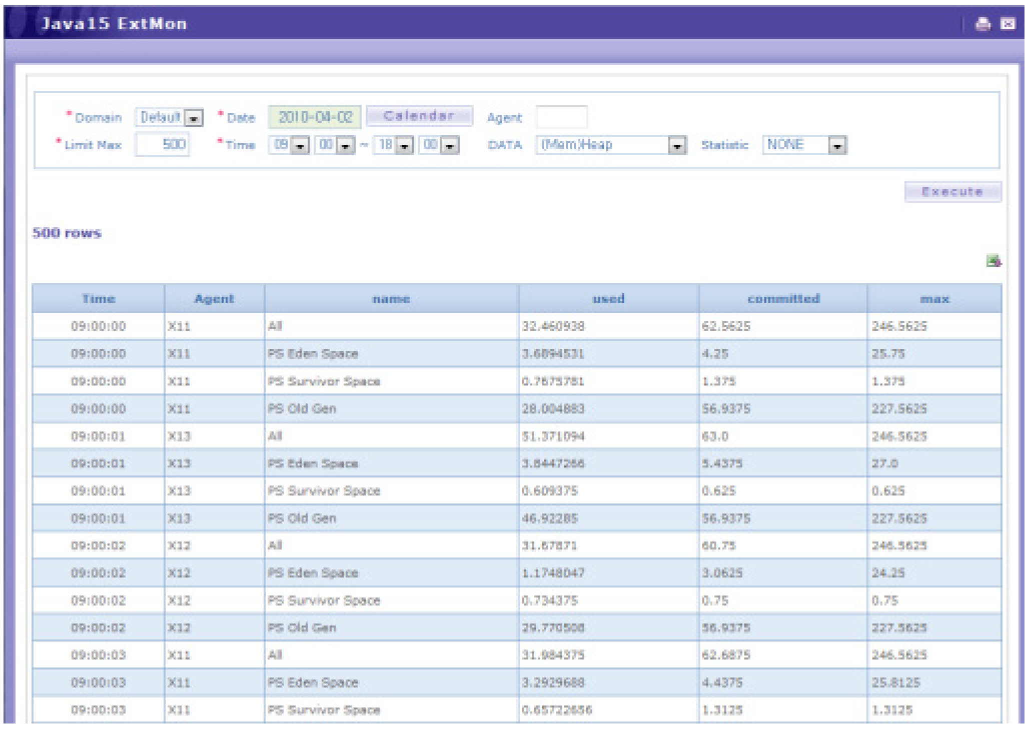
9.10.4.2. Stored Data Inquiry

When you set the memory monitoring function in your agent, you can save the important data in the Jennifer Server.

```
xmmon_file_save = true
```

Then, the detailed memory related information is saved as a REMON format. You can inquire about the data by clicking on the [Search] button in the lower right corner of the screen.

Figure 9-8:Stored Data Inquiry


Java15 ExtMon

*Domain Default *Date 2010-04-02 Calendar Agent
*Limit Max 500 *Time 09 00 ~ 18 00 DATA (Mem)Heap Statistic NONE

Execute

500 rows

Time	Agent	name	used	committed	max
09:00:00	X11	All	32.460938	62.5625	246.5625
09:00:00	X11	PS Eden Space	3.6894531	4.25	25.75
09:00:00	X11	PS Survivor Space	0.7675781	1.375	1.375
09:00:00	X11	PS Old Gen	28.004883	56.9375	227.5625
09:00:01	X13	All	51.371094	63.0	246.5625
09:00:01	X13	PS Eden Space	3.8447266	5.4375	27.0
09:00:01	X13	PS Survivor Space	0.609375	0.625	0.625
09:00:01	X13	PS Old Gen	46.92285	56.9375	227.5625
09:00:02	X12	All	31.67871	60.75	246.5625
09:00:02	X12	PS Eden Space	1.1748047	3.0625	24.25
09:00:02	X12	PS Survivor Space	0.734375	0.75	0.75
09:00:02	X12	PS Old Gen	29.770508	56.9375	227.5625
09:00:03	X11	All	31.984375	62.6875	246.5625
09:00:03	X11	PS Eden Space	3.2929688	4.4375	25.8125
09:00:03	X11	PS Survivor Space	0.65722656	1.3125	1.3125

13

Resources, DB Connection Monitoring for .NET System

This chapter describes how to monitor system resources such as CPU, memory utilization and DB.

10.1. CPU Monitoring

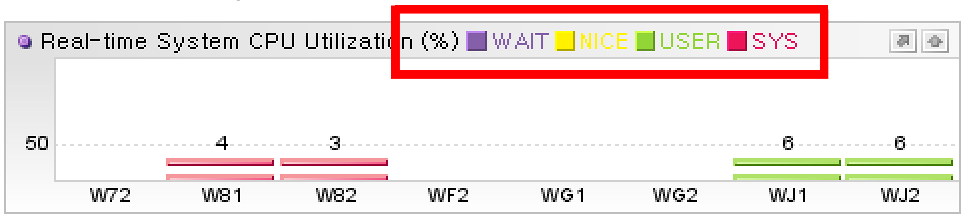
Using the JENNIFER agent, you can monitor the H/W system resources such as CPU utilization and the CPU utilization operated by the application and you can also monitor the CPU occupancy time by specific transaction. In addition, by using the WMOND module, you can monitor the arbitrary H/W CPU utilization per each CPU number, regardless of whether or not the JENNIFER agent is installed.

10.1.1.System CPU Utilization and Java Process CPU Utilization

The system CPU utilization rate is the total ratio of CPU usage by the system, while the process CPU utilization rate is the ratio of CPU usage by the application in which the JENNIFER agent is installed.

The system CPU utilization and the process CPU utilization are the general performance data collected by the JENNIFER agent. The average values can be viewed on equalizer and runtime line charts in real-time.

Figure 10-1:Real Time System CPU Utilization Chart



However, the CPU utilization is displayed in each region of CPU usage and an equalizer chart distinguishes the regions of CPU use with different colors. The following is a description of the regions of CPU usage.

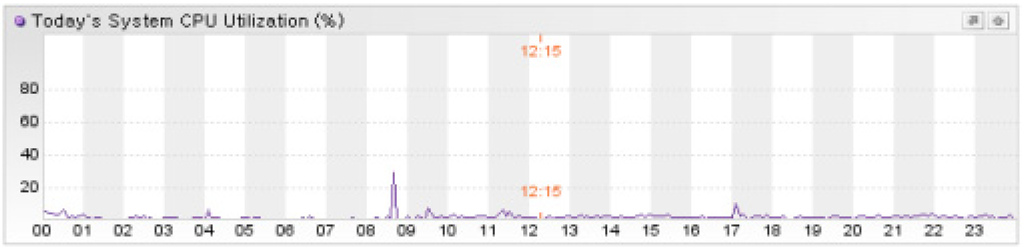
Table 10-1: Region of CPU Use

Regions of CPU use	Description
WAIT	(Not used in JENNIFER .NET)
NICE	(Not used in JENNIFER .NET)
USER	CPU utilization by the user (application)
SYS	CPU utilization by the system (kernel)

Notice: The equalizer chart distinguishes the regions of Java CPU utilization and individual CPU utilization collected by WMOND by using different colors.

In addition, the five-minute averages of system CPU utilization and process CPU utilization are stored in the SYS_CPU and JVM_CPU columns of the PERF_X_01~31 table. The CPU utilization trend for a certain date can be viewed on a line chart. .

Figure 10-2:Today's System CPU Utilization



10.1.2.CPU Occupancy Time by Transaction

The CPU occupancy time by a transaction is the amount of CPU use time while a transaction is executed. It can be collected from the application processing statistical data and the X-view profile data.

Using tpmC, you can objectively compare the CPU occupancy times by transaction for each H/W. tpmC represents the maximum number of the TPC-C benchmark scenario of TPC (Transaction Processing Performance Council, http://www.tpc.org) processed in one minute, and it is the most popular way to measure the H/W (mainly, CPU) performance.

Since CPU processing capacity varies greatly between computers, the CPU occupancy time by a transaction alone may not objectively illustrate the extent to which the transaction affects the performance. For example, even if the CPU occupancy time by transaction A processed by the H/W with a large CPU processing capacity is less than the CPU occupancy time by transaction B processed by the H/W with a small CPU processing capacity, transaction A may affect the performance more severely than transaction B. In other words, if transaction A is processed by H/W with a lower capacity, then it could occupy more CPU time than transaction B.

Therefore, by using tpmC, you can understand the manner in which the transaction affects the performance when the CPU processing capacity of the H/W is fixed.

To achieve this, use the approximate_tpmc_on_this_system option of the JENNIFER agent to set the tpmC value for the H/W in which the JENNIFER agent is installed.

```
approximate_tpmc_on_this_system = 30000
```

10.1.3.CPU Monitoring and Issuing Alerts

When the system CPU utilization and the process CPU utilization exceed the threshold, the JENNIFER server issues an alert. Please refer to the Alerts and Exception Monitoring for more detailed information.

10.2. Memory Monitoring

Using the JENNIFER agent, you can monitor the memory utilization of the H/W system operated by the application and the memory utilization of the process operated by the application. In addition, you can monitor the heap memory utilization by the application.

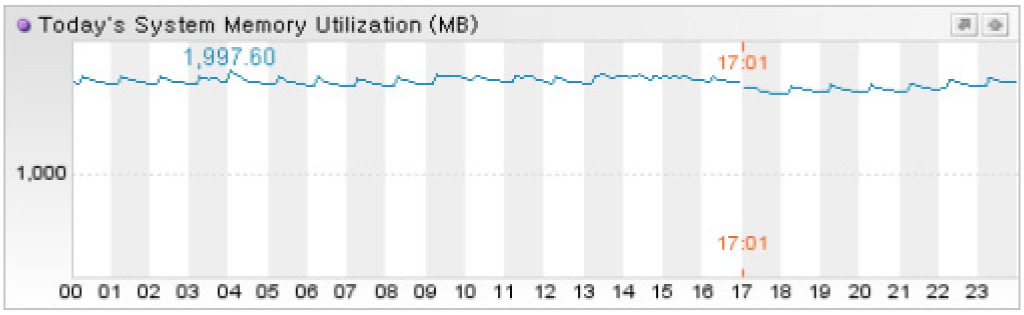
10.2.1.System Memory Utilization and Process Memory Utilization

The system memory utilization is the memory utilized by the system, while the process memory utilization is the memory utilization by the application in which the JENNIFER agent is installed. It is expressed in MB.

The system memory utilization and the process memory utilization is the general performance data collected by the JENNIFER agent. The thirty-second averages can be viewed on a runtime line chart in real-time.

In addition, the five-minute average of the system memory utilization and the process memory utilization is saved in the SYS_MEM_USED and JVM_NAT_MEM columns of the PERF_X_01~31 table. The memory utilization trend on a certain date can be viewed on a line chart.

Figure 10-3:Today's System Memory Utilization



Notice: Like the system CPU utilization, the system memory utilization and the Java process memory utilization require the correct installation of the native module.

10.2.2.CLR Heap Memory Utilization

The CLR heap memory utilization is obtained from the following code.

```
_jvmmem.used = GC.GetTotalMemory(false);
```

The value obtained from the above code is expressed in bytes, but JENNIFER converts it into Mbytes.

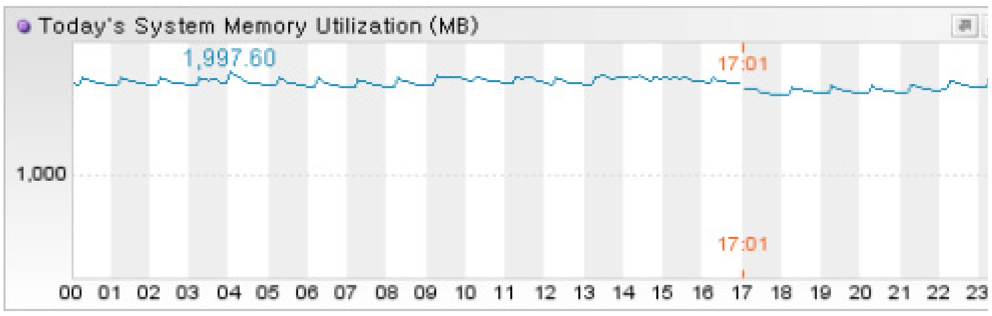
The CLR heap memory utilization is the general performance data collected by the JENNIFER agent. The thirty-second averages can be viewed on a runtime line chart in real-time.

In addition, the five-minute average of the CLR heap memory utilization is saved in the HEAP_TOTAL and HEAP_USED columns of the PERF_X_01~31 table. The memory utilization trend on a certain date can be viewed on a line chart.

Notice: The heap memory utilization is the ratio of the heap memory being used to the total heap memory. It is expressed as a %.

On a runtime line chart illustrating the heap memory utilization and the heap memory utilization rate, you can perform garbage collection on a specific JENNIFER agent.

Figure 10-4:Heap Memory Utilization



Notice: The contextual menu will appear only if the user's group has a gc privilege.

10.2.3.CLR Heap Memory Utilization

The CLR heap memory utilization is obtained from the following code.

```
_jvmmem.used = GC.GetTotalMemory(false);
```

The value obtained from the above code is expressed in bytes, but JENNIFER converts it into Mbytes.

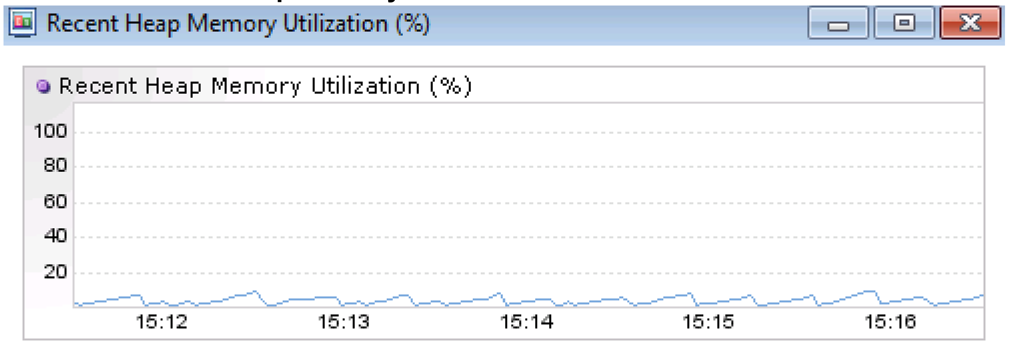
The CLR heap memory utilization is the general performance data collected by the JENNIFER agent. The thirty-second averages can be viewed on a runtime line chart in real-time.

In addition, the five-minute average of the CLR heap memory utilization is saved in the HEAP_TOTAL and HEAP_USED columns of the PERF_X_01~31 table. The memory utilization trend on a certain date can be viewed on a line chart.

Notice: The heap memory utilization is the ratio of the heap memory being used to the total heap memory. It is expressed as a %.

On a runtime line chart illustrating the heap memory utilization and the heap memory utilization rate, you can perform garbage collection on a specific JENNIFER agent.

Table 10-2: Recent Heap Memory Utilization


Recent Heap Memory Utilization (%)
Recent Heap Memory Utilization (%)
100
80
60
40
20
15:12
15:13
15:14
15:15
15:16

10.2.4.DB Connection Monitoring

Database interconnection greatly affects the enterprise application performance. For this reason, JENNIFER provides DB monitoring for performance analysis and troubleshooting. The following data is collected from DB monitoring.

- SQL and parameters executed by the application
- SQL response time and fetch counts
- DB connections
- Non-returned DB objects and exceptions

• Default Configuration for DB Monitoring

For DB monitoring, set the enable_db_sql_trace option of the JENNIFER agent to true. By default, it is set to true. If this option is set to false, no DB monitoring will be performed.

```
enable_db_sql_trace = true
```

• The number of DB Connection Monitoring

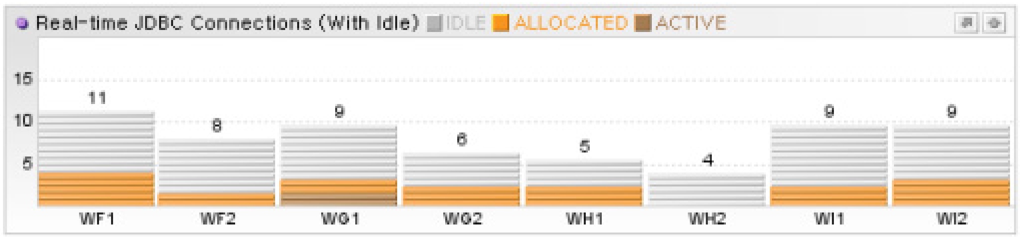
The available types of DB connections are as follows:

Table 10-3: DB Connection Type

Type	Description
Idle DB	The number of DB connections waiting in the DB connection pool

Type	Description
Allocated DB	The number of DB connections that are not currently performing SQL queries among all DB connections assigned to the application thread by the DB connection pool
Active DB	(N/A for JENNIFER .NET)

Figure 10-5:Real-Time DB Connections



In addition, the five-minute average of JDBC connections is saved in the JDBC_IDLE, JDBC_ALLOC and JDBC_ACTIVE columns of the PERF_X_01~31 tables.

10.2.5.SQL Data Collection Method

DB monitoring involves the transmission of an excessive amount of data compared to the monitoring of other types of resources. It has to collect all of the SQL and the parameters executed by the Java application, so that it can tune the problematic SQL.

Therefore, to minimize the amount of data transmitted, the JENNIFER agent uses the SQL hash values, not the SQL on the data sent to the JENNIFER server. It collects the SQL corresponding to the hash value and stores it in the SQLS table of the JENNIFER server.

To reduce the amount of SQL stored in the SQLS table, the JENNIFER agent processes the constants in the SQL as parameters and saves them. For example, the application processes the following SQL as additional SQL.

```
SELECT * FROM TABLE WHERE ID = ? AND AGE = 32 AND NAME = ‘KIM’
SELECT * FROM TABLE WHERE ID = ? AND AGE = 27 AND NAME = ‘LEE’
```

However, excluding the constants in the above code, the basic structure is the same. If the SQL is saved in the table, the data size would be too large. For example, if the following code is executed, 10,000 SQLs would need to be saved in the SQLS table.

```
IDbCommand command = ...;
IDataReader dr = null;
for (int i = 0; i < 10000; i++)
{
  command.CommandText = "SELECT * FROM USERS WHERE ID = " + I;
  dr = command.ExceuteReader();
...
}
```

To resolve this problem, the characters are converted to [$] and the numbers are converted to [#] before being saved in the SQLS table.

Therefore, only the following SQL is only saved in the SQLS table. .

```
SELECT * FROM TABLE WHERE ID = ? AND AGE = # AND NAME = '$'
```

In the X-view profile data, the constant parameters are expressed as 1 and the binding parameters are expressed as 2. The binding parameters refer to the parameters expressed as [?] in the SQL executed by the IDbCommand object.

10.2.6.SQL Exception Log Recording

When a SqlException occurs in the application, you can record it in the JENNIFER agent log file. To record it, set the enable_sql_error_trace option of the JENNIFER agent to true. By default, it is set to false:

```
enable_sql_error_trace = true
```

However, in a normal situation, a SqlException may occur, dependently on the business logic. Therefore, a SqlException must not be always interpreted as an error or failure. In addition, if you record to the log too often due to the frequent occurrence of SqlException that occurs too frequently, it may lead to performance degradation. Set this option to true only if necessary, and in normal situations, set it to false.

10.2.7.DB Monitoring and Exceptions

The following exceptions related to DB monitoring may occur.

• Delayed SQL Response Time

When the SQL response time exceeds the threshold, the WARNING_DB_BAD_RESPONSE exception will occur. The threshold can be set using the sql_bad_responsetime option of the JENNIFER agent. The default is 20000, and it is expressed in milliseconds.

```
sql_bad_responsetime = 20000
```

• Excessive Fetch Counts

To “Fetch” is to call the Read method of the IDataReader object. This refers to the number of hits for the Read method. If the fetch count for the same IDataReader object exceeds the threshold while executing a transaction, then the WARNING_DB_TOOMANY_FETCH exception will occur. The threshold can be set using the db_reader_warning_fetch_count option of the JENNIFER agent. The default is 10000.

Notice: db_reader_warning_fetch_count = 10000

• Non-returned DB Resource Tracking

The SqlConnection objects are called the DB resource. If the Close(or Dispose) method of the object is not called after the object is used, a non-returned DB resource exception will occur.

The problem is the question of when the non-returned DB resource must be detected. In case of web applications, non-returned DB resource which is requested at Application.EndRequest time is detected. On the other hand, non-returned DB resource is detected by each method level in case of general applications case.

The following exceptions are related to the non-returned DB resource.

WARNING_DB_CONN_UNCLOSED

If the Close(or Dispose) method is not called after the application uses the IDbConnection object, then the WARNING_DB_CONN_UNCLOSED will occur.

14 Alerts and Exception Monitoring

JENNIFER detects exceptions that occur in the Java application, and analyzes the status of the Java application to generate proper alerts. The alerts are classified into Critical, Error and Warning. The user can check the alerts and analyze the contents of previous alerts through the user interface in real time.

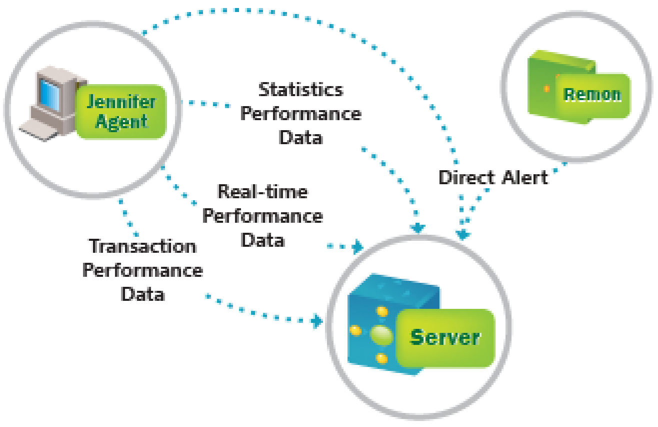
11.1. Understanding Alerts and Exceptions

An exception is generated by the JENNIFER agent when the java.lang.Throwable event occurs, or the response time is delayed while processing the transaction in the Java application in which the JENNIFER agent is installed. On the other hand, alerts are derived from the analysis of various performance data that the JENNIFER server collects from the JENNIFER agents.

Notice: One of the important types of performance data that forces the JENNIFER server to generate an alert is an exception occurring in the Java application. However, the JENNIFER server also generates alerts that are not related to exceptions.

To understand exceptions and alerts in JENNIFER, you must first understand how they are generated or created.

Figure 11-1:Alerts and Exceptions


Jennifer Agent
Statistics Performance Data
Remon
Real-time Performance Data
Direct Alert
Transaction Performance Data
Server

11.1.1.Application Exceptions and Alerts

Each alert is generated by the JENNIFER server, rather than by the JENNIFER agent. In otherwords, the JENNIFER server generates alerts that are based on the performance data, including the exceptions collected from the JENNIFER agent.

The JENNIFER agent transmits the transaction performance data corresponding to each transaction to the JENNIFER server. If an exception occurs while executing the transaction, an exception code will be included in the transaction performance data for transmission.

Since the transaction performance data is transmitted very frequently, only data that is small in size should be transmitted to the JENNIFER server. Therefore, the JENNIFER agent inserts only one exception code in the transaction performance data, regardless of the number of exceptions occurring in the transaction. Thus, even when more than one exception occurs in a transaction, the JENNIFER server issues only one alert.

However, the exception statistic data that the JENNIFER server collects at 10 minute-minute intervals includes every exception occurring in the transaction.

For instance, if it fails to make a DB connection while executing the transaction, the JENNIFER agent detects it as an ERROR_JDBC_CONNECTION_FAIL exception. However, if no exception processing is performed for the DB connection failure in the application code, then another exception will occur at the service start point, which the JENNIFER agent detects as an ERROR_UNCAUGHT_EXCEPTION. In this case, the JENNIFER agent will only include the ERROR_UNCAUGHT_EXCEPTION code in the transaction performance data for transmission to the JENNIFER server. However, the exception statistic data that the JENNIFER server collects at 10-minute intervalsincludes the ERROR_JDBC_CONNECTION_FAIL and ERROR_UNCAUGHT_EXCEPTION exceptions.

Therefore, the JENNIFER server detects the ERROR_UNCAUGHT_EXCEPTION in the transaction and issues the ERROR_UNCAUGHT_EXCEPTION alert. You can check the ERROR_JDBC_CONNECTION_FAIL exception and the ERROR_UNCAUGHT_EXCEPTION using the **[Real-time monitoring | Application]** menu.

Warning: Exception types and alerts types use the same codes.

11.1.2.Issuing Alerts for Multiple Exceptions in the Transaction

As described earlier, if several exceptions occur in a single transaction, only one of them will be sent to the JENNIFER server, and only one alert will be issued. The following criteria are used to select the exception that will be transmitted to the JENNIFER server. First, the exception that occurs the earliest will be sent to the JENNIFER server. For example, let's assume that the response is delayed while processing the external transaction and the WARNING_TX_BAD_RESPONSE exception is generated. Following this exception, the response to SQL processing is delayed, and the WARNING_DB_BAD_RESPONSE exception is generated as a result. In this case, only the WARNING_TX_BAD_RESPONSE exception will be transmitted to the JENNIFER server. However, if one of the exceptions listed below occurs at the end of transaction, it will be transmitted to the JENNIFER server, as it has a higher priority than other exceptions.

- ERROR_RECURRSIVE_CALL
- ERROR_UNCAUGHT_EXCEPTION
- ERROR_HTTP_IO_EXCEPTION
- ERROR_OUTOFMEMORY
- ERROR_UNKNOWN_ERROR
- ERROR_PLC_REJECTED

11.1.3.Real-Time Performance Data and Alerts

The JENNIFER agent transmits performance data such as the response time, the throughput and the number of concurrent users to the JENNIFER server at one-minute intervals. The JENNIFER server then generates various alerts based on this information. For instance, if the CPU utilization is high, the JENNIFER server will generate the WARNING_SYSTEM_CPU_HIGH alert.

11.1.4.User Defined Alerts

The JENNIFER agent or the REMON module can generate its own alerts and send them to the JENNIFER server. Alerts such as this could include: USER_DEFINED_FATAL, USER_DEFINED_ERROR, USER_DEFINED_WARNING, andUSER_DEFINED_MESSAGE.

Reference: It is possible to generate a user-defined alert using a JENNIFER agent adaptor like ExtraAgent, or the alertfilter of the REMON module. For mode details, refer to [ExtraAgent] and [Alerts].

In ExtraAgent, you can set the maximum number of alerts that can be buffered in one second through the max_num_of_direct_aler option. 10 is default. If a user-defined alert exceeding the setting for one second occurs in the agent, then the last ten such alerts will be sent to the JENNIFER server..

```
max_num_of_direct_alert = 10
```

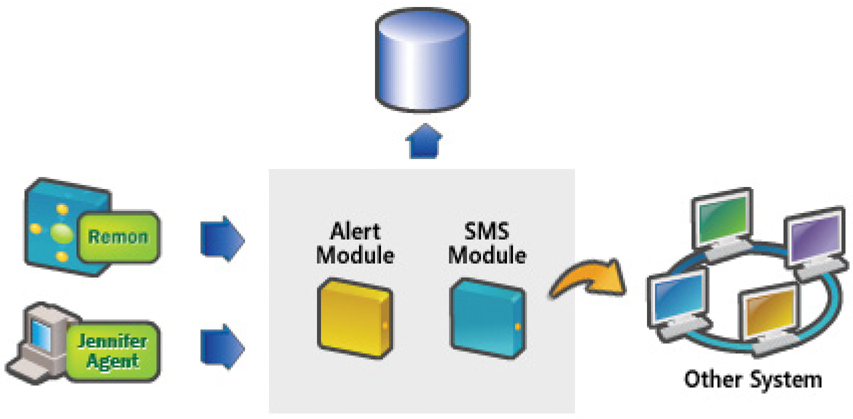
11.1.5.Alert Control

It is possible to block certain types of alerts. In other words, even if a certain exception occurs in the Java application, you can prevent the JENNIFER server from issuing an alert for it.

Warning: This is different from interconnecting the generated alerts data with other applications.

To do this, you should differentiate alerts generation from alert data interconnection with other applications.

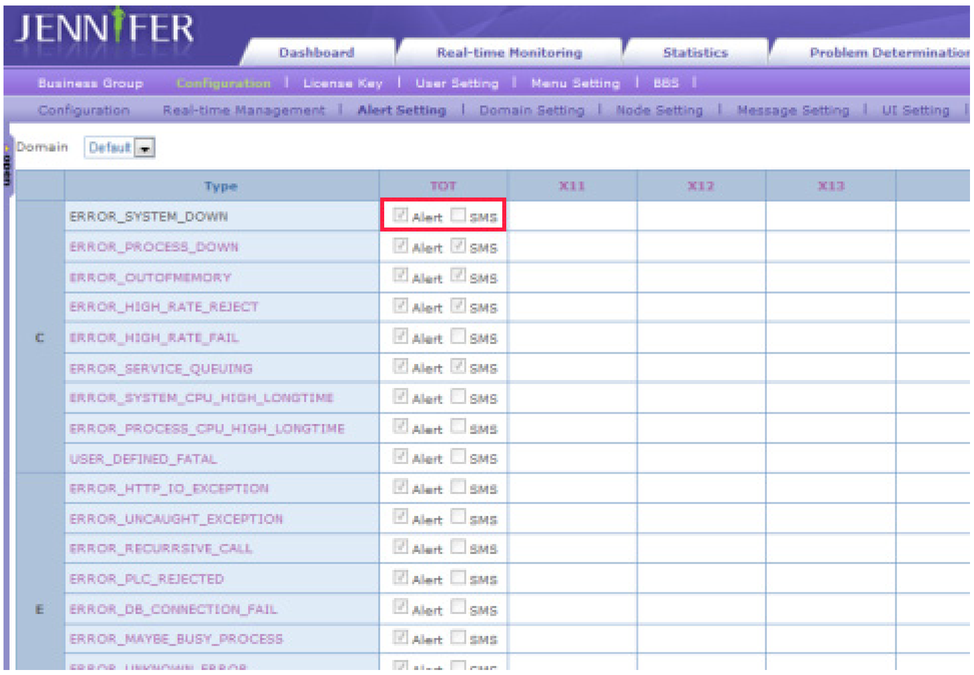
Figure 11-2:Alert Generation Flow



- Alert generation - The JENNIFER server generates an alert through the alert module.
- Alerts data interconnection - The JENNIFER server interconnects the alert data generated through the SMS module with other applications.

To control alarm issuance or alarm data interconnections, you need to use the [Configure | Alert] menu.

Figure 11-3:Alert Menu

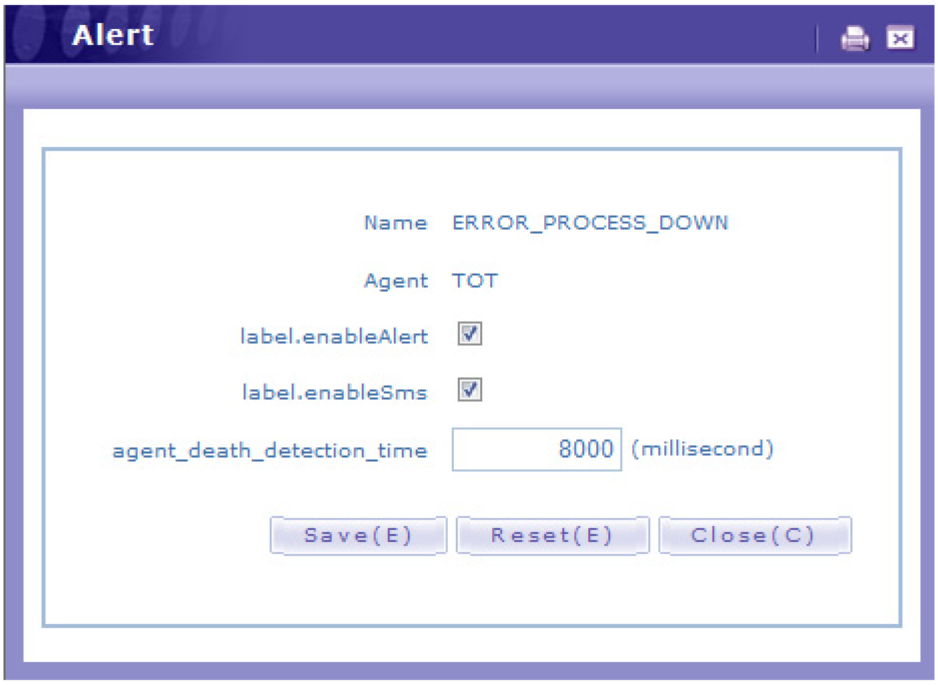


Domain Default

	Type	TOT	X11	X12	X13
C	ERROR_SYSTEM_DOWN	✓ Alert ☐ SMS			
	ERROR_PROCESS_DOWN	✓ Alert ✓ SMS			
	ERROR_OUTOFMEMORY	✓ Alert ✓ SMS			
	ERROR_HIGH_RATE_REJECT	✓ Alert ✓ SMS			
	ERROR_HIGH_RATE_FAIL	✓ Alert ☐ SMS			
	ERROR_SERVICE_QUEUING	✓ Alert ✓ SMS			
	ERROR_SYSTEM_CPU_HIGH_LONGTIME	✓ Alert ☐ SMS			
	ERROR_PROCESS_CPU_HIGH_LONGTIME	✓ Alert ☐ SMS			
	USER_DEFINED_FATAL	✓ Alert ☐ SMS			
E	ERROR_HTTP_IO_EXCEPTION	✓ Alert ☐ SMS			
	ERROR_UNCAUGHT_EXCEPTION	✓ Alert ☐ SMS			
	ERROR_RECURRSIVE_CALL	✓ Alert ☐ SMS			
	ERROR_PLC_REJECTED	✓ Alert ☐ SMS			
	ERROR_DB_CONNECTION_FAIL	✓ Alert ☐ SMS			
	ERROR_MAYBE_BUSY_PROCESS	✓ Alert ☐ SMS			
	ERROR_UNKNOWN_ERROR	✓ Alert ☐ SMS			

Click on this cell to control the alarm for each agent. In addition, you can set the main threshold values.

Figure 11-4:Control the Alarm



Alert

Name	ERROR_PROCESS_DOWN
Agent	TOT
label.enableAlert	✓
label.enableSms	✓
agent_death_detection_time	8000 (millisecond)

11.2. Alerts and Exception Types

JENNIFER classifies the alert levels as Critical, Error or Warning.

11.2.1. Critical

The following is a list of critical alerts.

• ERROR_SYSTEM_DOWN

When the WMOD process that monitors the CPU utilization is stopped, the JENNIFER server will generate this alert.

Warning: This alert is not related to the JENNIFER agent.

• ERROR_PROCESS_DOWN

When the Java application in which the JENNIFER agent is installed is stopped, then the JENNIFER server will generate this alert. If no data is received from the JENNIFER agent with a set period of time (default is 8 seconds), then the JENNIFER server attempts to establish a TCP connection to check the operation of the JENNIFER agent. If this TCP connection attempt fails, the JENNIFER server concludes that the Java application of the JENNIFER agent has been stopped.

Warning: If the Java application performs garbage collection for a long time when the Java application is not stopped, the JENNIFER agent will not be able to send the data to the JENNIFER server and a TCP connection attempt may fail as well. So, even if the Java application is operating, the JENNIFER server can issue the ERROR_PROCESS_DOWN alert.

The threshold can be set using the agent_death_detection_time option of the JENNIFER server. It is expressed in milliseconds.

```
agent_death_detection_time = 8000
```

• ERROR_OUTOFMEMORY

When the java.lang.OutOfMemoryError exception occurs in the Java application in which the JENNIFER agent is installed, the JENNIFER server will generate this alert.

• ERROR_HIGH_RATE_REJECT

If the ratio of the PLC service reject rate to the five second average service rate over a five-second period of the Java application in which the JENNIFER agent is installed exceeds the predefined threshold (default is 50%), this alert will be generated. You can set the threshold by using the high_rate_reject_alert_limit option of the JENNIFER server. It is expressed as a % and 50 is default.

```
high_rate_reject_alert_limit = 50
```

• ERROR_HIGH_RATE_FAIL

If the ratio of the failed rate to the five second average service rate over a five-second period of the Java application in which the JENNIFER agent is installed exceeds the pre-defined threshold (default is 50%), this alert will be generated. You can set the threshold by using the high_rate_failed_alert_limit option of the JENNIFER server. It is expressed as a % and 50 is default.

```
high_rate_failed_alert_limit = 50
```

• ERROR_SERVICE_QUEUING

If the average # of active services over a five-second period of the Java application in which the JENNIFER agent is installed exceeds the predefined threshold (default is 70), this alert will be generated. You can set the threshold by using the active_service_alert_limit option of the JENNIFER server and 70 is default.

```
active_service_alert_limit = 70
```

• ERROR_SYSTEM_CPU_HIGH_LONGTIME

If the system CPU utilization over a thirty-second period of the Java application in which the JENNIFER agent is installed exceeds the predefined threshold (default is 95%), this alert will be generated. You can set the threshold by using the sys_cpu_alert_limit option of the JENNIFER server. It is expressed as a % and 95 is default.

```
sys_cpu_alert_limit = 95
```

• ERROR_JVM_CPU_HIGH_LONGTIME

If the Java process CPU utilization over a thirty-second period of the Java application in which the JENNIFER agent is installed exceeds the predefined threshold (default is 90%), this alert will be generated. You can set the threshold by using the jvm_cpu_alert_limit option of the JENNIFER server. It is expressed as a % and 90 is default..

```
jvm_cpu_alert_limit = 90
```

• USER_DEFINED_FATAL

This is a user-defined alert that the user can generate through REMON.

11.2.2.Error

The following is a list of error alerts.

• ERROR_HTTP_IO_EXCEPTION

If the java.io.IOException occurs in the Java application in which the JENNIFER agent is installed, then the JENNIFER server will generate this alert. For instance, if the user closes the web browser after requesting the service but before receiving the result, this alert will be generated.

• ERROR_UNCAUGHT_EXCEPTION

If a java.lang.Exception that does not belong to other alert types occurs in the Java application in which the JENNIFER agent is installed, the JENNIFER server will generate this alert.

Notice: This alert is for exceptions sent to the application start point. If an exception occurs while processing the application and is handled by the catch, then the alert is not generated.

• ERROR_RECURSIVE_CALL

When the Java application in which the JENNIFER agent is installed calls the JSP or other sublets in excess of a certain threshold (default is 50,000), the JENNIFER agent will generate this alert. You can set the threshold by using the recurrsive_call_max_count option of the JENNIFER agent.

```
recurrsive_call_max_count = 50000
```

The fact that the sublet or JSP calls anther sublet or JSP implies that it calls the forward or include method of the javax.servelt.RequestDispatcher object in the Java code.

If you want to record the stacktrace information in the JENNIFER agent log file to investigate the causen of such alerts, you should set the recurrsive_call_trace option of the JENNIFER agent to “true”.

```
recurrsive_call_trace = true
```

Bb default, the number of stacktrace letters recorded in the JENNIFER agent log file is limited to 10,000. You can adjust this limit using the recurrsive_call_trace_size option of the JENNIFER agent.

```
recurrsive_call_trace_size = 10000
```

• ERROR_PLC_REJECTED

When the PLC rejects the request from the Java application in which the JENNIFER agent is installed, the JENNIFER server will generate this alert.

• ERROR_DB_CONNECTION_FAIL

When the java.sql.Connection object is not obtained from the Java application in which the JENNIFER agent is installed (database connection failure), the JENNIFER server will generate this alert.

• ERROR_MAYBE_BUSY_PROCESS

If no data is received from the JENNIFER agent within a predefined period of time (default is 8 seconds), then the JENNIFER server attempts to establish a TCP connection to check the operation of the JENNIFER agent. If this TCP connection attempt fails, then the JENNIFER server concludes that garbage collection is being performed in the Java application in which the JENNIFER agent is installed.

Warning: Garbage collection is only one of many possible reasons for a data transmission failures. You should not conclude that this alert inevitably means that there is a delay due to garbage collection.

You can set the threshold in the agent_death_detection_time option of the JENNIFER server. It is expressed in milliseconds and 8000 is default.

```
agent_death_detection_time = 8000
```

• ERROR_UNKNOWN_ERROR

If the java.lang.Error exception that does not belong to other alert types occurs in the Java application in which the JENNIFER agent is installed, the JENNIFER server will generate this alert

Notice: This alert is for exceptions sent to the application start point. If an exception occurs while processing the application and is handled by the catch, then the alert will not be generated.

• USER_DEFINED_ERROR

This is a user-defined alert with an error that the user can generate through REMON.

• ERROR_LOGICAL_PROCESS

If a logical error occurs in the ALSB(AquaLogic Service Bus) or the WLI(WebLogic Integration), the JENNIFER server will generate this alert.

11.2.3.Warning

The following is a list of warning alerts.

• WARNING_DB_CONN_UNCLOSED

When the Java application in which the JENNIFER agent is installed does not call the close method after using the java.sql.Connection object, the JENNIFER server will generate this alert.

To resolve the problem, the JENNIFER agent collect the basic stacktrace. If you want to collect the more detailed stack trace, then you should set the enable_jdbc_connection_fullstack_trace option of the JENNIFER agent to “true”. However, since this can cause additional load to the Java application, you are recommended to use this method for the purpose of analysis only.

```
enable_jdbc_connection_fullstack_trace = true
```

• WARNING_JDBC_STMT_UNCLOSED

When the Java application in which the JENNIFER agent is installed does not call the close method after using the java.sql.Statement object, the JENNIFER server will generate this alert.

To resolve the problem, the JENNIFER agent collect the basic stacktrace. If you want to collect the more detailed stack trace, then you should set the enable_jdbc_statement_fullstack_trace option of the JENNIFER agent to “true”. However, since this can cause additional load to the Java application, you are recommended to use this method for the purpose of analysis only.

```
enable_jdbc_statement_fullstack_trace = true
```

• WARNING_JDBC_PSTMT_UNCLOSED

When the Java application in which the JENNIFER agent is installed does not call the close method after using the java.sql.PreparedStatement object, the JENNIFER server will generate this alert.

To resolve the problem, the JENNIFER agent collect the basic stacktrace. If you want to collect the more detailed stack trace, then you should set the enable_jdbc_preparedstatement_fullstack_trace option of the JENNIFER agent to “true”. However, since this can cause additional load to the Java application, you are recommended to use this method for the purpose of analysis only.

```
enable_jdbc_preparedstatement_fullstack_trace = true
```

• WARNING_JDBC_CSTMT_UNCLOSED

When the Java application in which the JENNIFER agent is installed does not call the close method after using the java.sql.CallableStatement object, the JENNIFER server will generate this alert.

To resolve the problem, the JENNIFER agent collect the basic stacktrace. If you want to collect the more detailed stack trace, then you should set the enable_jdbc_callablestatement_fullstack_trace option of the JENNIFER agent to “true”. However, since this can cause additional load to the Java application, you are recommended to use this method for the purpose of analysis only.

```
enable_jdbc_callablestatement_fullstack_trace = true
```

• WARNING_JDBC_RS_UNCLOSED

When the Java application in which the JENNIFER agent is installed does not call the close method after using the java.sql.ResultSet object, the JENNIFER server will generate this alert.

To resolve the problem, the JENNIFER agent collect the basic stacktrace. If you want to collect the more detailed stack trace, then you should set the enable_jdbc_resultset_fullstack_trace option of the JENNIFER agent to “true”. However, since this can cause additional load to the Java application, you are recommended to use this method for the purpose of analysis only.

```
enable_jdbc_resultset_fullstack_trace = true
```

Notice: Non-returned JDBC resource alerts

The objects like java.sql.Connection, java.sql.Statement and java.sql.ResultSet are called JDBC resources. If the close method is not invoked after using one of these resources, the JENNIFER server generates the non-returned JDBC resource alert.

However the problem lies in the question of where to detect the non-returned JDBC resource. It is desirable to detect the non-returned JDBC resource at the end of the application transaction. However, if the JDBC resource is used and returned in the background thread, then it is likely to generate an incorrect alert. For this reason, to detect the non-returned JDBC resource, JENNIFER checks the execution of the close method when garbage collection is performed on the JDBC resource. Therefore, if no garbage collection is done, then the non-returned JDBC alert might not be issued. Even if garbage collection is done, there is some time elasped between the application transaction and the non-returned JDBC resource alert. In other words, the alert can be delayed.

• WARNING_DB_TOOMANY_FETCH

If the Java application in which the JENNIFER agent is installed performs fetching on the same java.sql.ResultSet object in exceed of the predetermined threshold (default is 5000), then the JENNIFER server will generate this alert.

Fetching implies that the next method for the java.sql.ResultSet object is invoked.

You can set the threshold using the jdbc_resultset_warning_fetch_count option of the JENNIFER agent and 5000 is default.

```
jdbc_resultset_warning_fetch_count = 5000
```

• WARNING_JDBC_STMT_EXCEPTION

If the java.sql.SQLException exception occurs when the Java application in which the JENNIFER agent is installed performs queries using the execute, executeBatch, executeQuery, and executeUpdate methods of the java.sql.Statement object, the JENNIFER server will generate this alert.

• WARNING_DB_BAD_RESPONSE

If the JDBC query time for the Java application in which the JENNIFER agent is installed exceeds the threshold (default is 20 seconds), the JENNIFER server will generate this alert. You can set the threshold using the sql_bad_responsetime option of the JENNIFER agent. It is expressed in milliseconds and 20000 is default.

```
sql_bad_responsetime = 20000
```

• WARNING_TX_CALL_EXCEPTION

If an exception occurs while executing the external transaction at the Java application in which the JENNIFER agent is installed, then the JENNIFER server will generate this alert.

• WARNING_TX_BAD_RESPONSE

If time out(10 second) occurs while executing the external transaction at the Java application in which the JENNIFER agent is installed, then the JENNIFER server will generate this alert.You can set the threshold using the tx_bad_responsetime option of the JENNIFER agent. It is expressed in milliseconds and 10000 is default

```
tx_bad_responsetime = 10000
```

• WARNING_APP_BAD_RESPONSE

If time out(30 second) occurs while executing the external transaction at the Java application in which the JENNIFER agent is installed, then the JENNIFER server will generate this alert.You can set the threshold using the app_bad_responsetime option of the JENNIFER agent. It is expressed in milliseconds and 10000 is default.

```
app_bad_responsetime = 10000
```

• WARNING_DB_UN_COMMIT_ROLLBACK

If the JDBC transaction is not terminated properly (by calling the rollback or commit methods of the java.sql.Connection object) after it has been started (by calling the setAutoCommit method of thejava.sql.Connection object while setting the parameter to false) at the Java application in which the JENNIFER agent is installed and the execute, executeBatch and executeUpdate methods of the java.sql.Statement object are called, the JENNIFER server will generate this alert.

However, if only the executeQuery method of the java.sqk.Statement object is called after starting the JDBC transaction, you do not need to terminate the JDBC transaction to avoid this alert.

However, if SELECT is executed using the execute method of the java.sql.Statement object, this alert will not describe the situation accurately. Therefore, you can avoid issuing this alert by using the ignore_rollback_uncommited_error option of the JENNIFER agent.

```
ignore_rollback_uncommited_error = true
```

For a long-term use, it is recommended to modify the source code such that SELECT is executed by the exeucteQuery method.

• WARNING_DUMMY_HTTPSESSION_CREATED

If the page attribute session is not set for the Java application in which the JENNIFER agent is installed or the javax.servlet.http.HttpSession is generated while executing the JSP set to true.

```
<%@ page session="false" %>
```

The JENNIFER server will generate this alert. In most environments, this alert is not suitable and it is not usually issued. To generate this alert, you should set the enable_dummy_httpsession_trace option of the JENNIFER agent to true. Default value is 'false'.

```
enable_dummy_httpsession_trace = true
```

• WARNING_DB_CONN_ILLEGAL_ACCESS

If two or more threads use the same java.sql.Connection object simultaneously in the Java application in which the JENNIFER agent is installed, the JENNIFER server will generate this alert.

• WARNING_ORACLE_XMLTYPE_UNCLOSED

If the close method of the oracle.xdc.XMLType object is not invoked when the Oracle database is used by the Java application in which the JENNIFER agent is installed, the JENNIFER server will generate this alert.

• USER_DEFINED_WARNING

This is a user-defined alert, which gives a warning that the user can generate through REMON.

• WARNING_CUSTOM_RUNTIME_EXCEPTION

This alert is issued when processing the java.lang.RuntimeException in the logic by using the Custom Trace adaptor.

• WARNING_CUSTOM_EXCEPTION

This alert is issued when processing the java.lang.Exception in the logic by using the Custom Trace adaptor.

• WARNING_CUSTOM_THROWABLE

This alert is issued when processing the java.lang.Throwable in the logic by using the Custom Trace adaptor.

• WARNING_RESOURCE_LEAK

The JENNIFER server will generate this alert when a user-defined resource leakage occurs.

• WARNING_JVM_HEAP_MEM_HIGH

If the average Java heap memory utilization by the system operating the Java application in which the JENNIFER agent is installed exceeds the threshold for a certain time period, the JENNIFER server will generate this alert. You can set the threshold in the jvm_heap_warning_limit option of the JENNIFER server. It is expressed as a % and 95 is default.

```
jvm_heap_warning_limit = 95
```

You can set the time period by using the jvm_heap_check_time option of the JENNIFER server. It is expressed in seconds and 300 is default.

```
jvm_heap_check_time = 300
```

In addition, you can set the time gap of alert generation by using the jvm_heap_warning_interval option of the JENNIFER server. It is expressed in seconds and 3600 is default.

```
jvm_heap_warning_interval = 3600
```

• WARNING_SYSTEM_CPU_HIGH

If the average system CPU utilization by the system operating the Java application in which the JENNIFER agent is installed exceeds the threshold over a thirty-second period, the JENNIFER server will generate this alert. You can set the threshold in the sys_cpu_warning_limit option of the JENNIFER server. It is expressed as a % and 80 is default.

```
sys_cpu_warning_limit = 80
```

The ERROR_SYSTEM_CPU_HIGH_LONGTIME alert can override this time period.

• WARNING_PROCESS_CPU_HIGH

If the average Java process CPU utilization by the system operating the Java application in which the JENNIFER agent is installed exceeds the threshold over a thirty-second period, the JENNIFER server will generate this alert. You can set the threshold in the jvm_cpu_warning_limitoption of the JENNIFER server. It is expressed as a % and 70 is default. .

```
jvm_cpu_warning_limit = 70
```

The ERROR_SYSTEM_CPU_HIGH_LONGTIME alert can override this time period.

11.2.4.Messages

• SYSTEM_MESSAGE

This alert is used by the JENNIFER server to notify the operation of the JENNIFER agent.

• USER_DEFINED_MESSAGE

This message corresponds to the user-defined alert type used by the JENNIFER agent or REMON.

11.3. Saving Alerts and Exception Data

The alerts and exception data are saved in the performance database. Alerts are saved in table ALERT_01 through 31, while exceptions are saved in table ERRORS_10M_01 through 31. However, different saving methods are used for alerts and exceptions.

Notice: All exceptions are included in alerts. Therefore, exceptions are also saved in table, ALERT_01~31.

First, the ALERT_01~31 tables contain all of the alerts. If one ERROR_SERVICE_QUEUING alert and two WARNING_JDBC_STMT_EXCEPTION alerts occur, then a total of three alerts are saved in the ALERT_01~31 tables.

However, the same exception will be saved only once in the ERRORS_10M_01~31 tables, while CNT column shows the number of occurrences. Therefore, if the WARNING_JDBC_STMT_EXCEPTION in the above example occurs two times in the same 10-minute segment, it will be saved only once in the ERRORS_10M_01~31 tables, while the CNT columns will show ‘two’ as the number of occurrences.

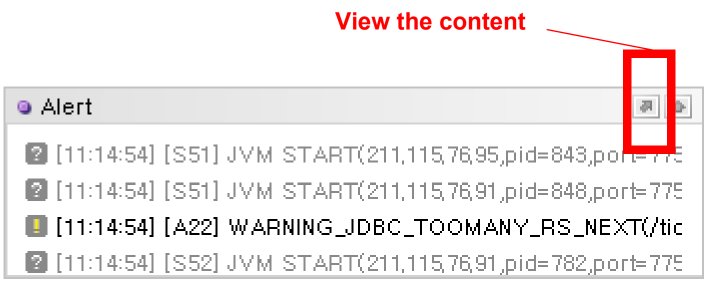
11.4. Alerts and Exception Inquiry

In this section, we will describe the method for detecting and analyzing exceptions and alerts through the JENNIFER user interface.

11.4.1.Alert Charts and Board Area

The alert chart at the bottom of the JENNIFER dashboard displays the content of an alert in real time.

Figure 11-5:Alert Chart



```
[11:14:54] [S51] JVM START(211,115,76,95,pid=843,port=77
[11:14:54] [S51] JVM START(211,115,76,91,pid=848,port=775
[11:14:54] [A22] WARNING_JDBC_TOOMANY_RS_NEXT(/tic
[11:14:54] [S52] JVM START(211,115,76,91,pid=782,port=775
```

The alert chart only shows a limited number of alerts. To look at alerts in more detail, you should click the [View the Content] button in the alert chart to launch the pop-up window.

If you want to delete the alert that is visible in the alert chart, proceed as follows.

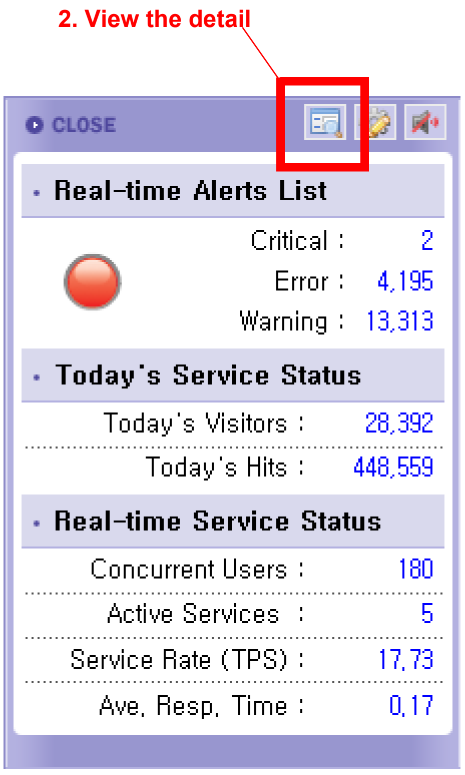
- Move the cursor into the alert chart, and right-click.

- From the contextual menu, select [Delete].

Notice: Even if you delete the alert from the alert chart, it will still be stored in the performance database.

When an alert occurs, the board area on the right that had been hidden previously will appear. The user can check the number of critical, error and warning alerts by using the real-time alert content in the board area. However, it is not possible to check the details of alerts from the board area. To view further details regarding an alert, the user should click the [View the Detail] button to launch the pop-up window.

Figure 11-6:Board Area



Critical :	2
Error :	4,195
Warning :	13,313
Today's Visitors :	28,392
Today's Hits :	448,559
Concurrent Users :	180
Active Services :	5
Service Rate (TPS) :	17.73
Ave. Resp. Time :	0.17

11.4.2.Alert Pop-up Window

The alert pop-up window consists of three tabs.

Table 11-1: Alert Tabs

Tabs	Description
Today's alert list	Displays all alerts that have occurred today.

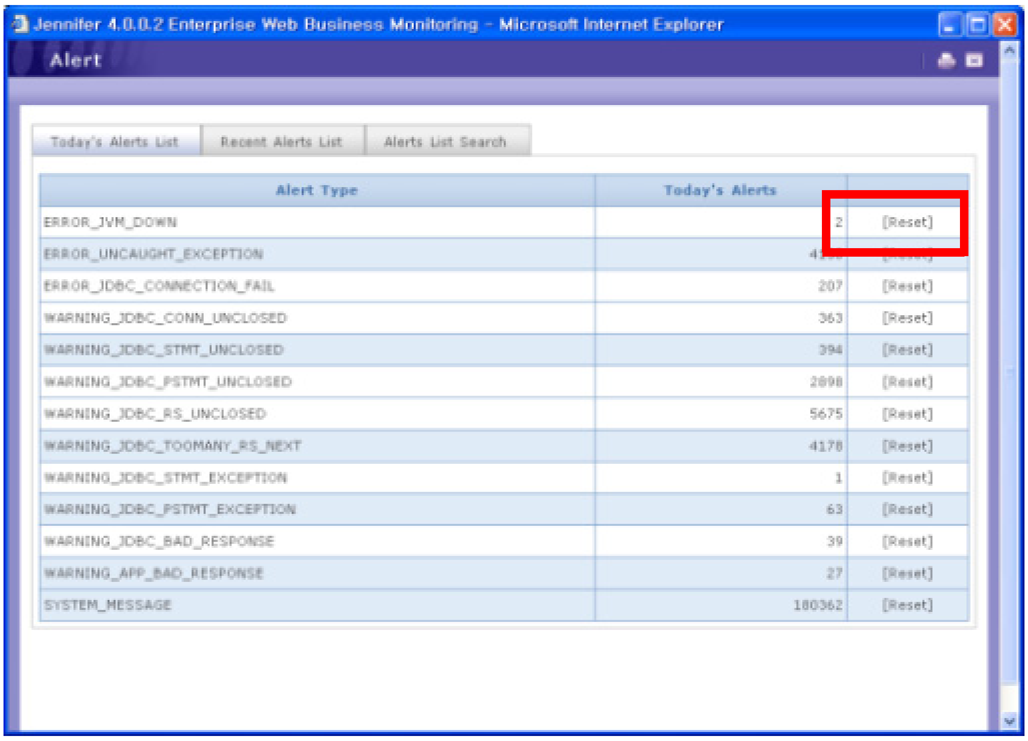
Table 11-1: Alert Tabs

Tabs	Description
Recent alert list	Displays the most recent alerts. Regardless of the time of occurrent, it shows up to 150 alerts of various types - critical, error, warning.
Alert list search	You can browse through alerts that occurred in the past.

11.4.2.1. Today's Alert List

It displays the alerts of various types that have occurred today. Using this tab, you can check the number of alerts of each type.

Figure 11-7:Today's Alert List


Jennifer 4.0.0.2 Enterprise Web Business Monitoring - Microsoft Internet Explorer
Alert
Today's Alerts List
Recent Alerts List
Alerts List Search

Alert Type	Today's Alerts	
ERROR_JVM_DOWN	2	[Reset]
ERROR_UNCAUGHT_EXCEPTION	4[illegible]	[illegible]
ERROR_JDBC_CONNECTION_FAIL	207	[Reset]
WARNING_JDBC_CONN_UNCLOSED	363	[Reset]
WARNING_JDBC_STMT_UNCLOSED	394	[Reset]
WARNING_JDBC_PSTMT_UNCLOSED	2898	[Reset]
WARNING_JDBC_RS_UNCLOSED	5675	[Reset]
WARNING_JDBC_TOOMANY_RS_NEXT	4178	[Reset]
WARNING_JDBC_STMT_EXCEPTION	1	[Reset]
WARNING_JDBC_PSTMT_EXCEPTION	63	[Reset]
WARNING_JDBC_BAD_RESPONSE	39	[Reset]
WARNING_APP_BAD_RESPONSE	27	[Reset]
SYSTEM_MESSAGE	180362	[Reset]

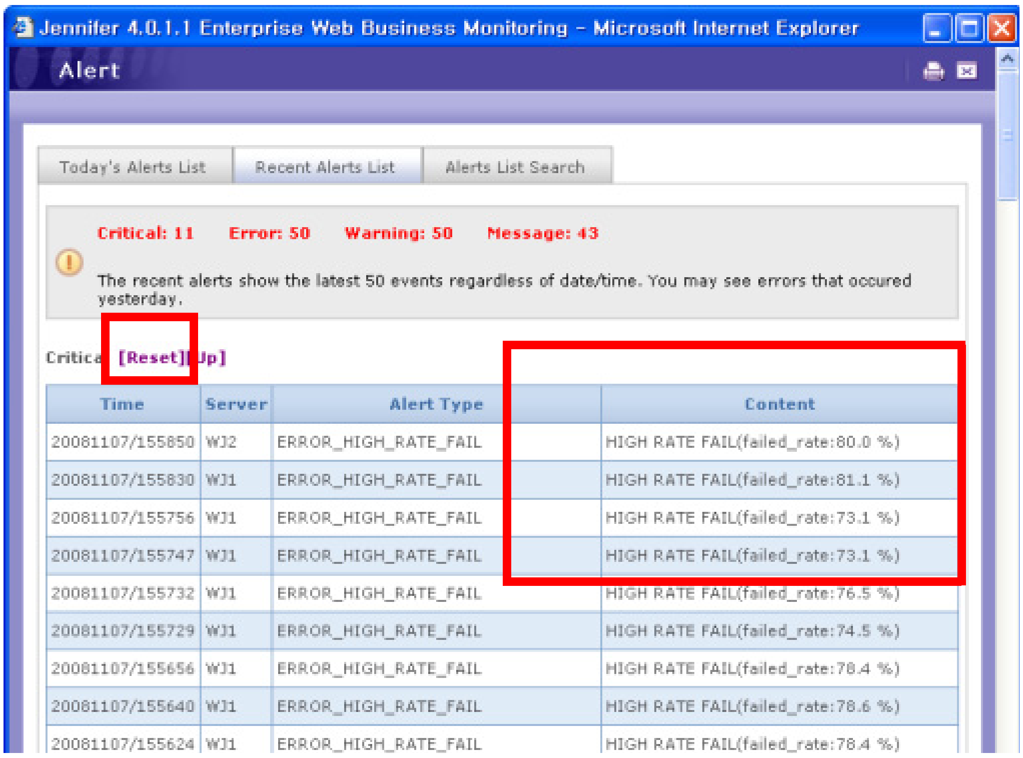
- Reset link - If you click the [Reset] link for a certain type of alert, then the number of occurrences will return to zero, and it will be deleted from Today's alert list.

Notice: Even if you click the reset link, the actual alert information will still be stored in the performance database. In addition, if you click the reset link, the real-time alert list in the board area is also initialized. Therefore, you may see the different data in the alert list searching tap and in the dashboard.

11.4.2.2. Recent Alert List

This list shows up to 150 critical, error, warning and message alerts. While Today's alert list shows the number of occurrences of a certain alert on the day, the recent alert list shows the details of each alert.

Figure 11-8:Recent Alert List


Jennifer 4.0.1.1 Enterprise Web Business Monitoring - Microsoft Internet Explorer
Alert
Today's Alerts List
Recent Alerts List
Alerts List Search

Critical: 11 Error: 50 Warning: 50 Message: 43

The recent alerts show the latest 50 events regardless of date/time. You may see errors that occured yesterday.

Critica [Reset][Up]

Time	Server	Alert Type	Content
20081107/155850	WJ2	ERROR_HIGH_RATE_FAIL	HIGH RATE FAIL(failed_rate:80.0 %)
20081107/155830	WJ1	ERROR_HIGH_RATE_FAIL	HIGH RATE FAIL(failed_rate:81.1 %)
20081107/155756	WJ1	ERROR_HIGH_RATE_FAIL	HIGH RATE FAIL(failed_rate:73.1 %)
20081107/155747	WJ1	ERROR_HIGH_RATE_FAIL	HIGH RATE FAIL(failed_rate:73.1 %)
20081107/155732	WJ1	ERROR_HIGH_RATE_FAIL	HIGH RATE FAIL(failed_rate:76.5 %)
20081107/155729	WJ1	ERROR_HIGH_RATE_FAIL	HIGH RATE FAIL(failed_rate:74.5 %)
20081107/155656	WJ1	ERROR_HIGH_RATE_FAIL	HIGH RATE FAIL(failed_rate:78.4 %)
20081107/155640	WJ1	ERROR_HIGH_RATE_FAIL	HIGH RATE FAIL(failed_rate:78.6 %)
20081107/155624	WJ1	ERROR_HIGH_RATE_FAIL	HIGH RATE FAIL(failed_rate:78.4 %)

- Reset link - If you click the [Reset] link, the alert that is currently being viewed among critical, error, warning and message alerts will be deleted from the list.

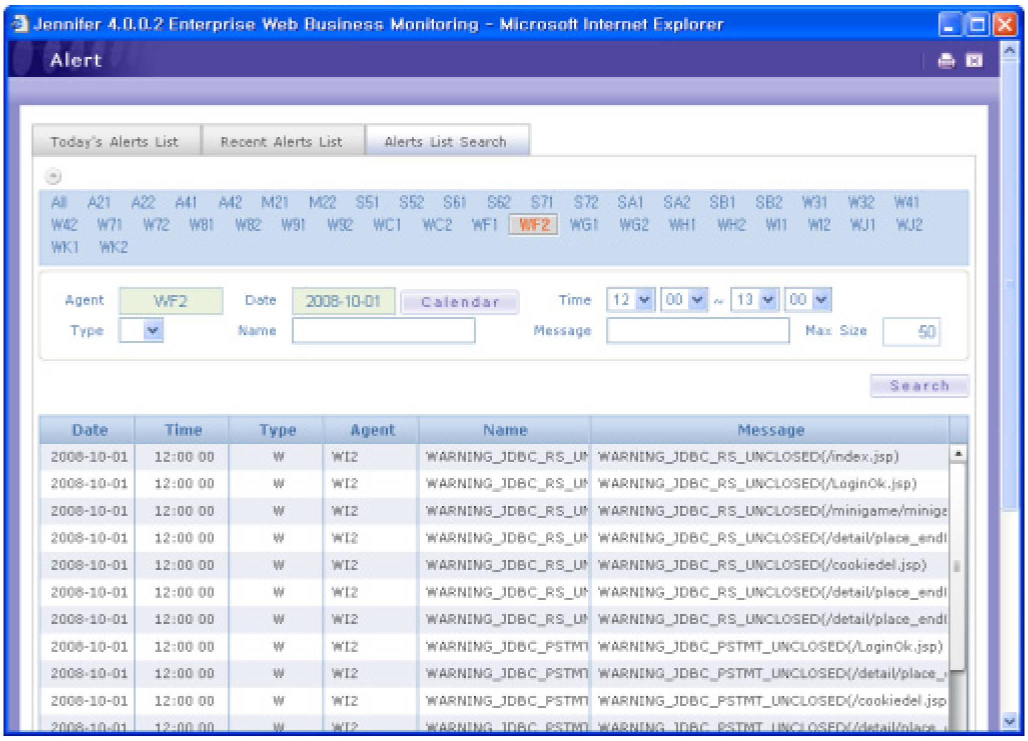
Notice: Even if you click the reset link, the actual alert information will still be stored in the performance database.

- Content- It shows the application name or related data that the alert corresponds to.

11.4.2.3. Alert List Search

You can check the details of previous alerts, which are defines as alerts that occurred 'prior to the current 10-minute segment'.

Figure 11-9:Alert List Search


Jennifer 4.0.0.2 Enterprise Web Business Monitoring - Microsoft Internet Explorer
Alert
Today's Alerts List
Recent Alerts List
Alerts List Search
All A21 A22 A41 A42 M21 M22 S51 S52 S61 S62 S71 S72 SA1 SA2 SB1 SB2 W31 W32 W41
W42 W71 W72 W81 W82 W91 W92 WC1 WC2 WF1 WF2 WG1 WG2 WH1 WH2 WI1 WI2 WJ1 WJ2
WK1 WK2
Calendar
Search

Agent: WF2 Date: 2008-10-01 Time: 12 : 00 ~ 13 : 00
Type: Name: Message: Max Size: 50

Date	Time	Type	Agent	Name	Message
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_RS_UN	WARNING_JDBC_RS_UNCLOSED(/index.jsp)
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_RS_UN	WARNING_JDBC_RS_UNCLOSED(/LoginOk.jsp)
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_RS_UN	WARNING_JDBC_RS_UNCLOSED(/minigame/minigz
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_RS_UN	WARNING_JDBC_RS_UNCLOSED(/detail/place_endi
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_RS_UN	WARNING_JDBC_RS_UNCLOSED(/cookiedel.jsp)
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_RS_UN	WARNING_JDBC_RS_UNCLOSED(/detail/place_endi
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_RS_UN	WARNING_JDBC_RS_UNCLOSED(/detail/place_endi
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_PSTMT	WARNING_JDBC_PSTMT_UNCLOSED(/LoginOk.jsp)
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_PSTMT	WARNING_JDBC_PSTMT_UNCLOSED(/detail/place_
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_PSTMT	WARNING_JDBC_PSTMT_UNCLOSED(/cookiedel.jsp
2008-10-01	12:00:00	W	WI2	WARNING_JDBC_PSTMT	WARNING_JDBC_PSTMT_UNCLOSED(/detail/place_

The search conditions are as follows.

Table 11-2: Search Conditions

Search condition	Description
Agent	Select a certain agent from the JENNIFER agent list at the top. If you want to perform searching on all JENNIFER agents, then select [All].
Date	Enter the date of the alert.
Time	Enter the time of the alert in hours and minutes.
Type	Select the type - Critical, Error, or Warning.
Name	Enter the name of the alert.
Message	Enter the alert message that you wish to search for.
Maximum size	Enter the maximum number of search results that you wish to receive. The default is 50.

11.4.3.Real-Time Application List

Using the **[Real-time monitoring | Application]** menu, you can check the exceptions that have occurred in the most recent 10-minute segment..

Notice: This list does not display alerts. It displays the exceptions occurring while processing the application transaction.

11.4.4.Statistics Application List

Using the **[Statistics | Application]** menu, you can check previous exceptions, which means exceptions that occurred 'prior to the current 10 minute segment'..

Notice: This list does not display alerts. It displays the exceptions occurring while processing the application transaction.

11.4.5.X-View Chart

The application transactions with exceptions are displayed in red on the X-View chart.

Notice: Exceptions related to a non-returned JDBC resource are not displayed in red on the X-View chart.

11.5. Alert Data Interconnection

The alerts that the JENNIFER server generates can be sent to other applications through a SMS adaptor.

11.5.1.Basic Setting for Data Interconnection

The SMS adaptor is responsible for alert data interconnections. The class to implement the SMS adaptor can be set using the sms_adapter_class_name option of the JENNIFER server.

```
sms_adapter_class_name = com.javaservice.jennifer.server.SMSExample
```

If you want to use two or more SMS adaptors, you should separate them with semicolons [;]. This can be set using the sms_adapter_class_name option of the JENNIFER server.

```
sms_adapter_class_name =
com.javaservice.jennifer.server.SMSExample;com.javaservice.jennifer.se
rver.AlertToLogFile
```

For the development method used for the SMS adaptors, please refer to [API for SMS Adaptors].

If too many alerts are issued by the JENNIFER server within a short period of time, the SMS adaptor will need to send too many alerts to other applications. To prevent the duplicate transmission of alerts, you should adjust the SMS transmission internal using the sms_alert_minimal_interval option of the JENNIFER server. It is expressed in milli-seconds and 60000 is default.

```
sms_alert_minimal_interval = 60000
```

You can also allow only a certain type of alert to be transmitted by the SMS adaptor to other applications. Use the alert type to transmit by the SMS adaptor as the name, and set it to true in the JENNIFER server option.

For instance, if you want the ERROR_PLC_REJECTED alert to be transmitted by the SMS adaptor to another application, then you should add the following option in the JENNIFER server.

```
ERROR_PLC_REJECTED = true
```

By default, the following alert types are set to be transmitted to other applications by the SMS adaptor.

- ERROR_JVM_DOWN
- ERROR_OUTOFMEMORY
- ERROR_HIGH_RATE_REJECT
- ERROR_SERVICE_QUEUING
- USER_DEFINED_FATAL

- SYSTEM_MESSAGE
- USER_DEFINED_MESSAGE

11.5.2.SMS Adaptor provided by JENNIFER

We will describe the SMS adaptor that JENNIFER provides as part of the basic package. To develope the new SMS adaptor, please refer to [Alert Data Transmission].

11.5.2.1. Alert Logging

The com.javaservice.jennifer.server.AlertToLogFile SMS adaptor records alerts in the JENNIFER server log file.

First, using the sms_adapter_class_name option of the JENNIFER server, you should set the SMS adaptor.

```
sms_adapter_class_name =
com.javaservice.jennifer.server.AlertToLogFile
```

The following message log format is used in the JENNIFER log file.

```
ALERT:SERVER_DOWN:W11:20060323/123350:000
ALERT:ERR_SYSTEM_CPU:W11:20060323/123350:000:96.6%
```

11.5.2.2. Sending Mail

The com.javaservice.jennifer.server.AlertMail SMS adaptor sends alerts via emails. First, using the sms_adapter_class_name option of the JENNIFER server, you should set the SMS adaptor.

```
sms_adapter_class_name = com.javaservice.jennifer.server.AlertMail
```

Next, you should set the sending mail server information using the options that begin with default_sms.

```
default_sms.mail.smtp.host = jennifersoft.com
default_sms.mail.smtp.port = 25
default_sms.emailToList = tech@jennifersoft.com,admin@jennifersoft.com
default_sms.emailFrom = tech@jennifersoft.com
```

Now, you should copy the Java library's activation.jar and mail.jar files for sending mail in the JENNIFER_HOME/server/common/lib directory.

11.5.2.3. SNMP TRAP

The com.javaservice.jennifer.server.SnmpTrap SMS adaptor sends alerts using SNMP Trap.

Notice: The following explanation assumes that the reader understands SNMP TRAP.

First, you should set the SMS adaptor using the sms_adapter_class_name option of the JENNIFER server.

```
sms_adapter_class_name = com.javaservice.jennifer.server.SnmpTrap
```

And using the options that begin with snmp_trap, you should set the SNMP TRAP information.

```
# Private Enterprise Number(PEN) for JenniferSoft is 27767.
snmp_trap_oid =  1.3.6.1.4.1.27767.1.1
snmp_trap_target_address = 127.0.0.1/162
snmp_trap_target_community = public
```

For testing, run the file, JENNIFER_HOME/server/doc/snmp-trap/trapview.bat. You can check the Trap message sent by the JENNIFER server using the console. The following messages will be shown.

```
TRAP[requestID=506943247, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1.
27767.1.1 = 10:46:19,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
TRAP[requestID=942904425, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1.
27767.1.1 = 10:46:21,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
TRAP[requestID=270685152, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1.
27767.1.1 = 10:46:27,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
TRAP[requestID=1772678772, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1
.27767.1.1 = 10:46:27,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
TRAP[requestID=1562294832, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1
.27767.1.1 = 10:46:27,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
```

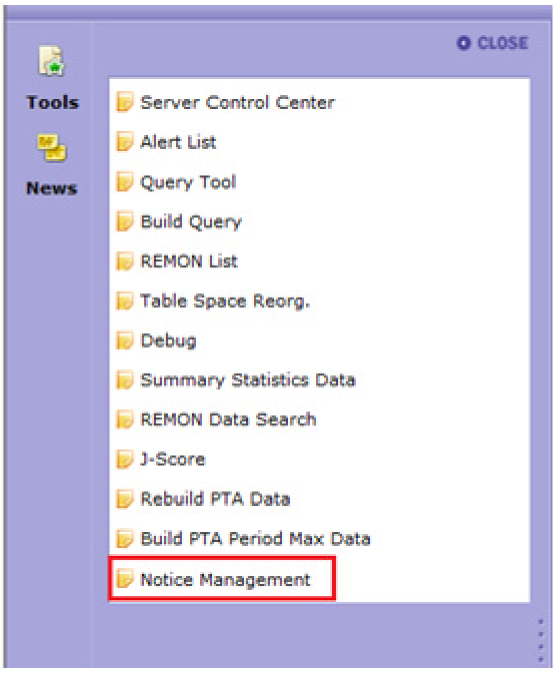
11.5.2.4. SMS Service

If you want to transmit alerts via SMS, you must use an external SMS service. Your JENNIFER license does not include an external SMS service. Therefore, to send alerts via SMS, you will need to sign a contract with an external SMS service provider, and develop your SMS adaptor based on the API provided by the SMS service provider.

11.6. Notice

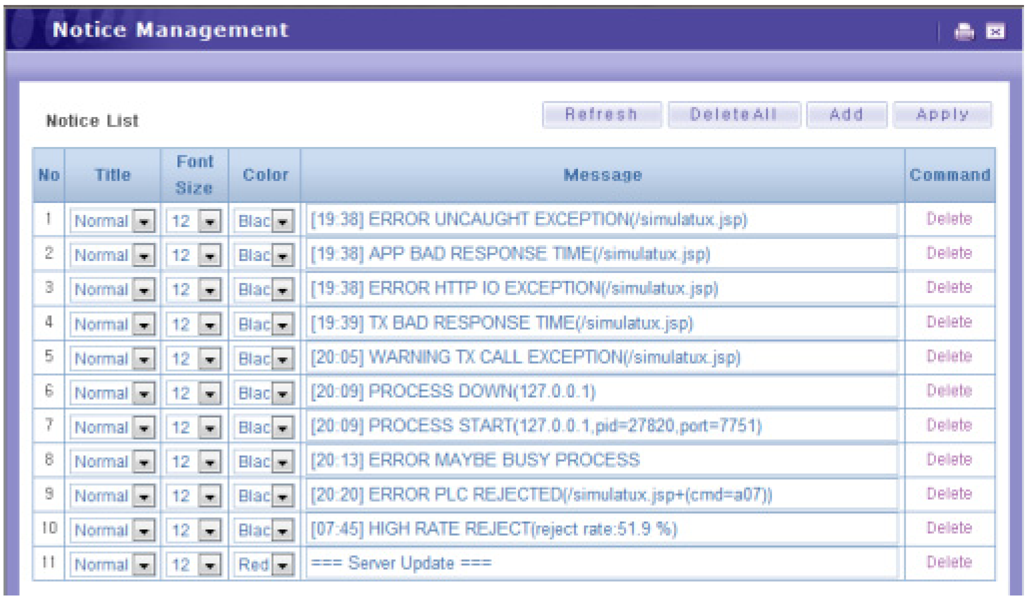
AutoNotice is new feature in JENNIFER® 4.5. System administrators are able to send messages to other parties who are you using JENNIFER with Notice function.

Figure 11-10:Notice



For adding message, simply click the [Add] button, or you can modify an existing message by directly typing to the Message Field.

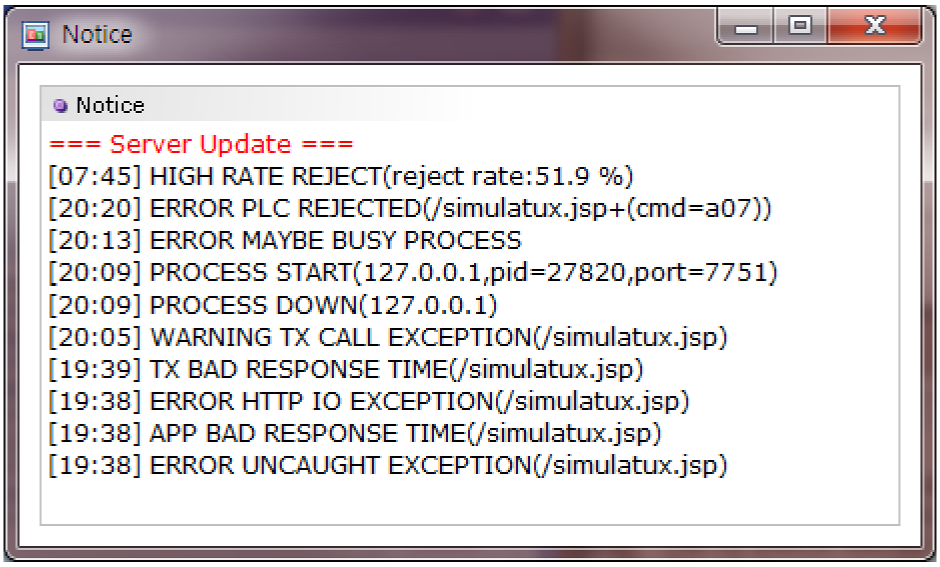
Figure 11-11:Notice Management



No	Title	Font Size	Color	Message	Command
1	Normal	12	Blac	[19:38] ERROR UNCAUGHT EXCEPTION(/simulatux.jsp)	Delete
2	Normal	12	Blac	[19:38] APP BAD RESPONSE TIME(/simulatux.jsp)	Delete
3	Normal	12	Blac	[19:38] ERROR HTTP IO EXCEPTION(/simulatux.jsp)	Delete
4	Normal	12	Blac	[19:39] TX BAD RESPONSE TIME(/simulatux.jsp)	Delete
5	Normal	12	Blac	[20:05] WARNING TX CALL EXCEPTION(/simulatux.jsp)	Delete
6	Normal	12	Blac	[20:09] PROCESS DOWN(127.0.0.1)	Delete
7	Normal	12	Blac	[20:09] PROCESS START(127.0.0.1,pid=27820,port=7751)	Delete
8	Normal	12	Blac	[20:13] ERROR MAYBE BUSY PROCESS	Delete
9	Normal	12	Blac	[20:20] ERROR PLC REJECTED(/simulatux.jsp+(cmd=a07))	Delete
10	Normal	12	Blac	[07:45] HIGH RATE REJECT(reject rate:51.9 %)	Delete
11	Normal	12	Red	=== Server Update ===	Delete

By clicking the [Apply] btton, the modified message is sent to everybody.

Figure 11-12:Notice Modification

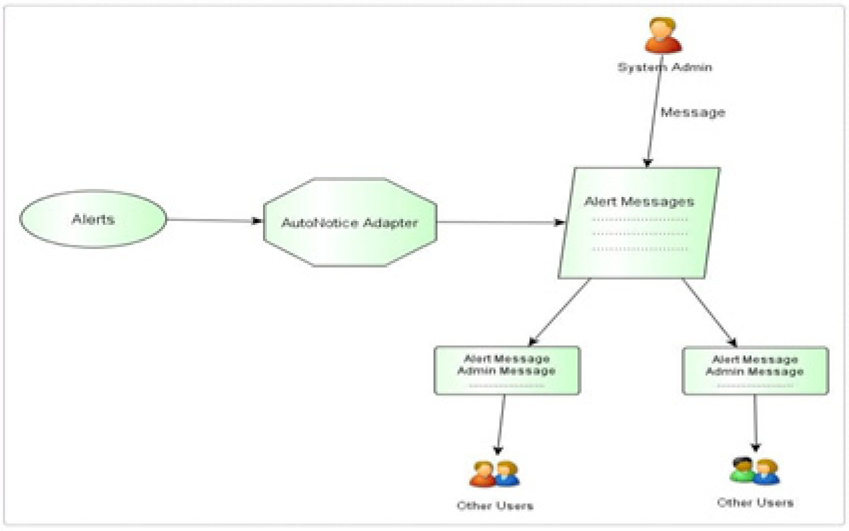


```
=== Server Update ===
[07:45] HIGH RATE REJECT(reject rate:51.9 %)
[20:20] ERROR PLC REJECTED(/simulatux.jsp+(cmd=a07))
[20:13] ERROR MAYBE BUSY PROCESS
[20:09] PROCESS START(127.0.0.1,pid=27820,port=7751)
[20:09] PROCESS DOWN(127.0.0.1)
[20:05] WARNING TX CALL EXCEPTION(/simulatux.jsp)
[19:39] TX BAD RESPONSE TIME(/simulatux.jsp)
[19:38] ERROR HTTP IO EXCEPTION(/simulatux.jsp)
[19:38] APP BAD RESPONSE TIME(/simulatux.jsp)
[19:38] ERROR UNCAUGHT EXCEPTION(/simulatux.jsp)
```

11.6.1.Auto Notice

It is AutoNotice that make a notice automatically for users.

Figure 11-13:Auto Notice



You can use AutoNotice by registering sms Adapter. .

```
sms_adapter_class_name = com.javaservice.jennifer.server.AutoNotice
```

11.6.2.Detete Notice

If you don't want to use Notice function as the previous version, set the JENNIFER server option as follows: .

```
ui_dashboard_notice=True
```

16 Server Configuration & Management

12.1. JENNIFER Server Startup & Shutdown

If you want to startup the JENNIFER server, execute startup.sh in the JENNIFER_HOME/server/bin directory.

```
./startup.sh
```

Notice: But if you want to monitor the .NET agents only, then use startup.net.bat to start the JENNIFER Server.

You should install Java 1.5 or higher in order to execute JENNIFER server. In case of using Java 1.4.2, you should follow the below steps.

- Copy the JENNIFER_HOME/server/doc/jdk142/bin/jmx.jar file to the JENNIFER_HOME/server/bin directory.
- Copy the JENNIFER_HOME/server/doc/jdk142/common/endorsed directory under the JENNIFER_HOME/server/common directory.
- Start the JENNIFER server.

You can identify the standard output in the JENNIFER_HOME/server/logs/catalina.out file.

If you want to stop the JENNIFER server, execute shutdown.sh in the JENNIFER_HOME/server/bin directory.

```
./shutdown.sh
```

12.2. JENNIFER Agent Configuration File

All JENNIFER agent configurations, including monitoring standards and methods, and network port number assignments, is set by the JENNIFER agent options. The JENNIFER agent configuration file is the file designated by the Java jennifer.config option when the JENNIFER agent is installed.

12.2.1.Format of the Configuration File

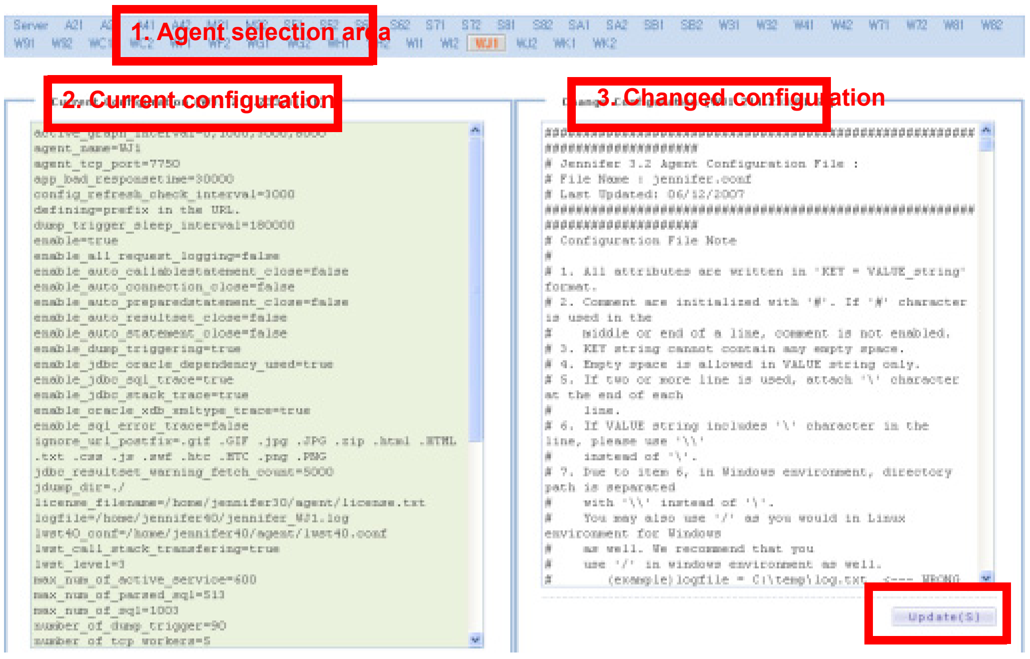
The JENNIFER agent option is saved in a text formatted configuration file. The file has the following characteristics:

- The format is ‘key = value’. The key represents the JENNIFER agent option.
- Any lines starting with [#] are comments. However, the appearance of a [#] symbol in the middle of a text string will not indicate a comment.
- Instead of [=], you can use ‘SPACE’ to separate the key and the value.
- For the above reason, it is prohibited to use ‘SPACE’ as the key.
- However, you can include ‘SPACE’ in the value.
- If you need to use more than one line, add [\] at the end of the line. If you need to use [\] for text, then use [\\] to finish the line.
- For the reason shown above, you must use [\\], not [\] to distinguish the directory in the Microsoft windows environment. However, you can use [/] in Windows, as in the Unix environment, and you are recommended to do so.
- With a few exceptions, option’s change is reflected in the system as soon as the change is made. Add these options to use.

12.2.2.Changing the Configuration

The configuration file can be directly modified using a text editor, or using the **[Properties | Configuration]** menu in the JENNIFER client. However, only the users in Administrator group are allowed to modify the configuration options through the JENNIFER client.

Figure 12-1:JENNIFER Option Setting Screen



Server A21 A2 A41 A42 M21 M22 S51 S52 S6 S62 S71 S72 S81 S82 SA1 SA2 SB1 SB2 W31 W32 W41 W42 W71 W72 W81 W82
W91 W92 WC1 WC2 WF1 WF2 WG1 WG2 WH1 W2 WI1 WI2 WJ1 WJ2 WK1 WK2
1. Agent selection area
2. Current configuration
3. Changed configuration
Update(S)

```
active_graph_interval=0,1000,3000,8000
agent_name=WJ1
agent_tcp_port=7750
app_bad_responsetime=30000
config_refresh_check_interval=3000
defining=prefix in the URL.
dump_trigger_sleep_interval=180000
enable=true
enable_all_request_logging=false
enable_auto_callablestatement_close=false
enable_auto_connection_close=false
enable_auto_preparedstatement_close=false
enable_auto_resultset_close=false
enable_auto_statement_close=false
enable_dump_triggering=true
enable_jdbc_oracle_dependency_used=true
enable_jdbc_sql_trace=true
enable_jdbc_stack_trace=true
enable_oracle_xdb_xmltype_trace=true
enable_sql_error_trace=false
ignore_url_postfix=.gif .GIF .jpg .JPG .zip .html .HTML
.txt .css .js .swf .htc .HTC .png .PNG
jdbc_resultset_warning_fetch_count=5000
jdump_dir=./
license_filename=/home/jennifer30/agent/license.txt
logfile=/home/jennifer40/jennifer_WJ1.log
lwst40_conf=/home/jennifer40/agent/lwst40.conf
lwst_call_stack_transfering=true
lwst_level=3
max_num_of_active_service=600
max_num_of_parsed_sql=513
max_num_of_sql=1003
number_of_dump_trigger=90
number of tcp workers=5
```

```
##########################################################
#####################
# Jennifer 3.2 Agent Configuration File :
# File Name : jennifer.conf
# Last Updated: 06/12/2007
##########################################################
#####################
# Configuration File Note
#
# 1. All attributes are written in 'KEY = VALUE_string'
format.
# 2. Comment are initialized with '#'. If '#' character
is used in the
#    middle or end of a line, comment is not enabled.
# 3. KEY string cannot contain any empty space.
# 4. Empty space is allowed in VALUE string only.
# 5. If two or more line is used, attach '\' character
at the end of each
#    line.
# 6. If VALUE string includes '\' character in the
line, please use '\\'
#    instead of '\'.
# 7. Due to item 6, in Windows environment, directory
path is separated
#    with '\\' instead of '\'.
#    You may also use '/' as you would in Linux
environment for Windows
#    as well. We recommend that you
#    use '/' in windows environment as well.
#       (example)logfile = C:\temp\log.txt  <--- WRONG
```

1. Agent selection area - To check or change the JENNIFER agent option, select the JENNIFER agent here.
2. Current configuration - In the current configuration on the left, review the JENNIFER agent option.
3. Changed configuration - Modify the existing option, or add a new one in the configuration change input form at the right. Click the [change] button at the bottom to apply the change.

You can use the config_refresh_check_interval option of the JENNIFER agent to periodically review the changes in the JENNIFER agent configuration file.

```
config_refresh_check_interval = 3000
```

12.3. JENNIFER Server Network Structure

This chapter describes the manner in which the network of the JENNIFER server is organized to process various network communications between the JENNIFER client or JENNIFER agent and the JENNIFER server. This chapter mainly deals with the port number configuration. If the default port number is already used by other applications, then it should be modified. In addition, when multiple JENNIFER servers are installed in the same hardware, the port number should be modified to prevent duplicate use.

12.3.1.Client Configuration

The JENNIFER server and the JENNIFER client use HTTP protocols to provide a web-based user interface. The default port number is 7900 and the default port number is 7999 to stop the JENNIFER server. If you want to change the HTTP port number, modify the following in the JENNIFER_HOME/server/bin/catalina.sh(bat) file.

In case of Unix or Linux, set the port number as following in the catalina.sh file. .

```
JAVA_HOME="$JAVA_HOME"

if [ -z "${STARTUP_PORT}" ]
then
    export STARTUP_PORT="7900"
fi

if [ -z "${SHUTDOWN_PORT}" ]
then
    export SHUTDOWN_PORT="7999"
fi
```

In case of windows, set the port number as following in the catalina.bat file.

```
set JAVA_HOME=%JAVA_HOME%

if "%STARTUP_PORT%" == "" SET STARTUP_PORT=7900
if "%SHUTDOWN_PORT%" == "" SET SHUTDOWN_PORT=7999
```

The Java applets obtain the data required to organize the charts from the JENNIFER server through TCP communication. The default TCP port number for the JENNIFER sever is set using the server_tcp_port option of the JENNIFER server. The default port number is 6701.

```
server_tcp_port = 6701
```

12.3.2.JENNIFER Agent Configuration

The JENNIFER agent transmits performance data to the JENNIFER server via the UDP. The JENNIFER server then receives the performance data sent by the JENNIFER agent through three different UDP ports. However, each UDP port receives unique performance data.

The JENNIFER agent transmits the data related to the beginning and the end of all the transactions through the UDP port, which is set using the server_udp_runtime_port option of the JENNIFER server. This data is very small, and it is mostly used for an X-view charts. The default port number is 6901.

```
server_udp_runtime_port = 6901
```

The JENNIFER agent transmits general performance data, such as the arrival rate and the average response time, through the UDP port, which is set in the server_udp_listen_port option of the JENNIFER server once every second. The default port number is 6902.

```
server_udp_listen_port = 6902
```

The JENNIFER agent transmits transaction frofile data of X-View through the UDP port, which is set in the server_udp_lwst_call_stack_port option of the JENNIFER server once every two-second. The default port number is 6903.

```
server_udp_lwst_call_stack_port = 6703
```

Notice: When modifying the UDP port number, make the change on the JENNIFER agent as well as the JENNIFER server.

It is necessary to assign the IP binding address that is used when the JENNIFER server receives the performance data transmitted by the JENNIFER agent via UDP. This value corresponds to the second parameter IP address of the new java.net.DatagramSocket(port, ip) creator in the Java TCP socket programming. If there are multiple network cards in the hardware, only the requests coming into each network card are accepted. However, if you set the udp_server_host attribute to "0.0.0.0", then you can receive the packets coming into all network cards. Please note that in some cases, it is necessary to use the server IP address instead of "0.0.0.0".

```
udp_server_host = 0.0.0.0
```

You can set a timeout option for TCP connections from the JENNIFER server to the JENNIFER agent. Use the agent_tcp_connect_timeout option to set a timeout for Socket connections. It is expressed in milliseconds and 3000 is default.

```
agent_tcp_connect_timeout = 3000
```

After connecting the Socket, use the agent_tcp_io_timeout option to set a timeout for the data reading process. It is expressed in milliseconds and 5000 is default.

```
agent_tcp_io_timeout = 5000
```

Notice: If the network connection is delayed, the JENNIFER server will log the errors. In this case, rather than changing the option, you are recommended to adjust the network environment between the JENNIFER agent and the server.

12.3.3.Firewall Configuration for Using JENNIFER

If there is a firewall between the JENNIFER server and the JENNIFER agent or client, change its configuration, so that the port number can successfully pass through it.

- A user should be able to access the JENNIFER server through the HTTP 7900 and TCP 6701 ports from the user's computer.
- The JENNIFER agent should be able to access the JENNIFER server through the UDP 6901, 6902 and 6703 ports.
- A user should be able to access the JENNIFER agent through the TCP 7705 ports from the JENNIFER server.

Reference: For firewall tests on the UDP network, refer to [Network Tests].

12.4. Server Log Management

The JENNIFER server log file can be set using the logfile option of the JENNIFER server.

```
logfile = ../logs/jennifer.log
```

To manage the log file daily, set the enable_logfile_daily_rotation option of the JENNIFER server into true. The default is true.

```
enable_logfile_daily_rotation = true
```

By default, a log is recorded in the JENNIFER_HOME/server/logs/jennifer.YYYY-MM-DD.log file.

Notice: After changing the logfile option, you must restart the JENNIFER server for the change to take effect.

The JENNIFER server uses Apache Tomcat, and the tomcat logs are sorted by dates and saved in the JENNIFER_HOME/server/logs directory.

Table 12-1: Tomcat Log File

Log file	Description
admin.YYYY-MM-DD.log	This is a log file that Tomcat uses internally. it does not record important messages.
catalina.YYYY-MM-DD.log	Records the messages displayed on the console.
host-manager.YYYY-MM-DD.log	This is a log file that Tomcat uses internally. it does not record important messages.
localhost.YYYY-MM-DD.log	Record the error and exception messages.
manager.YYYY-MM-DD.log	This is a log file that Tomcat uses internally. it does not record important messages.

Notice: The log related to the Tomcat server are catalina.YYYY-MM-DD.log and localhost.YYYY-MM-DD.log.

12.5. Operating Multiple JENNIFER Servers on the Same Hardware

This chapter describes how to operate multiple JENNIFER servers on the same hardware. Regardless of currently operating JENNIFER server, install one JENNIFER server. Then, write start/stop script according to the number of JENNIFER server. Input HTTP port number and stop port number of the JENNIFER server in the script and set the JENNIFER server configuration file path.

In case of Unix or Linux, write them as following in the JENNIFER_HOME/server/bin/ start_server_01.sh. You can use an arbitrary file name.

```
export JAVA_HOME=/usr/java/jdk1.6.0_11

export STARTUP_PORT=7901
export SHUTDOWN_PORT=7991

export JAVA_OPTS=-Djennifer.config=/jennifer/data/conf/
jennifer_01.properties

./startup.sh
```

Then, write as following in the JENNIFER_HOME/server/bin/shutdown_server_01.sh file. You can use an arbitrary file name.

```
export JAVA_HOME=/usr/java/jdk1.6.0_11

export STARTUP_PORT=7901
export SHUTDOWN_PORT=7991

export JAVA_OPTS=-Djennifer.config=/jennifer/data/conf/
jennifer_01.properties

./shutdown.sh
```

In case of windows, write as following in the JENNIFER_HOME/server/bin/ start_server_01.bat. You can use an arbitrary file name.

```
set JAVA_HOME=C:\Program Files\Java\jdk1.6.0_11

SET STARTUP_PORT=7901
SET SHUTDOWN_PORT=7991

set JAVA_OPTS=-Djennifer.config=C:/jennifer/data/conf/
jennifer_01.properties

startup run
```

Then, write as following in the JENNIFER_HOME/server/bin/shutdown_server_01.bat file. You can use an arbitrary file name.

```
set JAVA_HOME=C:\Program Files\Java\jdk1.6.0_11

SET STARTUP_PORT=7901
SET SHUTDOWN_PORT=7991
set JAVA_OPTS=-Djennifer.config=C:/jennifer/data/conf/
jennifer_01.properties

shutdown run
```

The following is different JENNIFER server options for each JENNIFER server.

- domain_name
- server_tcp_port
- logfile
- upload_directory
- data_directory
- system.derby.system.home
- backup_root

If you want to set the Java temp directory differently, set java.io.tmpdir in the start script as a JAVA_OPTS environment variable.

12.6. JENNIFER Server Debugging

12.6.1.Event Log Record

The main event logs related to users and privileges can be saved in the EVENT_LOG table of the admin database. If you want to record event logs, set the enable_event_log option of the JENNIFER server to true. By default, it is set to false.

```
enable_event_log = true
```

The following table describes the main event logs.

Table 12-2: Main Event Log

Event log ID	Description
JENNIFER_SERVER_START	An event log for when the JENNIFER server starts.
JENNIFER_SERVER_STOP	An event log for when the JENNIFER server stops.
ADD_USER	An event log for when a user is added (including the information of the group assigned to the user).
CHANGE_USER	An event log for when a user is changed (including the information of the group assigned to the user).
REMOVE_USER	An event log for when a user is removed.
ADD_GROUP	An event log for when a group is added (including the information of the JENNIFER agent list).
CHANGE_GROUP	An event log related to a group changes(including the JENNIFER agent list).
REMOVE_GROUP	An event log for when a group is remove.
ADD_ROLE	An event log for when an authority is added (including the information of the group assigned to the authority).
CHANGE_ROLE	An event log related to authority changes (including the information of the group assigned to the authority).
REMOVE_ROLE	An event log for when an authority is removed.
CHANGE_PASSWORD	An event log related to a password changes.
CHANGE_PASSWORD_BY_ADMIN	An event log for when a password is changed by a user belonging to an admin group.
LOGIN_SUCCESS	An event log related to a login success.
LOGIN_FAILURE	A login failure event log. It records the cause of a failed login.
LOGOUT_SUCCESS	A logout success event log. It distinguishes between a logout by the user and a logout due to a session timeout.
LOGOUT_FAILURE	A logout failure event log.
INVALID_LOGIN_LOCK	If a user enters an incorrect password a certain number of times, he will be blocked from using the account. This event log records this event.

12.6.2.JENNIFER Server SQL Logging

You can log the performance of the internal SQL used by the JENNIFER server and the existence of errors. This data is used to fine-tune the JENNIFER server. To do this, set the enable_server_trace option of the JENNIFER server to 'true'. The 'false' is default.

```
enable_server_trace = true
```

In addition, set the server_trace_filename option of the JENNIFER server for the log file. The servertrace.log is default.

```
server_trace_filename = servertrace.log
```

If the server_jdbc_trace_overthan option is set, the servertrace.log file records any erroneous SQL, or any SQL that is delayed for more than the setting time. It is expressed in milliseconds and 10000 is default.

```
server_jdbc_trace_overthan = 10000
```

If there are too many users for the JENNIFER client, the TCP connection pool could be insufficient and cause an exception. In this case, you can increase the number of Client TCP Worker using the number_of_tcp_pooled_workers option. Before increasing the pool size, it is necessary to check whether the actual users are successfully using the TCP connections. To check this, set the debug_tcp option of the JENNIFER server to 'true'. The 'false' is default. If you add the code above to the JENNIFER server configuration file, whenever a client requests a TCP connection, it will be recorded in a log file.

```
debug_tcp = true
```

12.7. JENNIFER Scheduler

The JENNIFER server supports a scheduler function to manage the data or to execute certain tasks. The scheduler is called TimeActor. JENNIFER provides a basic scheduler, and if necessary, can prepare an arbitrary scheduler to be added.

12.7.1.Basic Scheduler

JENNIFER provides basic schedulers including CleanerActor, SummaryActor , ReportActor and FileCleanerActor.

- CleannerActor - Periodically deletes performance data that has been saved in a file and in the database.
- SummaryActor - Creates statistical performance data to improve the search function for the daily table.
- ReportActor - Creates a report automatically.
- FileCleanerActor - Delete the log and temp file of the JENNIFER server. The temp file is a file in the directory that is set as a java.io.tmpdir environment variable.
- BackupActor - Backup the performance data collected by the JENNIFER.

Set the FileCleanerActor as follows;

```
time_actor_14 =
com.javaservice.jennifer.server.timeactor.FileCleanerActor 02 30
```

The first parameter's 02 means execution time and 30 in the second parameter represents the peoried. Therefore, JENNIFER deletes log/temp files over 30-days ago on every 2pm. But, only the file with log and png extension name is deleted.

Notice: The methods used to configure other schedulers will be discussed in the section that describes each scheduler.

12.7.2.Making a JENNIFER Scheduler

The following method is used to make an arbitrary scheduler.

12.7.2.1. Making a Scheduler Class

A scheduler class must inherit from the com.javaservice.jennifer.server.TimeActor class. However, the TimeActor class is located in the JENNIFER_HOME/server/common/lib/ jenniferserver.jar file, and this file must be registered in the class path.

A JENNIFER scheduler class should distinguish the process method.

```
package sample;

import com.javaservice.jennifer.server.TimeActor;

public class SampleActor extends TimeActor {
    public void process(String[] args) { }
}
```

The parameter of the process method is used to register a scheduler in the JENNIFER server. For example, if a scheduler is registered as follows, then the parameter of the process method becomes {"03", "sample"}. Although the meaning of the parameter varies depending on the scheduler, you are recommended to set the scheduler execution time using the first parameter.

```
time_actor_32 = sample.SampleActor 03 sample
```

The process method of the scheduler is called once per second. Therefore, it is the scheduler itself, not the JENNIFER server, who must determine whether to perform a certain task at a certain time.

The following table describes the main method API provided by the TimeActor class.

Table 12-3: TimeActor Class API

Method	Description
public String getYYYYMMDD()	The current time is expressed as yyyyMMdd.
public static String toString(Calendar day)	The time of the java.util.Calendar object is expressed as yyyyMMdd.
public String getYYYYMMDD(int type, int n)	The previous or subsequent date is expressed as yyyyMMdd. Calendar.YEAR, Calender.DATE or Calender.HOUR can be assigned to the first parameter type. A negative or positive number assigned to the second parameter represents the past or the future. For example, getYYYYMMDD(Calender.MONTH, -1) represents the date that is one month prior to the current date.
public String getHOUR()	The current time is expressed as hh.
Connection getAdmConn() throws SQLException	Returns the java.sql.Connection object for the JENNIFER admin database.
public Connection getDataConn() throws SQLException	Returns the java.sql.Connection object for the JENNIFER performance database.
public void log(String message)	A log message is recorded in the JENNIFER server log file.

Notice: When the scheduler performs certain tasks on the JENNIFER database, it must return the JDBC resource(java.sql.Connection, java.sql.Statement, java.sql.ResultSet) that is used. Otherwise, a failure can occur in the JENNIFER server.

12.7.2.2. Scheduler Class Package and Distribution

After packaging the compiled scheduler class to an arbitrary JAR file, it should be copied to the JENNIFER_HOME/server/common/lib directory.

12.7.2.3. Scheduler Setting

A new scheduler should be registered using the time_actor option of the JENNIFER server.

```
time_actor_32 = sample.SampleActor 03 sample
```

The time_actor_ option of the JENNIFER server should be used to register a scheduler. This option's name should end with two unique numbers, which are used to distinguish the option from other schedulers.

Notice: After you add a new scheduler or change the existing scheduler, restart the JENNIFER server.

12.8. Users, Authority, and Menu

We will now describe the relationship between the user and the group, and the method for assigning privileges to the groups. We will also describe the method for managing the menu, and setting a different menu for each user or group.

12.8.1.User Setting

A user is the holder of an account that is used to log into the JENNIFER server. You can set a different menu and privilege level for each user.

After the JENNIFER server is installed, the admin account will be created. Basically, the admin user can access all of the menus and has complete authority. If you want to add a new user or change the information, select the **[Properties | User Setting | User Setting]** menu.

12.8.1.1. Users and Groups

You can organize a different menu and assign a different privilege level for each user. However, if there are many users, this could become a cumbersome task. For this reason, it is very convenient to group users by type, and set the available menu or privileges for the group. The members of a group can then access the same menu with the same privileges.

12.8.1.2. Admin User and Group

When the JENNIFER server is installed, the admin user and group are automatically generated. The admin user and group cannot be deleted. In addition, the admin user belongs to the admin group, and this cannot be changed arbitrarily.

The user-related information is given as follows.

Table 12-4: User Information

TABLE 13.

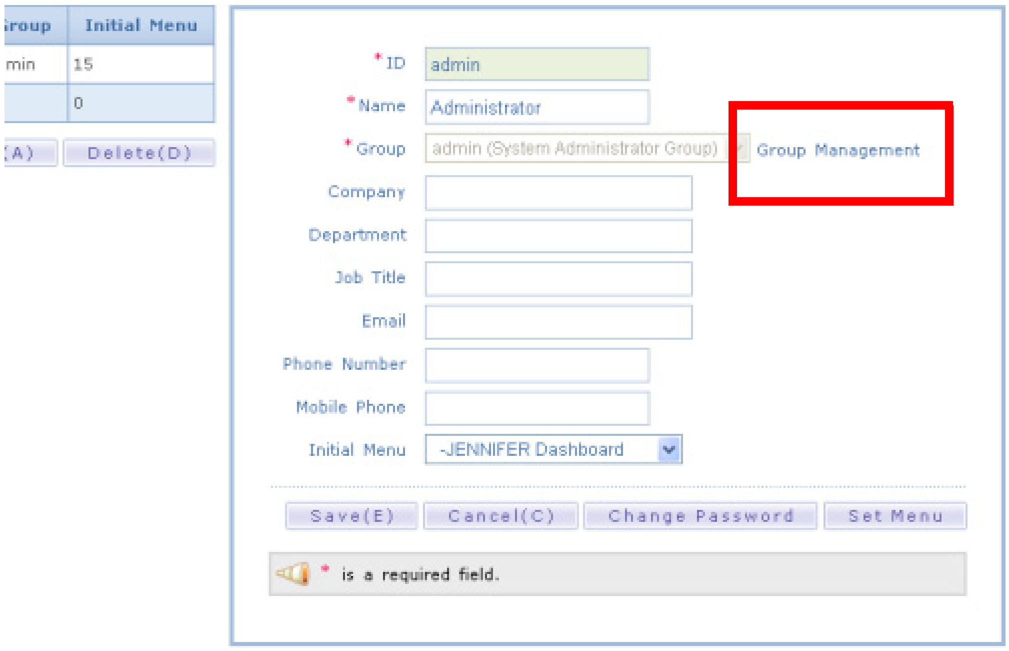
Item	Description
ID	Id used to log-in. Once created, it cannot be changed.
Name	A user's name
Group	A group to which a user belongs. A user can belong to only one group; multiple users can belong to the same group.
Company	A company to which a user belongs.
Department	A department to which a user belongs.
Job Title	A user's job title
E-mail	A user's e-mail address
Phone Number	A user's phone number
Mobile Phone	A user's mobile phone number
Initial Menu	The first menu that appears after the user logs in. It appears when the user's group has access to the menu.

Notice: Companies, departments, job title, email, phone numbers and mobile phone numbers are additional information that does not affect use of the JENNIFER server.

12.8.1.3. Group Management

By clicking the **[Group Management]** link next to the drop-down box to select a group in the user information editing screen, you can edit the group information.

Figure 12-1:Group Management Link



roup	Initial Menu
min	15
	0

* ID	admin
* Name	Administrator
* Group	admin (System Administrator Group)
Company	
Department	
Job Title	
Email	
Phone Number	
Mobile Phone	
Initial Menu	-JENNIFER Dashboard

* is a required field.

Group-related information is as follows:

Table 12-1: Group Information

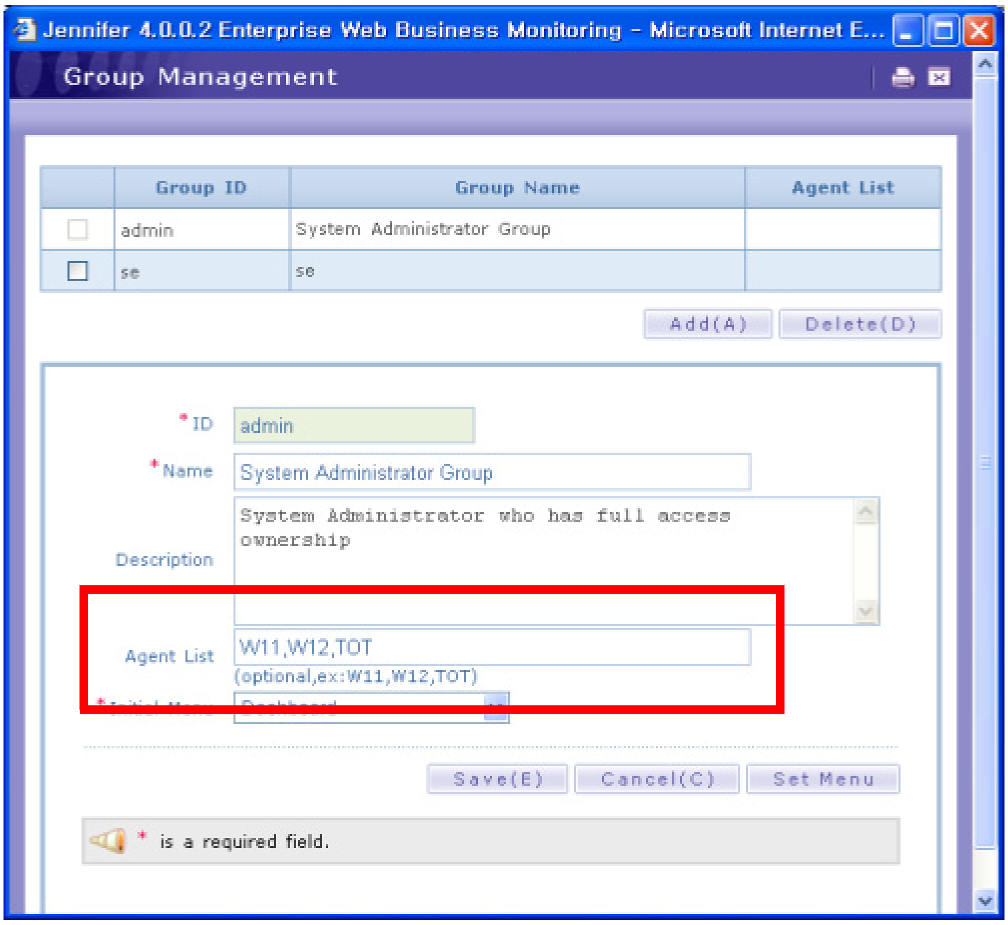
Item	Description
ID	Group ID
Name	Group name
Description	Description of the group
Agent List	Refer to Assigning an Agent to Each Group.
Initial Menu	The first menu that appears after the user logs in. It appears when the user's group has access to the menu. The initial menu set in the user information can override the initial menu set in the group information. If you do not set the initial menu, the **[Dashboard | JENNIFER Dashboard]** menu will be the initial menu**.**

12.8.1.4. Assigning an Agent to Each Group

You may not assign a different monitored agent to each user, but you can assign a different monitored agent to each group. For instance, if the W11, W12 and W13 agents are monitored by the JENNIFER server, you can tell the A group to monitor W11 and W12 and the B group to monitor W12 and W13.

In the group information editing screen, you can enter the names of the agents to be monitored by a certain group, using a comma (,) as a separator in the Agent List. If you do not enter the virtual agent TOT that refers to all agents, then you cannot check the information for all the agents in the dashboard.

Figure 12-2:Agent List Configuration Screen


Jennifer 4.0.0.2 Enterprise Web Business Monitoring - Microsoft Internet E...
Group Management
Add(A)
Delete(D)
Save(E)
Cancel(C)
Set Menu

	Group ID	Group Name	Agent List
☐	admin	System Administrator Group	
☐	se	se	

*ID	admin
*Name	System Administrator Group
Description	System Administrator who has full access ownership
Agent List	W11,W12,TOT (optional,ex:W11,W12,TOT)
*Initial Menu	Dashboard

* is a required field.

12.8.1.5. Limiting the Number of Users

By default, the number of simultaneous users of the JENNIFER server is unlimited. However, you can limit it by using the userlogin_count_limit option of the JENNIFER server.

```
userlogin_count_limit = 10
```

The JENNIFER server periodically performs TCP communication with the JENNIFER client(applet). The number of sockets in the JENNIFER server that perform TCP communication with the JENNIFER client is limited. For that reason, you must use the number_of_tcp_pooled_workers option of the JENNIFER server. The default is 80. On average, each JENNIFER client uses about three sockets. As the number of users increases, it becomes difficult to assign sufficient sockets for TCP connections. As a result, the JENNIFER client may not operate properly. To addrsss this issue, use the userlogin_count_limit option to limit the number of users.

12.8.1.6. Preventing Multiple Log-in's with the Same Account

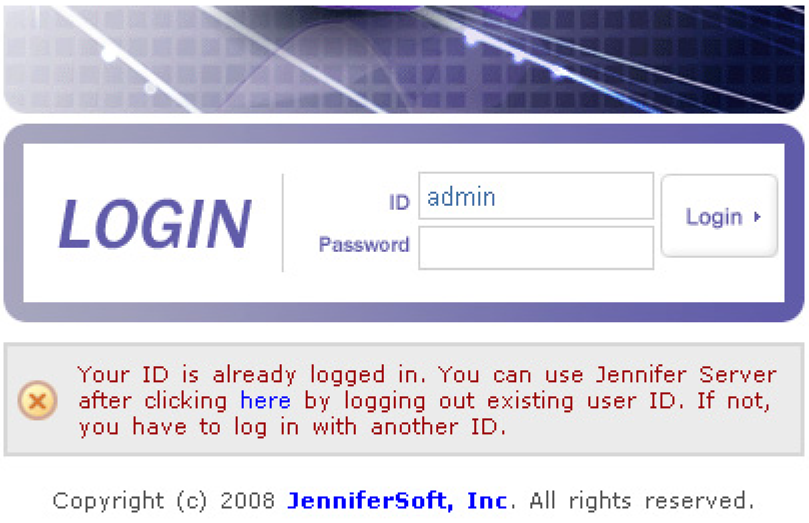
By default, multiple users can access the JENNIFER server with the same account. For instance, if user A uses the admin account to access the JENNIFER server, user B can also log into the JENNIFER server with the same admin account.

However, if you set the prevent_duplicated_login option of the JENNIFER server to true, then you can prevent multiple users from accessing the JENNIFER server with the same user account.

```
prevent_duplicated_login = true
```

If you prevent multiple log-ins with the same account, the user who logs in second can push away the user that is already logged in. For instance, if user B attempts to access the JENNIFER server with the same admin account that user A is already logged into the JENNIFER server with, the JENNIFER server tells user B that there is already another user logged in with that admin account.

Figure 12-3:Multiple Login Checking Screen



ID: admin

Your ID is already logged in. You can use Jennifer Server after clicking here by logging out existing user ID. If not, you have to log in with another ID.

Copyright (c) 2008 JenniferSoft, Inc. All rights reserved.

At this point, user B has the option of forcing user A, who is already logged in, to be logged off so that user B can log into the JENNIFER server. If user B chooses this option, user A is automatically logged out.

Notice: Regardless of the prevent_duplicated_login option, you are recommended to use different user IDs for different users..

12.8.1.7. Checking the User Logged In

By selecting the **[Properties | User Setting | Login Users]** menu, you can check the users that are logged into the JENNIFER server. The underlined row in the login user list is the user himself.

Table 12-2: List of Login Users

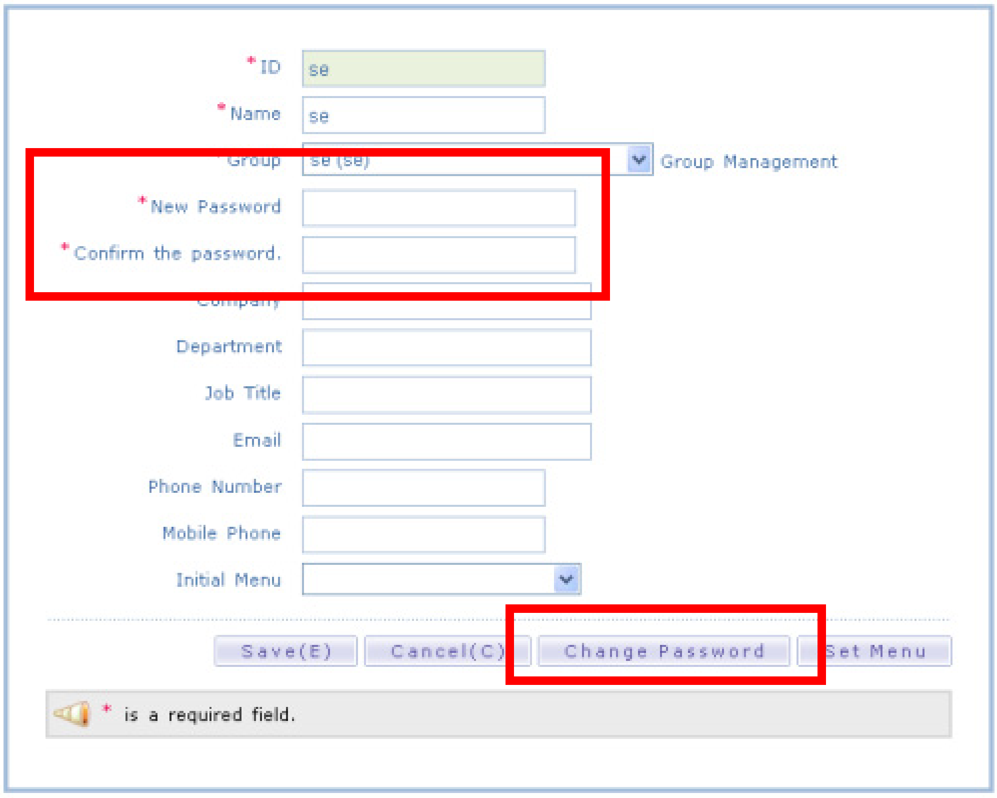
Items	Descriptions
ID	HTTP session ID
User ID	User ID
User IP	User IP
Login time	Time to log into the JENNIFER server
Last access time	Most recent time to access the JENNIFER server
Language	User language

12.8.1.8. Password Change and Lost Password

Any user can change his ID by selecting the **[Properties | User Setting | Password Change]** menu.

A user in the admin group can change any password, including his own, by selecting the **[Properties | User Setting | User Setting]** menu. Other users' passwords can be changed as follows:

Figure 12-4:Changing Other Users' Passwords



- In the user list, click the user ID for which the password is to be changed.
- If you click the **[Change Password]** button at the bottom of the user information editing screen, the password input field will appear.
- After entering the password in the input field, click the **[Save]** button. Your password will be changed.

If the password of every user in the admin group is lost, do the following:

- Stop the JENNIFER server.
- Change the context parameter, enable_password_check in the JENNIFER_HOME/server/webapps/ROOT/WEB-INF/web.xml file to false. If set to false, then no password is checked when logging in.
- Start the JENNIFER server.
- In the login screen, enter the user ID for the admin group and an arbitrary password to log in.
- Select the **[Properties | User Setting | User Setting]** menu, and change the user password for the admin group.
- Stop the JENNIFER server.

- Change the context parameter, enable_password_check in the JENNIFER_HOME/server/webapps/ROOT/WEB-INF/web.xml file to false.
- Start the JENNIFER server.

12.8.2.Authority Setting

A user can perform garbage collection from the Java application through the JENNIFER server, or edit the JENNIFER server user and group information. The main activities that a user can perform are called authorities. You can set a different authority for each group to which a user belongs.

For example, gc authority means that a user can perform garbage collection from the Java application. If you grant gc authority to group A and do not grant it to group B, then all the users in group A can perform garbage collection, but no one in group B can.

By selecting the **[Properties | User Setting | Authority Setting]** menu, you can check the authority list and decide to grant a specific authority to a specific group.

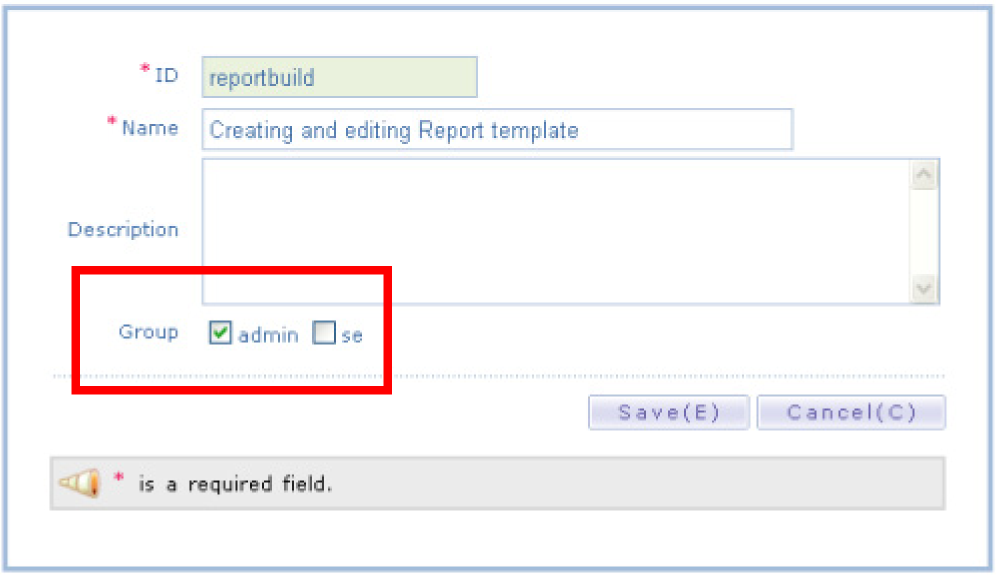
Table 12-3: Authority List

Authority ID	Descriptions
codedisassemble	Authority to view the Java class code in binary format
gc	Authority to perform garbage collection from the Java application
groupedit	Authority to edit a group
killthread	Authority to stop the Java thread of the Java application
menuedit	Authority to edit the menu
pageedit	Authority to set a user defined dashboard
reportbuild	Authority to prepare a report template
sqlparam	Authority to view the SQL parameters
svcdump	Authority to record the main information of the Java application in a dump file
useredit	Authority to edit users
board	Authority to display the board area

If you want to assign a specific authority to a specific group, do the following.

- Click on the authority ID to be assigned to the group.
- In the group field of the authority information editing screen, check and save the group to which you want to assign an authority.

Figure 12-5:Authority Editing Screen

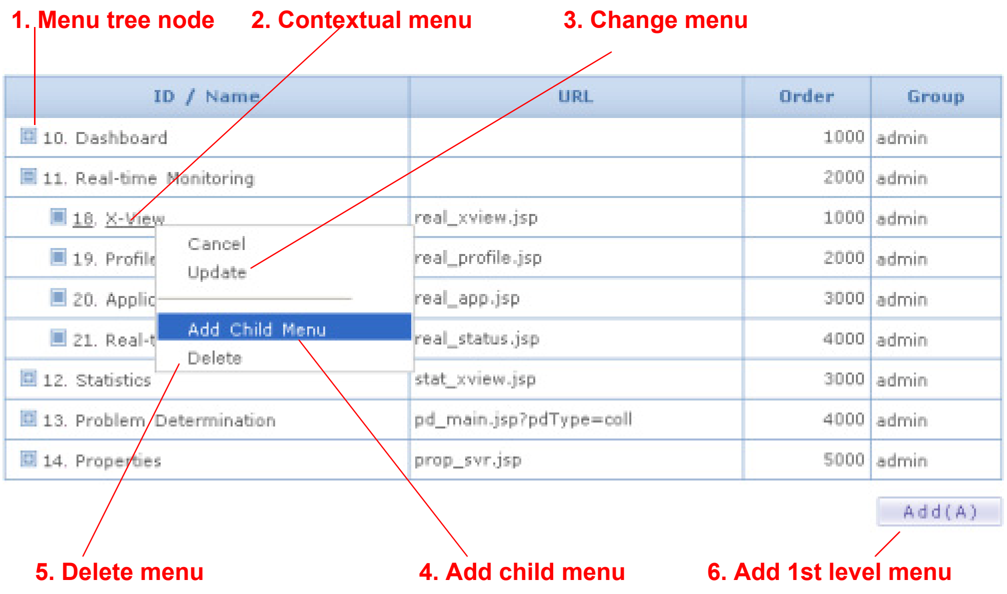


* ID	reportbuild
* Name	Creating and editing Report template
Description	
Group	☑ admin ☐ se

12.8.3.Menu Setting

By selecting the **[Properties | Menu Setting]** menu, you can manage the menu. The JENNIFER server uses three levels of menus.

Figure 12-6:Menu List Screen



ID / Name	URL	Order	Group
10. Dashboard		1000	admin
11. Real-time Monitoring		2000	admin
18. X-View	real_xview.jsp	1000	admin
19. Profile	real_profile.jsp	2000	admin
20. Applic	real_app.jsp	3000	admin
21. Real-t	real_status.jsp	4000	admin
12. Statistics	stat_xview.jsp	3000	admin
13. Problem Determination	pd_main.jsp?pdType=coll	4000	admin
14. Properties	prop_svr.jsp	5000	admin

1. Menu tree node - The menu is comprised of three levels. Using the menu tree node, you can open or close the child menu.
2. Contextual menu - If you click the menu ID/name column, the contextual menu list will appear.
3. Change menu - If you click the **[Change]** menu in the contextual menu list, an input form will be launched at the right, where you can edit the menu.
4. Add child menu - If you click the **[Add child menu]** menu in the contextual menu list, an input field will be launched at the right, where you can enter the child menu. You can enter up to three levels of menus. This menu is deactivated for menu level 3.
5. Delete menu - In the contextual menu list, click **[Delete].** The menu will be deleted.
6. Add1st level menu - If you click the **[Add]** button in the lower right corner of the menu list, you can add a 1st level menu.

When a new menu is created, the JENNIFER server will automatically generate the menu ID.

The basic menu information is as follows:

Table 12-4: Menu Information

Item	Description
Name code	Enter the message key for multiple languages.
Parent menu	To change the location of the menu, change the value in the parent menu field.
Default child menu	When a user clicks on the menu in the upper area, if the URL for the menu is not assigned, then he will be taken to the URL that has been set in the default child menu.
Order	This field is used to determine the order of menus with the same parent menu. The menu with a smaller field value is shown first.
Group	Designates the group that can access the menu. If a user without the authority to access the menu logs in, the menu will not appear.

The menus are classified into several types. Entries for the menu vary depending on the menu type.

- URL link menu - A general menu that is linked to a specific URL. If you select this menu type, then the URL field will appear.
- URL pop-up menu - Opens a specific URL in a new window.
- User defined dashboard menu - Using a drag and drop method, a user can directly organize the screen. If you click the menu in the upper area, it will launch the editing screen in the lower right corner. If you click this button, you will be moved to the user defined dashboard editing screen.

- Report menu - This menu shows a report. If you select this menu type, a drop-down box will be launched, from which you can select various report templates.

12.8.3.1. User/Group Menu Setting

In the menu information editing screen, you can designate groups that can access the menu. It is more convenient to use the user/group setting menu when you set the menu for a certain user or group. The menu for a specific user can be set as follows:

- From the user list of the **[Properties | User Setting | User Setting]** menu, click the user ID to be set.
- Click the **[Set Menu]** button at the bottom of the user information editing screen, and a pop-up menu with the menu list will appear.
- After checking the menu that the user can access, click the **[Save]** button at the bottom.

The menu for a group can be set as follows:

- From the user list of the **[Properties | User Setting | User Setting]** menu, click the user ID to be set.
- Click the **[Group Management]** button at the bottom of the user information editing screen, and a pop-up menu with the group list will appear.
- Click the group ID in the group list to set menu.
- Click the **[Set Menu]** button at the bottom of the user information editing screen, and a pop-up menu with the menu list will appear.
- After checking the menu that the group can access, click the **[Save]** button at the bottom.

Notice: You should select the parent and child menu independently. Even if you select the parent menu, the child menu is not automatically selected. And if you try to select the child menu, not selecting parent menu, the child menu is not displayed in the screen.

The user menu configuration can override the group menu configuration.

12.9. Password Setting

12.9.1.Log-in Setting

For security reasons, you can set various policies related to logging in. First, you can force the user to change his password after logging in for the first time. To enable this function, set the enable_initial_password_change option of the JENNIFER server to true. By default, it is set to false.

```
enable_initial_password_change = true
```

If set to true, the user who has logged in for the first time is forced to change his password on the password change screen before he is allowed to move to other menus.

You can also force the user to change his password after a certain period of time has elasped. To enable this function, set the enable_password_expiration_check option of the JENNIFER server to true. By default, it is set to false.

```
enable_password_expiration_check = true
```

If set to true, general users must change their passwords after 60 days, and users belonging to the admin group must change their passwords after 30 days. Using the password_expiration_days option of the JENNIFER server, you can set the expiration period for your password.

```
password_expiration_days = 60
```

This option is applied to general users and users who belong to the admin group. If you want to set the expiration period for users belonging to the admin group, use the admin_password_expiration_days option of the JENNIFER server.

```
admin_password_expiration_days = 30
```

Finally, you can prevent users from using their account if they failed to enter the correct password within a certain number of attempts. To enable this function, set enable_invalid_login_lock option of the JENNIFER server to true. By default, it is set to false.

```
enable_invalid_login_lock = true
```

If the threshold is set as below, if you enter an incorrect password three times, you will be permanently blocked from using the account. You can set the threshold using the invalid_login_count option of the JENNIFER server.

```
invalid_login_count = 3
```

If a user account is terminated for the reasons described above, then a user belonging to the admin group can change the password in the **[Properties | User Setting]** menu, so that the user of the terminated account can use his account again.

12.9.2.Password Change Setting

To prevent the illegal distribution of passwords by a malicious user, you can set various password policies.

First, you can limit the length of passwords. Using the password_length_min and password_length_max options of the JENNIFER server, you can set the minimum and maximum length of password.

```
password_length_min = 3
password_length_max = 10
```

You can also ask users to include a certain number of uppercase and lowercase letters, numbers and/or special characters. For example, using the password_uppercase_count option of the JENNIFER server, you can set a minimum number of uppercase letters to be included in the password.

```
password_uppercase_count = 3
```

Similarly, minimum numbers of lowercase letters, numbers and special characters can be set using the password_lowercase_count, password_number_count and password_specialcase_count options.

```
password_lowercase_count = 2
password_number_count = 2
password_specialcase_count = 2
```

The following is a list of special characters that can be used in passwords.

```
`~!@#$%^&*()-_=+[{]}\|;:'",<.>/?
```

Using the password_specialcase_list option of the JENNIFER server, you can designate arbitrary special characters that can be included in a password.

```
password_specialcase_list = !@#$%?
```

In addition, you can prevent the reuse of passwords that have been previously used. Using the password_history_count option of the JENNIFER server, you can set the number of previous passwords that cannot be used. Once you set the number, that number of the most recent passwords cannot be used.

```
password_history_count = 6
```

You can also prevent repetitive or consecutive use of the same characters in a password. To prevent repetitive use of the same characters, set the password_enable_repetive_charactor_check option of the JENNIFER server to true. To prevent consecutive use of the same characters, set the password_enable_consecutive_charactor_check option of the JENNIFER server to true.

```
password_enable_repetive_charactor_check = true
password_enable_consecutive_charactor_check = true
```

Finally, you can prevent the use of the user ID or certain words in a password. To use this function, set the password_enable_common_word option of the JENNIFER server to true.

```
password_enable_common_word = true
```

If set to true, then the user cannot use his ID as part of a password. To prevent the use of certain words in a password, first make the password_common_word.txt file in the JENNIFER_HOME/server/bin directory, and enter prohibited words in it. Each word must be separated by a line break.

Notice: If a user belonging to the admin group changes another user's password in the **[Properties | User Setting | User Setting]** menu, the above rules do not apply.

12.10. BBS Setting

By selecting the **[Properties | BBS]** menu, you can access the BBS functions.

12.10.1. BBS Types

BBS types can be set in the bbs_type_list option of the JENNIFER server. BBS types are in compliance with the [BBS type code:BBS type name] format. Each type is separated by a semicolons [;].

```
bbs_type_list = biz:Business;rpt:Report;chat:Personal
```

The following BBS types are provided.

Table 12-5: BBS Types

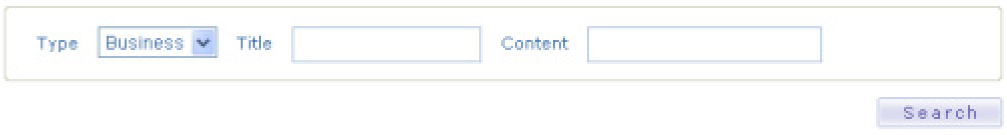
BBS type codes	BBS type names	Descriptions
biz	Business	Business-related BBS
rpt	Report	Report-related BBS
chat	Personal	Personal BBS

A new BBS type can be added by changing the bbs_type_list option of the JENNIFER server.

12.10.2.BBS Searching

You can search for postings by type, title and content. LIKE searching is available for titles and contents.

Figure 12-7:Search Conditions



12.10.3.Posting

A new posting is registered through the following method.

1. Select the **[Properties | BBS]** menu.
2. Click the **[Add]** button at the bottom of the list.
3. After entering the content in the BBS input form, click the **[Write]** button at the bottom.

Table 12-6: BBS Input Fields

Field	Descriptions
BBS type	Select a BBS type. This cannot be changed after posting.
Title	Posting title
Assigned to	If the posting requires some processing, assign a person to process it.

Table 12-6: BBS Input Fields

Field	Descriptions
Due date	If the posting requires some processing, set the due date.
Status	To set the status, select either [N/A], [Service Request], [Processing] or [Done].
Content	Posting content
Attached file	Select the file to be attached. There is no limitations in terms of file size and type.

Notice: You can use the upload_directory option to set the location in theJENNIFER server to save the attached file. The default directory is JENNIFER_HOME/data/upload.

The posting is changed through the following method.

1. Select the **[Properties | BBS]** menu.
2. Click the content title that you want to change in the list.
3. Click the [Update] button in the window.
4. After changing the content in the BBS input form, Click the [Write] button.

The posting is deleted through the following method.

1. Select the **[Properties | BBS]** menu.
2. Click the content title that you want to delete in the list.
3. Click the [Delete] button in the window

Notice: Once a reply has been made to a posting cannot be deleted.

12.10.4.Replying

The replying is registered through the following method.

1. Select the **[Properties | BBS]** menu.
2. Click the content title to be replyed in the list.
3. Click the **[Reply]** button in the window.
4. After replying to the content in the BBS input form, Click the **[Save]** button.

The replying is changed through the following method.

1. Select the **[Properties | BBS]** menu.
2. Click the reply to be changed in the list.

3. Click the **[Update]** button in the window.
4. After changing the reply in the BBS input form, Click the **[Save]** button.

The replying is deleted through the following method.

1. Select the **[Properties | BBS]** menu.
2. Click the reply title to be deleted in the list.
3. Click the **[Delete]** button in the window

12.11. Domain Organization

A single JENNIFER server can only handle a limited number of JENNIFER agents and the related workload. If the system environment exceeds a certain limit, operate multiple JENNIFER servers to distribute the load. This distribution is accomplished through domain organization.

In addition, regardless of domain organization, classify the monitored JENNIFER agents according to work types. Node organization serves this purpose.

12.11.1. What is a Domain?

A single JENNIFER server can only handle a limited number of JENNIFER agents and the related workload. If the system environment exceeds a certain limit, operate multiple JENNIFER servers to distribute the load.

Notice: A single JENNIFER server can handle about 50 JENNIFER agents and 1000 TPS.

However, if you operate multiple JENNIFER servers, the operational cost will rise and the convenience of the user will be reduced. To resolve this problem, JENNIFER 4.0 provides an integrated user interface that organizes the domain to comprehensively manage multiple JENNIFER servers.

A domain refers to an individual JENNIFER server. Domain organization refers to the integration of multiple JENNIFER servers as one.

Notice: When assigning the Java application monitored by the JENNIFER agent to a specific JENNIFER server, first consider similarities in work. If you assign JENNIFER agents according to similarities in work only, then the number of JENNIFER servers tends to increase. Therefore, it is recommended to assign all JENNIFER agents with a low similarity in work and a low TPS to a single JENNIFER server.

12.11.2.Integrated Server

The integrated server is the only server that the user accesses through his web browser.Thus, the integrated server is also a JENNIFER server. A user uses this integrated server to organize the domain and set the users, the authorities and the menu.

When selecting a candidate for the integrated server, consider the following aspects.

- The JENNIFER server can monitor the Java application and play the role of the integrated server at the same time. You are not recommended to operate a JENNIFER server as an integrated server only.
- By assigning the JENNIFER agents and the TPS to multiple JENNIFER servers appropriately, you can minimize the number of JENNIFER servers.
- The integrated server generates additional workloads due to user interface processing as well as performance data collection and analysis. It is recommended to select the JENNIFER server with the least load as the integrated server.
- When the data is collected by REMON, it is more desirable to collect the REMON data from one JENNIFER server than from many JENNIFER servers. In addition, you must designate a JENNIFER server that only collects the REMON data, and use it as the integrated server.

12.11.3.Domain Setting

All tasks related to domain setting are performed by the JENNIFER server that is designated as the integrated server.

The domain is registered through the following method.

1. Select the **[Properties | Domain Setting]** menu.
2. Click the **[Add]** button at the bottom of the list.
3. After entering the content in the domain input form, click the **[Save]** button at the bottom.

The following is a description of the domain input fields.

Table 12-7: Domain Input Fields

Field	Description
ID	A unique ID for the JENNIFER server. No spaces are allowed. Only alphanumeric characters can be used.
Name	The JENNIFER server name
IP	The JENNIFER server's IP address

Table 12-7: Domain Input Fields

Field	Description
HTTP port	The HTTP port number for the JENNIFER server
TCP port	The port number for the JENNIFER server set in the server_tcp_port option
Version	The JENNIFER server version. Enter 4.0. A Jennier server version lower than 4.0 may not integrate the domain.
Status	The JENNIFER server that is set as inactive is not used.

When you add a domain, you can not sometimes call the JENNIFER server's external IP in the hardware where the JENNIFER server installed. To solve this problem, input the JENNIFER server's external and internal IP together by separating with [/] when you set the IP as follows:

```
External_Jennifer_Server_IP/Internal_Jennifer_Server_IP
```

When a domain is added, allow a single sign-on. To do this, set the HTTP address for the integrated server using the domain_url option of the JENNIFER server.

```
domain_url = http://Integrated server IP:port number
```

Notice: If your external and internal IP is separated, use the external IP.

The integrated server also operates as a single sign-on server. A user who is logged into the integrated server can use the service provided by the JENNIFER server registered as a domain in the integrated server without logging into it..

Warning: Before this option is set, the JENNIFER server that is registered as a domain will not be connected to the integrated server with a single sign-on. Log into the JENNIFER server and change the option, or directly modify the configuration file for the JENNIFER server.

The domain is changed through the following method.

1. Select the **[Properties | Domain Setting]** menu.
2. Click the domain ID that you want to change in the list.
3. After changing the content in the domain input form, click the **[Save]** button at the bottom.

The domain is deleted through the following method.

1. Select the **[Properties | Domain Setting]** menu.
2. Click the domain ID to be deleted in the list.
3. Select the check box of the domain that you want to delete in the list.
4. Click the **[Delete]** button at the bottom.

The domain is sorted through the following method.

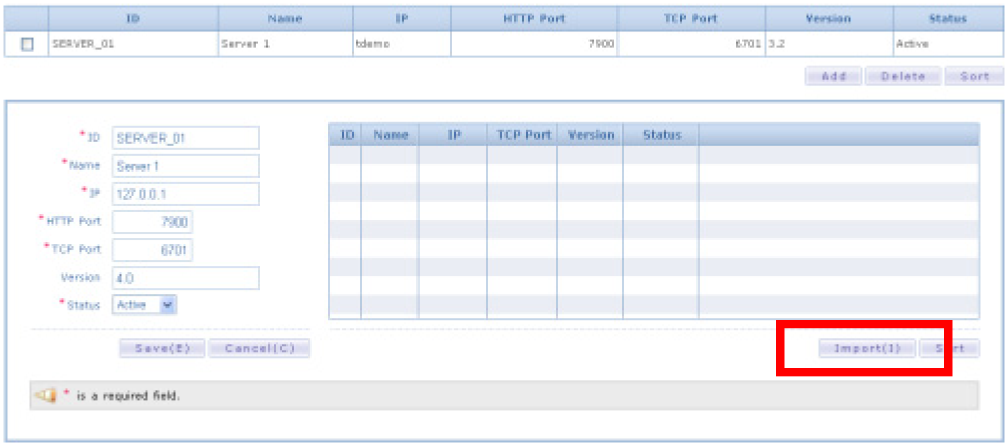
1. Select the **[Properties | Domain Setting]** menu.
2. Click the **[Sort]** button at the bottom.
3. Change the arrangement of domain in the sorting pop-up window.

Notice: After changing the domain organization, log into the integrated server to reflect the change in the system.

12.11.4.Importing a JENNIFER Agent

To organize the node or assign a name to the JENNIFER agent, import the JENNIFER agent after organizing the domain.

Figure 12-8:Importing a JENNIFER Agent



	ID	Name	IP	HTTP Port	TCP Port	Version	Status
☐	SERVER_01	Server 1	tdemo	7900	6701	3.2	Active

*ID	SERVER_01
*Name	Server 1
*IP	127.0.0.1
*HTTP Port	7900
*TCP Port	6701
Version	4.0
*Status	Active

ID	Name	IP	TCP Port	Version	Status

* is a required field.

You can import a JENNIFER agent as follows:

- Select the **[Properties | Domain Setting]** menu for the integrated server.
- Click the domain ID for the JENNIFER agent that you want to import.
- Click the **[Import]** button in the right bottom of the JENNIFR agent list.

Notice: When you click the [Import] button, the data is saved regardless of [Cancel] on the domain field..

You can change the certain name of JENNIFER agent as follows:

- Select the **[Properties | Domain Setting]** menu for the integrated server.
- Click the domain ID beloning to the JENNIFER agent that you want to change the name.

- After selecting the JENNIFER agent name to be changed from the JENNIFER agent list, imput the new name in the name field.
- Click the **[Save]** link in the same row.

You can delete a JENNIFER agent that is no longer being used as follows:

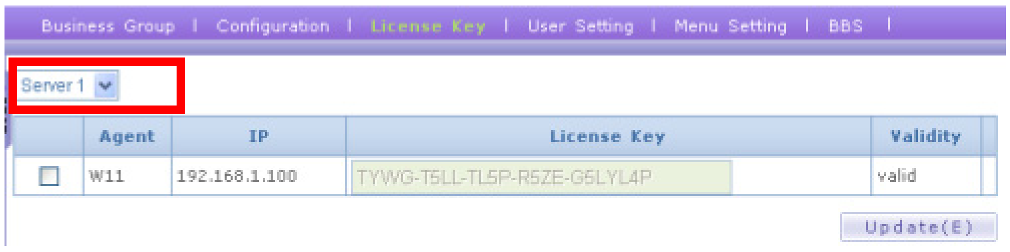
- Select the **[Properties | Domain Setting]** menu for the integrated server.
- Click the domain ID for the JENNIFER agent that you want to delete.
- After selecting the JENNIFER agent to be deleted from the JENNIFER agent list in the lower right corner, click the **[Delete]** link in the same row.

12.11.5.Change in the Interface due to Domain Organization

When a domain is organized, specifically select the domain in the user interface screen.

For example, from the **[Properties | License Key]** menu, select a domain from the drop-down box.

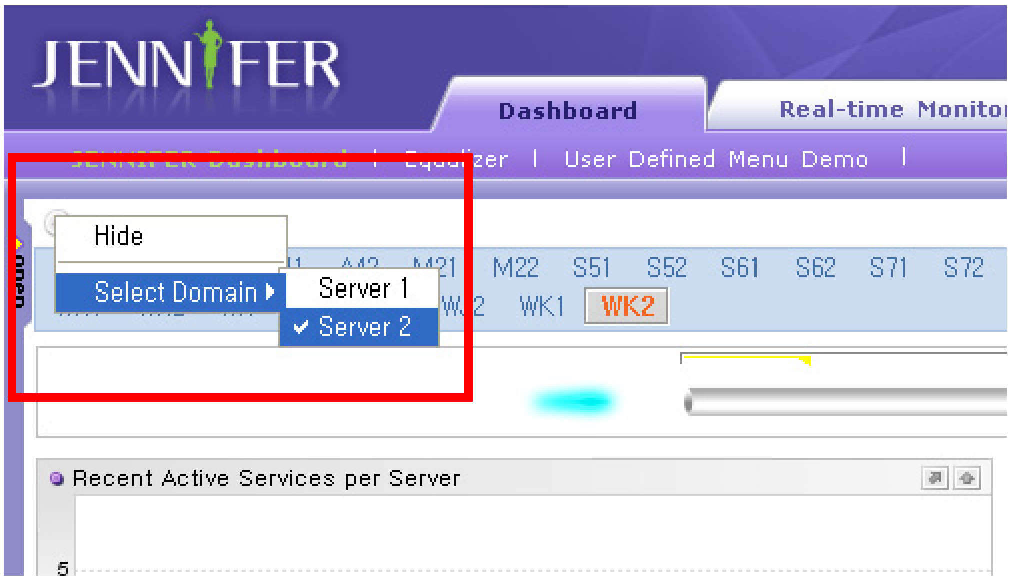
Figure 12-9:License Key Setting - Domain Selection


Business Group | Configuration | License Key | User Setting | Menu Setting | BBS |
Server 1
Update(E)

	Agent	IP	License Key	Validity
☐	W11	192.168.1.100	TYWG-T5LL-TL5P-R5ZE-G5LYL4P	valid

The JENNIFER agent selection area is sorted by domain names. To check the JENNIFER agent list that belongs to other domains, proceed as follows.

Figure 12-10:JENNIFER Agent Selection Area - Domain Selection


JENNIFER
Dashboard
Real-time Monito
JENNIFER Dashboard | Equalizer | User Defined Menu Demo |
Hide
Select Domain
Server 1
✓ Server 2
M22 S51 S52 S61 S62 S71 S72
WK1 WK2
Recent Active Services per Server

- Click the option icon.
- From the contextual menu, select the **[Select Domain]** menu and click a domain.

12.11.6.Node Organization

12.11.6.1.What is a node?

When the JENNIFER server monitors too many JENNIFER agents, you can group the JENNIFER agents performing similar types of work as a tree and monitor the groups. A node is a group of JENNIFER agents. Node organization refers to the grouping of JENNIFER agents as a tree by using the node.

For example, if the JENNIFER server monitors JENNIFER agents W11 through W19, you can group W11 through W15 as the [Groupware] node and W16 through W19 as the [HR] node. Using the [Groupware] and [HR] nodes, you can organize the [Internal System] node.

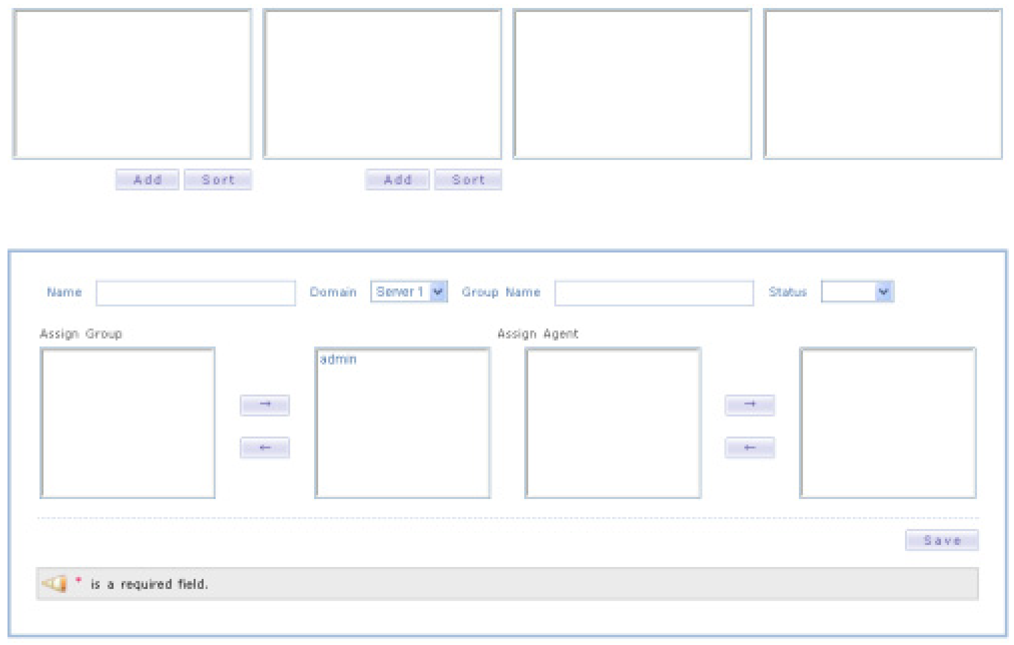
Notice: A Maximum of four node levels are allowed.

12.11.6.2.Node Setting

To organize a node, register the domain and then import the JENNIFER agent.

Warning: When you want to organize a node, you need to register the domain even if there is only one JENNIFER server. The domain to be registered is the JENNIFER server that you are currently using.

Figure 12-11:Node Organization Screen



If you want to add a node, first enter the top node.You can add the top node as follows:

- Select the **[Properties | Node Setting]** menu for the integrated server.
- Click the **[Add]** button under the first node group box.
- After addign the content in the node input form, click the **[Save]** button.

The following is the node input field.

Table 12-8: Node Input Field

Field	Description
Name	Node name

Table 12-8: Node Input Field

Field	Description
Domain	This is the domain for the JENNIFER agent to be assigned to the node. The top node and all lower-ranked nodes should only be set by JENNIFER agents belonging to the same domain. After creating a node, the domain cannot be modified.
Group name	To accurately monitor the number of simultaneous users and visitors, enter the group IDs for the JENNIFER agents that belong to the node.
Status	No node set as inactive is used.
Assign group	This is the group that uses the node. It only appears for users of the group. If the group is assigned to the child node but no group is assigned to the parent node, then the child node will not be shown to users who belong to the group.
Assign agent	Assigns the JENNIFER agent that belongs to the node. If the node is only used to group the child nodes, then you do not need to assign a JENNIFER agent.

If the top node includes all JENNIFER agents monitored by the JENNIFER server, enter TOT for the group name. If you want to accurately display the total number of simultaneous users and visitors based on HTTP cookies in the general node, enter the JENNIFER agent group ID set in the agent_group option of the JENNIFER server in the group name of the node. If the group name is not specified, the total number of simultaneous users and visitors simply becomes the sum of the simultaneous users and visitors for all JENNIFER agents.

```
agent_group = @01:W11,W12; @02:W13,W14
```

The JENNIFER agent group ID starts with @ and ends with a two-digit number. Multiple JENNIFER agent groups are separated by semicolons. The JENNIFER agent group ID and the JENNIFER agents in the group are separated by colons. In addition, the JENNIFER agents in the same JENNIFER agent group are separated by commas. If you change this option, you must restart the JENNIFER server for it to take effect.

Notice: If multiple JENNIFER servers are used to organize a domain, set the agent_group option of the JENNIFER server where the JENNIFER agents exist. You cannot set the agent_group option for JENNIFER agents with different JENNIFER servers.

You can add the child node as follows:

- Select the **[Properties | Node Setting]** menu for the integrated server.
- Select the top node that you want to add the child node.
- Click the **[Add]** button under the node group box.
- After adding the content in the node input form, click the **[Save]** button.

You can change the node as follows:

- Select the **[Properties | Node Setting]** menu for the integrated server.

- Select the top node that you want to change in the node group box.
- After changing the content in the node input form, click the **[Save]** button.

You can delete the node as follows:

- Select the **[Properties | Node Setting]** menu for the integrated server.
- Select the top node that you want to delete in the node group box.
- Click the **[Delete]** button.

You can sort the child nodes belonging to the same top node as follows:

- Select the **[Properties | Node Setting]** menu for the integrated server.
- Select the top node that you want to sort in the node group box.
- Click the **[Sort]** button under the node group box.
- Change the arrangement of node in the sorting pop-up window.

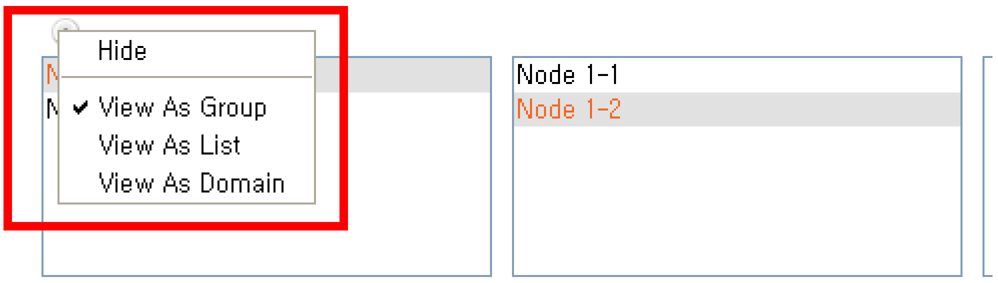
Notice: You can only assign JENNIFER agents with the same domain as the parent node to the child node. You cannot assign JENNIFER agents that are already assigned to another node.

12.11.6.3.Change in the User Interface due to Node Organization

Once you organize a node, you will find that most of the JENNIFER agent selection areas at the top of the user interface screen are different than before.

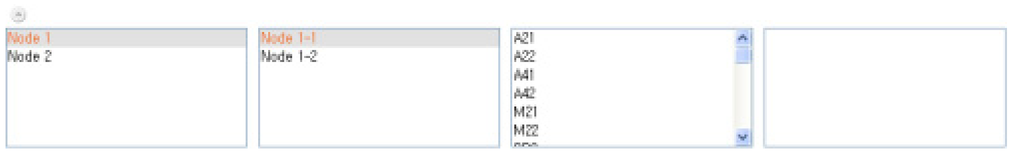
If a node is organized, you can select one of the following views in the JENNIFER selection area.

Figure 12-12:JENNIFER Agent Selection Area - Node Selection



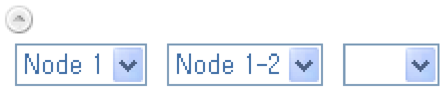
- View as group - Node organization is displayed as a node group area.

Figure 12-13:JENNIFER Agent Selection Area - View as Group



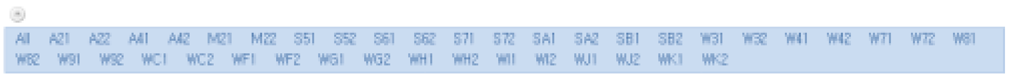
- View as list - Node organization is displayed as a drop-down box.

Figure 12-14:JENNIFER Agent Selection Area - View as List



- View as domain - Regardless of node organization, all JENNIFER agents in a certain domain are displayed.

Figure 12-15:JENNIFER Agent Selection Area - View as Domain


All A21 A22 A41 A42 M21 M22 S51 S52 S61 S62 S71 S72 SA1 SA2 SB1 SB2 W31 W32 W41 W42 W71 W72 W81
W82 W91 W92 WC1 WC2 WF1 WF2 WG1 WG2 WH1 WH2 WI1 WI2 WJ1 WJ2 WK1 WK2

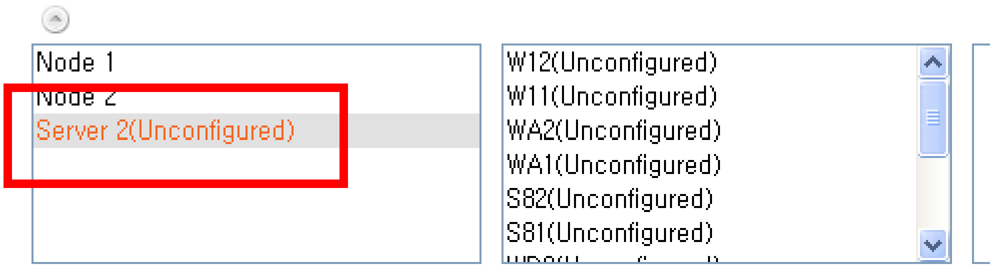
You can change the view of the JENNIFER agent selection area as follows.

- Click the option icon.
- Select a view from the contextual menu.

If you select a certain node in the node group area or drop-down box, the JENNIFER agents in the node and its child nodes are displayed.

If you add new JENNIFER agents, or if certain JENNIFER agents have not been assigned to the node due to a mistake, then the top node with the name [Domain name(unconfigured)] is created for the users of the admin group, and the JENNIFER agents that are not already assigned to a node will be assigned to the top node.

Figure 12-16:Unassigned JENNIFER Agents



12.12. Data Setting

The JENNIFER server saves the data collected from the JENNIFER agents and the JENNIFER independent agents in a file and in the database.

12.12.1. Data File

It is not sufficient to save the X-view transaction and profile data in the database. For this reason, the JENNIFER server stores them in a file.

The directory to save the data files is set using the data_directory option of the JENNIFER server.

```
data_directory = ../../data/file/
```

By default, the data files are saved in the JENNIFER_HOME/data/file directory. Below this directory, sub directories for different dates will be created. The directory structure and the file format can be changed arbitrarily.

Warning: Do not change the data_directory option while the JENNIFER server is operating. It could damage the files. If you want to change this option, you must first stop the JENNIFER server, and then directly change the configuration file of the JENNIFER server.

12.12.2. Database

The JENNIFER server saves most of the performance and configuration data in the database. The database is divided into the performance database, in which the performance data is stored, and the admin database, in which the configuration data is stored.

Notice: By default, the JENNIFER server uses Apache Derby as its database.

If there are high number of the JENNFER agents or TPS, you can improve your system performance by using commercial DBMS. You can use each different DBMS as a performance database and admin database because the admin database does not give any negative impacts on the performance. Therefore, we recommend that you use commercial DBMS for the performance database and Apache Derby for the admin database.

12.12.2.1.Using Apache Derby

The location of the Apache derby database is set in the system.derby.system.home option of the JENNIFER server.

```
system.derby.system.home = ../../data/database
```

By default, the database is stored in the JENNIFER_HOME/data/database directory.

The JDBC connections for using Apache derby at the JENNIFER server are set in the JENNIFER_HOME/server/conf/Catalina/localhost/ROOT.xml file.

The following illustrates the JDBC configuration for the performance database.

```
<Resource name="jdbc/Jennifer" auth="Container"
          type="javax.sql.DataSource" maxActive="100" maxIdle="30"
          maxWait="10000" username="jennifer" password="jennifer"
          driverClassName="org.apache.derby.jdbc.EmbeddedDriver"
          url="jdbc:derby:jennifer;create=true" />
```

The following is the JDBC configuration for the admin database.

```
<Resource name="jdbc/JenniferAdmin" auth="Container"
          type="javax.sql.DataSource" maxActive="100" maxIdle="30"
          maxWait="10000" username="jennifer" password="jennifer"
          driverClassName="org.apache.derby.jdbc.EmbeddedDriver"
          url="jdbc:derby:jenniferadm;create=true" />
```

If you want to access the Apache derby database from an additional Java application, add the following options for the JENNIFER server, and then restart the server to make the changes take effect.

```
system.derby.drda.startNetworkServer = true
system.derby.drda.portNumber = 1527
system.derby.drda.host = 127.0.0.1
```

For Apache derby, you can minimize the wasted table space by using the Reorg scheduler.

```
time_actor_2 = com.javaservice.jennifer.server.timeactor.ReorgActor 03
```

By default, the table space is rearranged at 3 AM everyday.

Notice: By default, Apache derby is set as the database. When you use Apache derby, you do not need to change the configuration.

12.12.2.2.Using Oracle

You must use Oracle 9i or higher. The JDBC driver should be Oracle 10g or higher and the database should be made using UTF-8 encoding.

The JDBC connection for using Oracle at the JENNIFER server is set in the JENNIFER_HOME/server/conf/Catalina/localhost/ROOT.xml file.

The following is the JDBC configuration for the performance database.

```
<Resource name="jdbc/Jennifer" auth="Container"
          type="javax.sql.DataSource" maxActive="100" maxIdle="30"
          maxWait="10000" username="jennifer" password="jennifer"
          driverClassName="oracle.jdbc.driver.OracleDriver"
          url="jdbc:oracle:thin:@127.0.0.1:1521:jennifer" />
```

The following is the JDBC configuration for the admin database.

```
<Resource name="jdbc/JenniferAdmin" auth="Container"
          type="javax.sql.DataSource" maxActive="100" maxIdle="30"
          maxWait="10000" username="jennifer" password="jennifer"
          driverClassName="oracle.jdbc.driver.OracleDriver"
          url="jdbc:oracle:thin:@127.0.0.1:1521:jenniferadm" />
```

Various attributes such as username, password and url should be changed to fit the Oracle database that is to be used.

In addition, you must copy the JDBC driver library(ojdbc14.jar) into the JENNIFER_HOME/server/common/lib directory.

Notice: If the configuration has been changed, restart the JENNIFER server.

12.12.2.3.Using IBM DB2

You must use DB2 9.5 or higher. The database should be made using UTF-8 encoding and the page size should be greater than 32k.

The JDBC connection for using IMD DB2 at the JENNIFER server is set in the JENNIFER_HOME/server/conf/Catalina/localhost/ROOT.xml file.

The following is the JDBC configuration for the performance database.

```
<Resource name="jdbc/Jennifer" auth="Container"
          type="javax.sql.DataSource" maxActive="100" maxIdle="30"
          maxWait="10000" username="jennifer" password="jennifer"
          driverClassName="com.ibm.db2.jcc.DB2Driver"
          url="jdbc:db2://120.0.0.1:50000/jennifer" />
```

The following is the JDBC configuration for the admin database.

```
<Resource name="jdbc/JenniferAdmin" auth="Container"
          type="javax.sql.DataSource" maxActive="100" maxIdle="30"
          maxWait="10000" username="jennifer" password="jennifer"
          driverClassName="com.ibm.db2.jcc.DB2Driver"
          url="jdbc:db2://120.0.0.1:50000/jenniferadm" />
```

Various attributes such as username, password and url should be changed to fit IBM DB2 that is to be used.

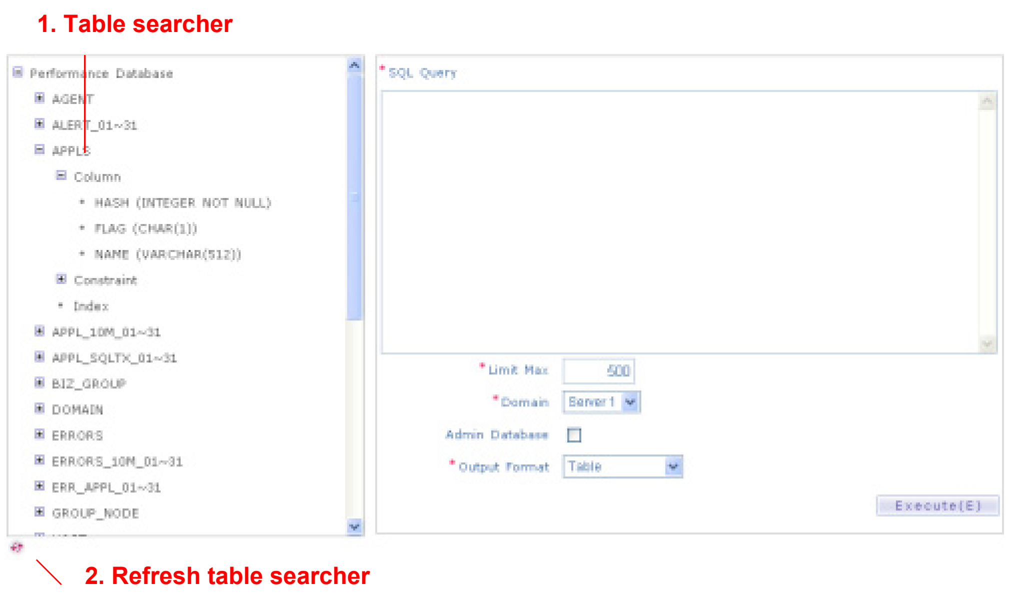
In addition, you must copy the JDBC driver library(db2jcc.jar, db2jcc_license_cu.jar) into the JENNIFER_HOME/server/common/lib directory.

Notice: When the configuration has been changed, restart the JENNIFER server.

12.12.3.Query Tool

By selecting the **[Statistics | Query Tool]** menu, you can execute arbitrary SQL for the JENNIFER database.

Figure 12-17:Query Tool



- Performance Database
  - AGENT
  - ALERT_01~31
  - APPLS
    - Column
      - HASH (INTEGER NOT NULL)
      - FLAG (CHAR(1))
      - NAME (VARCHAR(512))
    - Constraint
    - Index
  - APPL_10M_01~31
  - APPL_SQLTX_01~31
  - BIZ_GROUP
  - DOMAIN
  - ERRORS
  - ERRORS_10M_01~31
  - ERR_APPL_01~31
  - GROUP_NODE

*SQL Query

*Limit Max: 500
*Domain: Server1
Admin Database: ☐
*Output Format: Table

1. Table searcher - Using the table searcher at the left, you can check the table schema for the performance database and the admin database.
2. Refresh table searcher - If you want to refresh the content in the table searcher, click the button.

You can execute arbitrary SQL for the JENNIFER database as follows:

- Enter your query in the SQL field.
- Specify the limit max, the domains, the admin database and the output formats.
- Click the [Execute] button in the lower right corner.

The SQL results are shown at the bottom, The following table describes each field.

Table 12-9: SQL Tool Fields

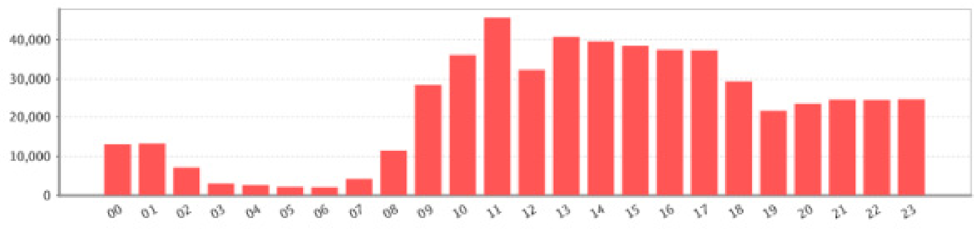
Field	Description
SQL	If you want to execute multiple SQLs at the same time, enter multiple entities separated by semicolons. A line beginning with -- is a comment.
Limit max	The maximum number of records returned when the SELECT text is executed
Domain	Select a domain
Admin database	When activated, it performs the SQL on the admin database. Otherwise, it performs the SQL on the performance database.
Output format	Selects a format for the SQL result

If a table-type output format is selected, the data is displayed for each HTML table. If an excel-type output format is selected, then the data is downloaded to a file in CSV format.

If a bar chart is selected, then a bar chart is displayed. For example, the following SQL searches for the number of hourly visitors. The first column in the SELECT text is the X-coordinate, and the remaining columns are the Y-coordinates. The column names are used as legends.

```
SELECT LOG_HH, SUM(VISIT_HOUR) AS VISIT
  FROM PERF_X_01
 WHERE LOG_DT = '20081001'
   AND AGENT_ID = 'TOT'
 GROUP BY LOG_HH
```

Figure 12-18:Hourly Visitors



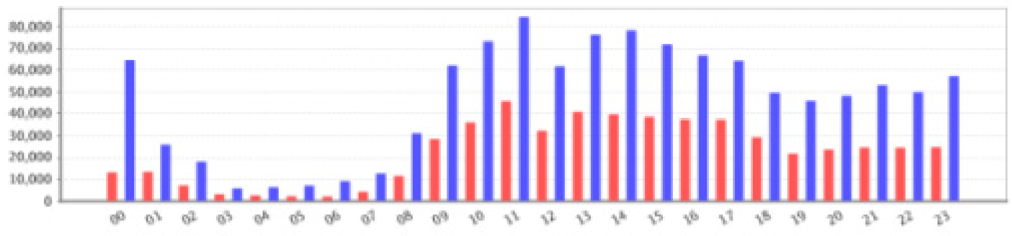
The chart size is set using the report_image_width and report_image_height options of the JENNIFER server.

```
report_image_width = 800
report_image_height = 200
```

If you want to display various types of data for the same time period, add a column in the SELECT text. For instance, if you want to display the number of visitors and hits on a bar chart, use the following SQL:

```
SELECT LOG_HH, SUM(VISIT_HOUR) AS VISIT, SUM(HIT) AS HIT
  FROM PERF_X_01
 WHERE LOG_DT = '20081001'
   AND AGENT_ID = 'TOT'
 GROUP BY LOG_HH
```

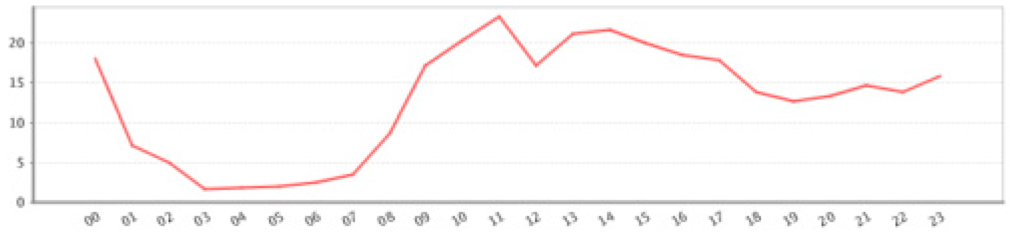
Figure 12-19:Hourly Visitors and Hits



If a line-type output format is selected, then a line chart is displayed. For example, the following SQL is used to obtain the hourly arrival rates. The first column in the SELECT text is the X-coordinate, and the remaining columns are the Y-coordinates. The column names are used as legends.

```
SELECT LOG_HH, AVG(ARRIVAL_RATE) AS ARRIVAL_RATE
  FROM PERF_X_01
 WHERE LOG_DT = '20081001'
   AND AGENT_ID = 'TOT'
 GROUP BY LOG_HH
```

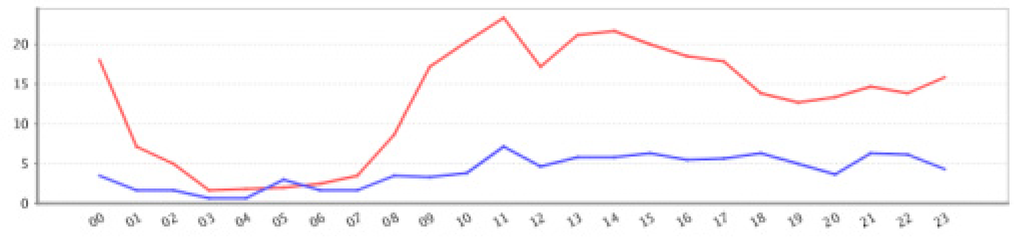
Figure 12-20:Hourly Arrival Rates



If you want to display various types of data for the same time period, add a column in the SELECT text. For instance, if you want to display the arrival rates and the number of active services on a line chart, use the following SQL:

```
SELECT LOG_HH, AVG(ARRIVAL_RATE) AS ARRIVAL_RATE,
        AVG(ACTIVE_SERVICE) AS ACTIVE_SERVICE
  FROM PERF_X_01
 WHERE LOG_DT = '20081001'
   AND AGENT_ID = 'TOT'
 GROUP BY LOG_HH
```

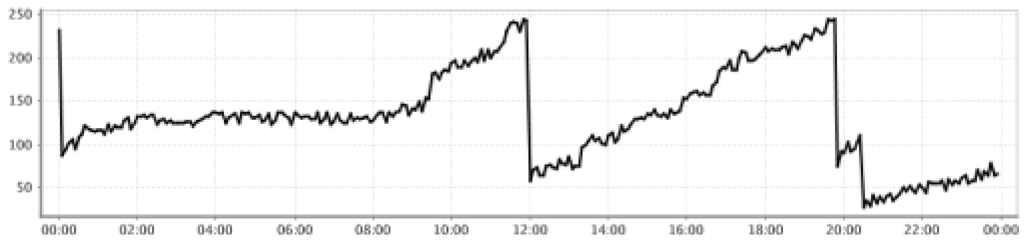
Figure 12-21:Hourly Arrival Rates and Active Services



If a timeline type of output format is selected, then a timeline chart is displayed. for example, the following SQL is used to obtain the five-minute average of Java heap memory utilization. The first column in the SELECT text is the X-coordinate, and the remaining columns are the Y-coordinates. The column names are used as legends.

```
SELECT LOG_DT || LOG_HH || LOG_MM, HEAP_USED
  FROM PERF_X_01
 WHERE LOG_DT ='20081001'
   AND AGENT_ID = 'A21'
```

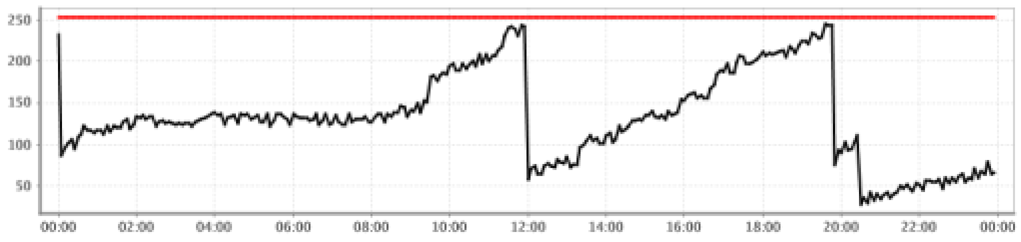
Figure 12-22:Five-Minute Average of Java Heap Memory Utilization



If you want to display various types of data for the same time period, add a column in the SELECT text. For instance, if you want to display the Java heap memory utilization and the total Java heap memory on a timeline chart, use the following SQL:

```
SELECT LOG_DT || LOG_HH || LOG_MM, HEAP_USED, HEAP_TOTAL
  FROM PERF_X_01
 WHERE LOG_DT ='20080801'
   AND AGENT_ID = 'A21'
```

Figure 12-23:Five-Minute Average of Java Heap Memory Utilization and Total Java Heap Memory



Notice: If there is a small amount of data (displayed for each hour), select a bar or line chart as the output format. Otherwise (displayed for every five minutes), select a timeline chart as the output format.

12.12.4.Saving and Deleting the Data

To improve the searching time for the database, JENNIFER uses the SummaryActor scheduler to create a summary of the performance data periodically. Set the Summary-Actor option of the JENNIFER server as follows:

```
time_actor_1=com.javaservice.jennifer.server.timeactor.SummaryActor 01
```

As indicated by the first parameter, 01, the SummaryActor scheduler is executed at 1AM everyday. If it is not executed properly, you can use **[Toolbar Area| Summary Statistics Data]** menu to execute theSummaryActor scheduler on a certain day.

Notice: If SummaryActor is repeatedly executed on a certain day by using the prop_rpt.jsp option, the data wil not be calculated repeatedly.

The increase in the performance data collected by JENNIFER is directly proportional to the number of JENNIFER agents and the workload. Therefore, to optimize disk use, by default any data that was created 30 days ago or more is deleted by the CleanerActor scheduler. The following is the configuration for CleanerActor of the JENNIFER server.

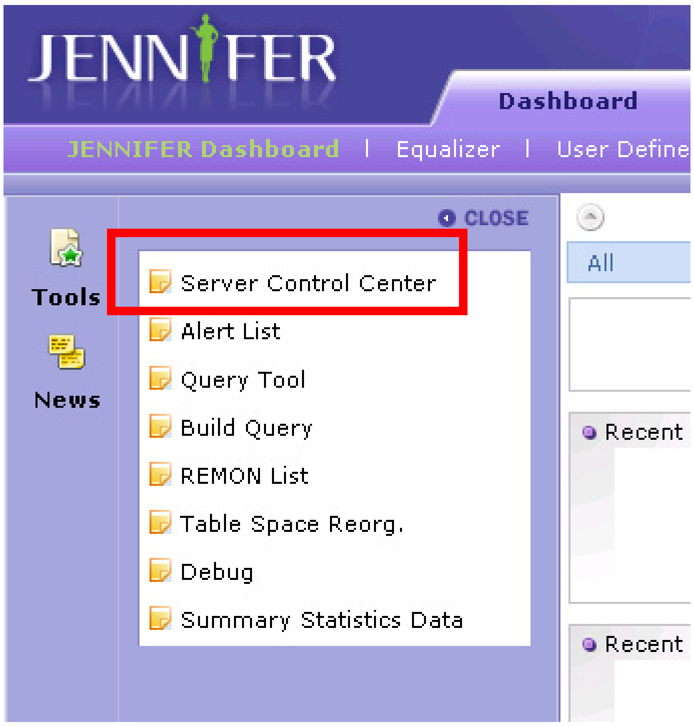
```
time_actor_03
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 ALERT MONTH 1
time_actor_04
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 ERROR  MONTH 1
time_actor_05
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 APPL MONTH 1
time_actor_06
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 SQL MONTH 1
time_actor_07
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 TX MONTH 1
time_actor_08
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 PERF MONTH 1
time_actor_09
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 FILE MONTH 1
time_actor_11
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 REMON DAY 7
```

By default, data that is one month old is deleted from the tables, ALERT_01~31, ERRORS_10M_01~31, APPL_10M_01~31, SQLS_10M_01~31, TX_10M_01~31 and PERF_X_01~31. In addition, seven-day-old data is deleted from the table that stores the data collected by REMON. As well, one-month-old data files are deleted. These tasks are performed consecutively at 2AM.

12.12.5.Backup and Recover the Data

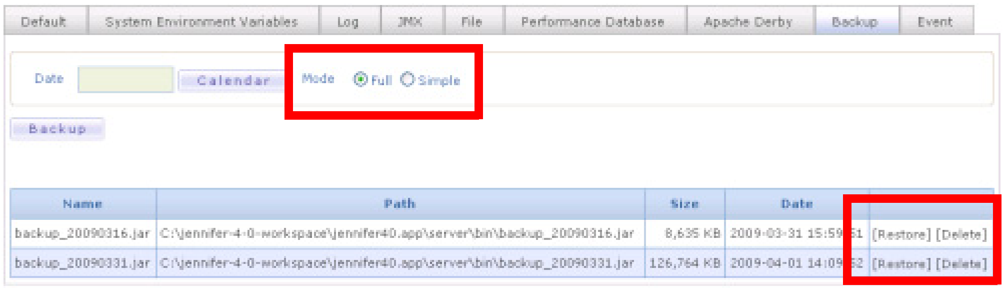
You can backup and restore the data file and database contents based on the date. If you click the **[Tool | Server Control Center]** menu in the tools area, a pop-up window is shown.

Figure 12-24:Server Control Center



If you select the backup tap in the pop-up window, you can execute a backup/restoring process on the page.

Figure 12-25:Backup and Restore the Data



Mode [✓ Full] Simple

Name	Path	Size	Date	
backup_20090316.jar	C:\jennifer-4-0-workspace\jennifer40.app\server\bin\backup_20090316.jar	8,635 KB	2009-03-31 15:59:51	[Restore] [Delete]
backup_20090331.jar	C:\jennifer-4-0-workspace\jennifer40.app\server\bin\backup_20090331.jar	126,764 KB	2009-04-01 14:09:52	[Restore] [Delete]

Specify the location of the backup file using the backup_root option of the JENNIFER server. The option is not set, but in ths case, the backup file is saved in the JENNIFER_HOME/server/bin directory.

```
backup_root = /home/jennifer/backup
```

Warning: If you run the JENNIFER server in the same hardware, you need to specify the backup directory differently by defining the backup_root option per the JENNIFER server.

The data backup process is as follows:

1. Choose the date that you want to back up but, you can not back up the today's data.
2. Choose the **[Mode]**. If you select the **[Full]** mode, JENNIFER can back up all data files and contents in the database. However, if you select the **[Simple]** mode, JENNIFER can back up the only contents in the database.
3. If you click **[Backup]**, data is backed up. It will take a quite long time if you want to back up the large amount of data. The data backup process completed, the data backup file is added on the data backup file list based on the date.

The data restoring process is as follows:

1. Choose the **[Mode]**. If you select the **[Full]** mode, JENNIFER can restore all data files and contents in the database. However, if you select the **[Simple]** mode, JENNIFER can restore the only contents in the database.
2. If you click **[Restore]**, data is stored. It will take a quite long time if you want to back up the large amount of data.

Notice: To minimize the CPU utilization of the JENNIFER server, you can not execute the backup and restore presess at the same time. For instance, you can not back up the other data for the other date while you are in back up/restore process of the data in 2009-04-02. Also, you can not restore the other data for the other date while you are in back up/restore process of the data in 2009-04-04.

The data restoring process completed, confirm the data is restored successfully as follows:

1. Refer to [Remove the arbitrary file of Java Plug-in] and delete the arbitrary file.
2. Log in again.

To delete the data backup fiel is as follows:

1. If you click the **[Delete]** button according to the certain date to delete the data back file on ther list, the backup file is deleted.
2. The data backup file deleting process completed, the data backup file on that certain date is deleted on the list.

In addition, you can back up the data regularly using the BackupActor scheduler. The configuration guide about BackupActor is as follows:

```
time_actor_15 = com.javaservice.jennifer.server.timeactor.BackupActor
0220 FULL
```

Notice: BackupActor scheduler is not set.

In the above configuration, the data backup is processed at 02: 20 as a FULL mode. You can set SIMPLE mode instead of FULL.

Warning: The data backup and restore presess increases the CPU utilization of the JENNIFER server. Therefore, when multiple JENNIFER servers are installed in the same hardware, the running time of BackupActor scheduler should be modified differently.

12.13. Table Schema

Apache derby is used for the data in the table column.

12.13.1. Performance Database

12.13.1.1. AGENT

This table stores the JENNIFER agent data.

Table 12-10: AGENT Table

Column	Type	Restriction	Description
ID	VARCHAR(20)	PK, NOT NULL	JENNIFER agent ID
NAME	VARCHAR(50)		JENNIFER agent name
IP	VARCHAR(50)	NOT NULL	IP adress
TCP_PORT	INTEGER	NOT NULL	The port number set in the agent_tcp_port option of the JENNIFER agent
DOMAIN_ID	VARCHAR(10)	NOT NULL	Domain ID
NODE_ID	BIGINT		Node ID
VERSION	VARCHAR(50)		JENNIFER agent version
TYPE	INTEGER		Type
STATUS	INTEGER	NOT NULL	A status code. 1 means that it is used. 0 means it is not.
SORT	INTEGER		Sorting criteria

12.13.1.2.ALERT_01~31

This daily table stores the alert data.

Notice: When the volum of the performance data is high, the data is saved in the database based on the date. The daily table is commonly used in this case. For example, the alert data collected in 3rd March is saved in the ALERT_03 table and the data collected in 17th October is saved in the ALERT_17 table. ALERT_01~31 is a conceptual name to define the daily table and the table does not exist substantially.

Table 12-11: ALERT_01~31 Table

Column	Type	Restriction	Description
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
LOG_SS	CHAR(2)		Second(SS)
TYPE	CHAR(1)		A alert type code. C means Critical, E means Error, W means Warning.
ALERT_NM	VARCHAR(100)		Alert name
ALERT_MSG	VARCHAR(32672)		Alert message
AGENT_ID	VARCHAR(20)		JENNIFER agent ID including alert

12.13.1.3.APPLS

This table stores the application name.

Table 12-12: APPLS Table

Column	Type	Restriction	Description
HASH	INTEGER	PK, NOT NULL	Hash value of application name
NAME	VARCHAR(512)		Application name
FLAG	CHAR(1)		A code to indicate whether the data is normal. T means normal. F means abnormal.

12.13.1.4.APPL_10M_01~31

This daily table stores the statistical data of application service rate for 10 minutes.

Table 12-13: APPL_10M_01~31 Table

Column	Type	Restriction	Description
APPL_HASH	INTEGER		Hash value of application name
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
CNT	INTEGER		# of calls
FAIL_CNT	INTEGER		# of failure
ELAPSED_SUM	BIGINT		Sum of total response time
ELAPSED2_SUM	BIGINT		A sum of the square of the response time used to calculate the standard deviation.
ELAPSED_MIN	BIGINT		Minimum response time
ELAPSED_MAX	BIGINT		Maximum response time
CPU_SUM	BIGINT		A sum of total CPU time
TPMC_SUM	BIGINT		A sum of TPMC

12.13.1.5.APPL_SQLTX_01~31

This daily table stores the data that maps the application and the SQL or the external transactions for 10 minutes.

Table 12-14: APPL_SQLTX_01~31 Table

Column	Type	Restriction	Description
APPL_HASH	INTEGER		Hash value of application name
SQLTX_HASH	INTEGER		Hash value of SQL or external transaction value
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
TYPE	CHAR(1)		A code to define type. S means SQL and T means extra transaction.
CNT	INTEGER		# of executions
ELAPSED_SUM	BIGINT		A sum of total response time

Table 12-14: APPL_SQLTX_01~31 Table

Column	Type	Restriction	Description
ELAPSED2_SUM	BIGINT		A sum of the square of the response time used to calculate the standard deviation.
ELAPSED_MIN	BIGINT		Minimum response time
ELAPSED_MAX	BIGINT		Maximum response time

12.13.1.6.BIZ_GROUP

This table stores the application business group data.

Table 12-15: BIZ_GROUP Table

Column	Type	Restriction	Description
UUID	BIGINT	PK, NOT NULL	A unique ID that is generated automatically.
ID	VARCHAR(256)	NOT NULL	ID
DIV	VARCHAR(100)	NOT NULL	Division name
NAME	VARCHAR(100)	NOT NULL	Business group name
DOMAIN_ID	VARCHAR(10)		Domain ID
AGENT_ID	VARCHAR(300)		JENNIFER agent ID
APP	VARCHAR(1000)		Application list and it uses (\) as a delimeter.
TX	VARCHAR(1000)		External transaction list and it uses (\) as a delimeter.
TYPE	INTEGER		Type
STATUS	INTEGER		A status code. 1 means that it is active. 0 means it is inactive.
SORT	INTEGER		Sorting criteria

12.13.1.7.DOMAIN

This table stores the domain data.

Table 12-16: DOMAIN Table

Column	Type	Restriction	Description
ID	VARCHAR(10)	PK, NOT NULL	Domain ID
NAME	VARCHAR(20)	NOT NULL	Domain name
IP	VARCHAR(50)	NOT NULL	JENNIFER server’s IP address

Table 12-16: DOMAIN Table

Column	Type	Restriction	Description
HTTP_PORT	INTEGER	NOT NULL	JENNIFER server's HTTP port number
TCP_PORT	INTEGER	NOT NULL	The port number set in the server_tcp_port option of the JENNIFER server
VERSION	VARCHAR(50)		JENNIFER server's version
TYPE	INTEGER		Type
STATUS	INTEGER	NOT NULL	A status code. 1 means that it is active. 0 means it is inactive.
SORT	INTEGER		Sorting criteri

12.13.1.8.ERRORS

This table stores the error name.

Table 12-17: ERRORS Table

Column	Type	Restriction	Description
HASH	INTEGER	PK, NOT NULL	Hash value of error
NAME	VARCHAR(32672)		Error name
FLAG	CHAR(1)		A code to indicate whether the data is normal. T means normal. F means abnormal.

12.13.1.9.ERRORS_10M_01~31

This daily table stores the statistical error data for 10 minutes.

Table 12-18: ERRORS_10M_01~31 Table

Column	Type	Restriction	Description
ERR_HASH	INTEGER		Hash value of error
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
TYPE	CHAR(1)		A alert type code. C means Critical, E means Error, W means Warning.
CNT	INTEGER		# of errors
ERROR_CD	INTEGER		Error code

Table 12-18: ERRORS_10M_01~31 Table

Column	Type	Restriction	Description
ERROR_NM	VARCHAR(100)		Error name

12.13.1.10.ERR_APPL_01~31

This table stores the data that maps the application and the error.

Table 12-19: ERR_APPL_01~31 Table

Column	Type	Restriction	Description
ERR_HASH	INTEGER		Hash value of error
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
CNT	INTEGER		# of errors
APPL_HASH	INTEGER		Hash value of aplication that has occured error.
MSG_HASH	INTEGER		Hash value of error message
TX_UUID	BIGINT		X-View transaction ID

12.13.1.11.GROUP_NODE

This table stores the data that maps the group and the node.

Table 12-20: GROUP_NODE Table

Column	Type	Restriction	Description
GROUP_ID	VARCHAR(20)	PK, NOT NULL	Group ID
NODE_ID	BIGINT	PK, NOT NULL	Node ID

12.13.1.12.HOST

This table stores the data of computer that is executing WMOND.

Table 12-21: HOST Table

Column	Type	Restriction	Description
HOST_ID	VARCHAR(20)		Host ID
HOST_NM	VARCHAR(50)		Host name

Table 12-21: HOST Table

Column	Type	Restriction	Description
IPADDR	VARCHAR(20)	NOT NULL	Host IP address

12.13.1.13.METHODS

This table stores the method name.

Table 12-22: METHODS Table

Column	Type	Restriction	Description
HASH	INTEGER	PK, NOT NULL	Hash value of method name
NAME	VARCHAR(32672)		Method name
FLAG	CHAR(1)		A code to indicate whether the data is normal. T means normal. F means abnormal.

12.13.1.14.NODE

This table stores the node data.

Table 12-23: NODE Table

Column	Type	Restriction	Description
ID	BIGINT	PK, NOT NULL	Node ID
NAME	VARCHAR(50)	NOT NULL	Node name
GROUP_ID	VARCHAR(2)		The port number set in the agent_group option of the JENNIFER server
PARENT_ID	BIGINT		Top node ID
DOMAIN_ID	VARCHAR(10)		Domain ID
TYPE	INTEGER		Type
STATUS	INTEGER	NOT NULL	A status code. 1 means that it is active. 0 means it is inactive.
SORT	INTEGER		Sorting criteri

12.13.1.15.PERF_HOST

This table stores the CPU data collected by WMOND.

The storage period is five minutes. You can change it by setting the perf_host_update_interval option of the JENNIFER server. It is expressed in millisec-onds and 300000 is default..

```
perf_host_update_interval = 300000
```

Table 12-24: PERF_HOST Table

Column	Type	Restriction	Description
IPADDR	VARCHAR(20)		Host IP address
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
LAST_TM	CHAR(6)		Last time(HHMM)
CPU_USER	REAL		User application CPU utilization
CPU_SYS	REAL		Kernel CPU utilization
CPU_NICE	REAL		NICE CPU utilization
CPU_IOWAIT	REAL		IO CPU utilization
CPU_IDLE	REAL		IDLE CPU utilization

12.13.1.16.PERF_X_01~31

This daily table stores the general performance data for 5 munites.

You can change it by setting the perf_x_update_interval option of the JENNIFER server. It is expressed in milliseconds and the default is 300000.

```
perf_x_update_interval = 300000
```

Table 12-25: PERF_X_01~31 Table

Column	Type	Restriction	Description
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
LAST_TM	CHAR(6)		Last time(HHMM)
CONCURRENT_USER	REAL		# of concurrent users
ACTIVE_USER	REAL		# of active users

Table 12-25: PERF_X_01~31 Table

Column	Type	Restriction	Description
ACTIVE_SERVICE	REAL		# of active services
ARRIVAL_RATE	REAL		Arrival rate
SERVICE_RATE	REAL		Service rate
RESPONSE_TIME	REAL		Average response time
THINKTIME	REAL		Think time
HEAP_TOTAL	REAL		Total Java heap memory
HEAP_USED	REAL		Java heap memory utilization
SYS_CPU	REAL		System CPU utilization
JVM_CPU	REAL		Java process CPU utilization
SYS_MEM_USED	REAL		System memory utilization
JVM_NAT_MEM	REAL		Java process memory utilization
JDBC_ACTIVE	REAL		# of active JDBC connections
JDBC_ALLOC	REAL		# of allocated JDBC connections
JDBC_IDLE	REAL		# of idle JDBC connections
HIT	INTEGER		# of hits
VISIT_DAY	INTEGER		# of visit users(per day)
VISIT_HOUR	INTEGER		# of visit users(per hour)

12.13.1.17.SQLS

This table stores the SQL.

Table 12-26: SQLS Table

Column	Type	Restriction	Description
HASH	INTEGER	PK, NOT NULL	Hash value of SQL
NAME	CLOB		SQL
FLAG	CHAR(1)		A code to indicate whether the data is normal. T means normal. F means abnormal.

12.13.1.18.SQLS_10M_01~31

This daily table stores the statistical data of SQL execution for 10 minutes.

Table 12-27: SQLS_10M_01~31 Table

Column	Type	Restriction	Description
SQL_HASH	INTEGER		Hash value of SQL
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
CNT	INTEGER		# of execution cases
ELAPSED_SUM	BIGINT		A sum of total response time
ELAPSED2_SUM	BIGINT		A sum of the square of the response time used to calculate the standard deviation.
ELAPSED_MIN	BIGINT		Minimum response time
ELAPSED_MAX	BIGINT		Maximum response time
PARAM1	VARCHAR(32672)		Consistant parameter
PARAM2	VARCHAR(32672)		Binding parameter

12.13.1.19.S_ALERT

This table stores the alert statistical data for one hour.

Table 12-28: S_ALERT

Column	Type	Restriction	Description
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
TYPE	CHAR(1)		A alert type code. C means Critical, E means Error, W means Warning.
ALERT_NM	VARCHAR(100)		Alert name
AGENT_ID	VARCHAR(20)		JENNIFER agent ID including alert
CNT	INTEGER		# of alerts

12.13.1.20.S_APPL

This table stores the application processing statistical data for one day.

Notice: The table whose name starts with S_ is a statistics table managed by SummaryActor.

Table 12-29: S_APPL Table

Column	Type	Restriction	Description
APPL_HASH	INTEGER		Hash value of application
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
CNT	INTEGER		# of hits
FAIL_CNT	INTEGER		# of failures
ELAPSED_SUM	BIGINT		A sum of total response time
ELAPSED2_SUM	BIGINT		A sum of the square of the response time used to calculate the standard deviation.
ELAPSED_MIN	BIGINT		Minimum response time
ELAPSED_MAX	BIGINT		Maximum response time
CPU_SUM	BIGINT		A sum of CPU time
TPMC_SUM	BIGINT		A sum of TPMC

12.13.1.21.S_ERRORS

This table stores the statistical error data for an hour.

Table 12-30: S_ERRORS Table

Column	Type	Restriction	Description
ERR_HASH	INTEGER		Hash value of error
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
TYPE	CHAR(1)		A alert type code. C means Critical, E means Error, W means Warning.
ERROR_CD	INTEGER		Error code
ERROR_NM	VARCHAR(100)		Error name
CNT	INTEGER		# of errors

12.13.1.22.S_PERF_X

This table stores the general performance data for an hour.

Table 12-31: S_PERF_X Table

Column	Type	Restriction	Description
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
CONCURRENT_USER	REAL		# of concurrent users
ACTIVE_USER	REAL		# of active users
ACTIVE_SERVICE	REAL		# of active services
ARRIVAL_RATE	REAL		Arrival rate
SERVICE_RATE	REAL		Service rate
RESPONSE_TIME	REAL		Average response time
THINKTIME	REAL		Think time
HEAP_TOTAL	REAL		Total Java heap memory
HEAP_USED	REAL		Java heap memory utilization
SYS_CPU	REAL		System CPU utilization
JVM_CPU	REAL		Java process CPU utilization
SYS_MEM_USED	REAL		System memory utilization
JVM_NAT_MEM	REAL		Java process memory utilization
JDBC_ACTIVE	REAL		# of active JDBC connections
JDBC_ALLOC	REAL		# of allocated JDBC connections
JDBC_IDLE	REAL		# of idle JEBC connections
HIT	INTEGER		# of hits
VISIT_DAY	INTEGER		# of visit users(per day)
VISIT_HOUR	INTEGER		# of visit users(per hour)

12.13.1.23.S_SQLS

This table stores the statistical data of SQL execution for an hour.

Table 12-32: S_SQLS Table

Column	Type	Restriction	Description
SQL_HASH	INTEGER		Hash value of SQL
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)

Table 12-32: S_SQLS Table

Column	Type	Restriction	Description
CNT	INTEGER		# of execution cases
ELAPSED_SUM	BIGINT		A sum of total response time
ELAPSED2_SUM	BIGINT		A sum of the square of the response time used to calculate the standard deviation.
ELAPSED_MIN	BIGINT		Minimum response time
ELAPSED_MAX	BIGINT		Maximum response time

12.13.1.24.S_TX

This table stores the statistical data of daily external transaction execution.

Table 12-33: S_TX Table

Column	Type	Restriction	Description
TX_HASH	INTEGER		Hash value of external transaction
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
CNT	INTEGER		# of execution cases
ELAPSED_SUM	BIGINT		A sum of total response time
ELAPSED2_SUM	BIGINT		A sum of the square of the response time used to calculate the standard deviation.
ELAPSED_MIN	BIGINT		Minimum response time
ELAPSED_MAX	BIGINT		Maximum response time

12.13.1.25.TXNAMES

This table stores the external transaction name.

Table 12-34: TXNAMES Table

Column	Type	Restriction	Description
HASH	INTEGER	PK, NOT NULL	Hash value of external transaction
NAME	VARCHAR(32672)		Name of external transaction
FLAG	CHAR(1)		A code to indicate whether the data is normal. T means normal. F means abnormal.

12.13.1.26.TX_10M_01~31

This daily table stores the data of external transaction execution for 10 minutes.

Table 12-35: TX_10M_01~31 Table

Column	Type	Restriction	Description
TX_HASH	INTEGER		Hash value of external transaction
AGENT_ID	VARCHAR(20)		JENNIFER agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
CNT	INTEGER		# of execution cases
ELAPSED_SUM	BIGINT		A sum of total response time
ELAPSED2_SUM	BIGINT		A sum of the square of the response time used to calculate the standard deviation.
ELAPSED_MIN	BIGINT		Minimum response time
ELAPSED_MAX	BIGINT		Maximum response time

12.13.2.Admin Database

12.13.2.1.ADF

This table stores the charts organizing the user defined dashboard and the relevant data.

Table 12-36: ADF Table

Column	Type	Restriction	Description
UUID	VARCHAR(50)	PK, NOT NULL	A unique ID that is generated automatically.
VIEW_ID	VARCHAR(10)	NOT NULL	Menu ID of user-defined dashboard
CHART_ID	VARCHAR(50)	NOT NULL	Chart ID
LEFT_POSITION	INTEGER		The X-coordinate for the chart. The closer it is to zero, the further to the left the chart is.
TOP_POSITION	INTEGER		The Y-coordinate for the chart. The closer it is to zero, the further to the top the chart is.
WIDTH	INTEGER		Chart width

Table 12-36: ADF Table

Column	Type	Restriction	Description
HEIGHT	INTEGER		Chart height
CONFIG	VARCHAR(1000)		Chart option

12.13.2.2.AUTH

This table stores the authority data.

Table 12-37: AUTH Table

Column	Type	Restriction	Description
AUTH_ID	VARCHAR(20)	PK, NOT NULL	Authority ID
AUTH_NM	VARCHAR(200)		Authority name
AUTH_DESC	VARCHAR(2000)		Authority description
CONFIG	VARCHAR(1000)		Authority setting information
IS_FIXED	CHAR(1)		Column that is not used

12.13.2.3.CONTENTS

This table stores the BBS data.

Table 12-38: CONTENTS Table

Column	Type	Restriction	Description
BBS_ID	VARCHAR(20)	NOT NULL	BBS type
THREAD_ID	INTEGER	NOT NULL	BBS ID
PARENT_ID	INTEGER		BBS ID that posted a reply
TITLE	VARCHAR(500)		Title
BODY	CLOB		Content
DOC_TYPE	CHAR(1)		Column that is not used
HIT_CNT	INTEGER		# of hits
USER_ID	VARCHAR(20)		User ID that registered the posting
USER_NM	VARCHAR(50)		Column that is not used
PASSWORD	VARCHAR(20)		Column that is not used
STATUS_CD	CHAR(1)		Status code. 0 means N/A. 1 means service request. 2 means processing. 3 means done.

Table 12-38: CONTENTS Table

Column	Type	Restriction	Description
ASSIGNED_USER	VARCHAR(50)		The person who processes it
NEXT_CHECK_DT	VARCHAR(10)		Requested date of processing
CREATE_DT	VARCHAR(10)		Date when the posting is created
CREATE_TM	VARCHAR(8)		Time when the posting is created
UPDATE_DT	VARCHAR(10)		Date when the posting is updated
UPDATE_TM	VARCHAR(8)		Time when the posting is updated

12.13.2.4.EVENT_LOG

This table stores the event log data.

Table 12-39: EVENT_LOG Table

Column	Type	Restriction	Description
UUID	BIGINT	PK, NOT NULL	A unique ID that is generated automatically.
ID	VARCHAR(20)	NOT NULL	Event log ID
EVENT_LOG_DESC	VARCHAR(2000)		Event log description
USER_ID	VARCHAR(20)		User ID
CLIENT_IP	VARCHAR(50)		Client IP
CREATE_DT	CHAR(10)		Created date(YYYYMMDD)
CREATE_TM	CHAR(8)		Created time(HHMMSS)

12.13.2.5.GROUP_AUTH

This table stores the data that maps the group and the authority.

Table 12-40: GROUP_AUTH Table

Column	Type	Restriction	Description
AUTH_ID	VARCHAR(20)	PK, NOT NULL	Authority ID
GROUP_ID	VARCHAR(20)	PK, NOT NULL	Group ID

12.13.2.6.GROUP_MENU

This table stores the data that maps the group and the menu.

Table 12-41: GROUP_MENU Table

Column	Type	Restriction	Description
GROUP_ID	VARCHAR(20)	PK, NOT NULL	Group ID
MENU_ID	INTEGER	PK, NOT NULL	Menu ID

12.13.2.7.LOGIN_USER

This table stores the user data.

Table 12-42: LOGIN_USER Table

Column	Type	Restriction	Description
USER_ID	VARCHAR(20)	PK, NOT NULL	User ID
GROUP_ID	VARCHAR(20)	NOT NULL	Group ID belonging to a user
PASSWD	VARCHAR(50)		Password
NAME	VARCHAR(100)		User name
EMAIL	VARCHAR(100)		Email
COMPANY	VARCHAR(100)		Company
DEPT	VARCHAR(100)		Department
JOBTITLE	VARCHAR(100)		Job title
PHONE	VARCHAR(50)		Phone number
CELLPHONE	VARCHAR(50)		Mobile phone
LAST_IP	VARCHAR(20)		IP address that logged in at the last time
LAST_TM	CHAR(8)		The last logged-in time
LAST_DT	CHAR(10)		The last logged-in date
LOGIN_CNT	INTEGER		# of Logins
DEFAULT_MENU_ID	INTEGER		Initial menu ID
CONFIG	VARCHAR(1000)		User's option

12.13.2.8.MENU

This table stores the menu data.

Table 12-43: MENU Table

Column	Type	Restriction	Description
MENU_ID	INTEGER	PK, NOT NULL	Menu ID
PARENT_ID	INTEGER	NOT NULL	Parent menu ID
MENU_NM_CD	VARCHAR(1000)		Multi-language code
MENU_MN	VARCHAR(100)		Menu name
MENU_ORDER	INTEGER		Menu order
TYPE	CHAR(1)		A code to define menu type. U means URL link menu, O means URL pop-up menu, P means user-defined dashboard, S means report menu.
IS_FIXED	CHAR(1)		Column that is not used
MENU_PROP	VARCHAR(200)		URL
DEFAULT_CHILD_ID	INTEGER		Default child menu
COLUMN_CNT	INTEGER		Column that is not used
DEFAULT_HEIGHT	INTEGER		Column that is not used
DEFAULT_WIDTH	INTEGER		Column that is not used
CONFIG	VARCHAR(1000)		Menu option

12.13.2.9.MESSAGE

This table stores the multi-language data.

Table 12-44: MESSAGE Table

Column	Type	Restriction	Description
ID	BIGINT	PK, NOT NULL	A unique ID that is generated automatically.
MSG_KEY	VARCHAR(50)	NOT NULL	Message key
VALUE	VARCHAR(1000)	NOT NULL	Message value
LOCALE	VARCHAR(5)	NOT NULL	Language
STATUS	INTEGER		A status code. 1 means that it is active. 0 means it is inactive.

12.13.2.10.PASSWORD_HISTORY

This table stores the past password data.

Table 12-45: PASSWORD_HISTORY Table

Column	Type	Restriction	Description
UUID	BIGINT	PK, NOT NULL	A unique ID that is generated automatically.
PASSWD	VARCHAR(50)	NOT NULL	Password
USER_ID	VARCHAR(20)	NOT NULL	User ID
CREATE_DT	CHAR(10)		Created date(YYYYMMDD)
CREATE_TM	CHAR(8)		Created time(HHMMSS)

12.13.2.11.REPORT_TMPL

This table stores the report template data.

Table 12-46: REPORT_TMPL Table

Column	Type	Restriction	Description
TMPL_ID	INTEGER	PK, NOT NULL	Report template ID
TMPL_NAME	VARCHAR(200)		Report template title
USER_ID	VARCHAR(20)		User ID
USER_NM	VARCHAR(50)		User name
DEFAULT_PARAM	VARCHAR(2000)		Default parameter
HEADER	VARCHAR(1000)		Header
FOOTER	VARCHAR(1000)		Footer
COPYRIGHT	VARCHAR(1000)		Copyright
CREATE_DT	CHAR(8)		Created date
CREATE_TM	CHAR(6)		Created time

12.13.2.12.TMPL_ITEM

This table stores the report template item data.

Table 12-47: TMPL_ITEM Table

Column	Type	Restriction	Description
ITEM_ID	INTEGER	NOT NULL	Item ID
ITEM_NM	INTEGER	NOT NULL	Item name
ITEM_TYPE	VARCHAR(20)		Item type
ITEM_PROP	VARCHAR(32672)		Item parameter

Table 12-47: TMPL_ITEM Table

Column	Type	Restriction	Description
TMPL_ID	INTEGER	NOT NULL	Report template ID
CHAPTER	INTEGER		Chapter
SECTION	INTEGER		Section
TITLE	VARCHAR(500)		Same data is input like ITEM_NM column
HEADING_TEXT	VARCHAR(32672)		Header
TAIL_TEXT	CLOB		Column that is not used
LOOPING_KEY	VARCHAR(32672)		Looping parameter
RESERVED_PROP	VARCHAR(32672)		Column that is not used
SQL_QUERY	VARCHAR(32672)		SQL
SQL_PARAM	VARCHAR(32672)		Column that is not used

12.13.2.13.UPLOAD_FILE

This table stores the uploaded file data in the BBS.

Table 12-48: UPLOAD_FILE Table

Column	Type	Restriction	Description
THREAD_ID	INTEGER	NOT NULL	BBS ID
SUBDIR	VARCHAR(20)		The directory saving an uploaded file
FILENAME	VARCHAR(500)	NOT NULL	Uploaded file name
FILE_SZ	BIGINT		Uploaded file size

12.13.2.14.USER_AUTH

This table stores the data that maps the user and the authority.

Table 12-49: USER_AUTH Table

Column	Type	Restriction	Description
AUTH_ID	VARCHAR(20)	PK, NOT NULL	Authority ID
USER_ID	VARCHAR(20)	PK, NOT NULL	User ID

12.13.2.15.USER_GROUP

This table stores the group data.

Table 12-50: USER_GROUP Table

Column	Type	Restriction	Description
GROUP_ID	VARCHAR(20)	PK, NOT NULL	Group ID
GROUP_NM	VARCHAR(100)		Group name
DEFAULT_MENU_ID	INTEGER		Group initial menu ID
GROUP_DESC	VARCHAR(2000)		Group description
AGENT_LIST	VARCHAR(2000)		JENNIFER agent ID that is monitored by a user belonging to this agent group and uses (,) as a delimeter.

12.13.2.16.USER_MENU

This table stores the data that maps the user and the menu.

Table 12-51: USER_MENU Table

Column	Type	Restriction	Description
USER_ID	VARCHAR(20)	PK, NOT NULL	User ID
MENU_ID	INTEGER	PK, NOT NULL	Menu ID

12.13.2.17.USER_PREFERENCE

This table stores the each user’s settign data.

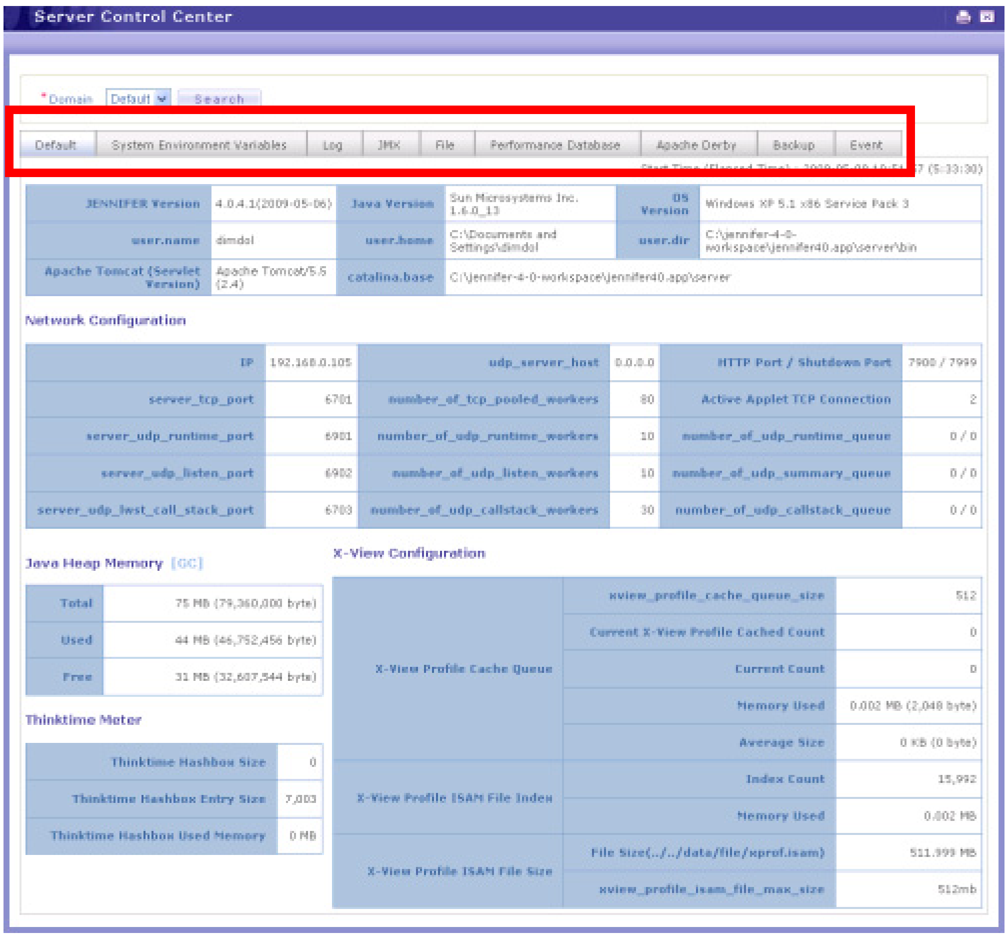
Table 12-52: USER_PREFERENCE Table

Column	Type	Restriction	Description
NAME	VARCHAR(50)	PK, NOT NULL	Setting name
VALUE	VARCHAR(255)	NOT NULL	Setting value
USER_ID	VARCHAR(20)	PK, NOT NULL	User ID

12.14.Server Control Center

Server Control Center is a pop-up window to display the Jennifer server’s status and it allows you to manage the JENNIFER server in the **[Toolbar | Tool| Server Control Center]** menu.

Figure 12-26:Server Control Center



Server Control Center

*Domain Default Search

Default | System Environment Variables | Log | JMX | File | Performance Database | Apache Derby | Backup | Event

Start Time (Elapsed Time) : 2009-05-09 10:5[illegible]7 (5:33:30)

JENNIFER Version	4.0.4.1(2009-05-06)	Java Version	Sun Microsystems Inc. 1.6.0_13	OS Version	Windows XP 5.1 x86 Service Pack 3
user.name	dimdol	user.home	C:\Documents and Settings\dimdol	user.dir	C:\jennifer-4-0-workspace\jennifer40.app\server\bin
Apache Tomcat (Servlet Version)	Apache Tomcat/5.5 (2.4)	catalina.base	C:\jennifer-4-0-workspace\jennifer40.app\server		

Network Configuration

IP	192.168.0.105	udp_server_host	0.0.0.0	HTTP Port / Shutdown Port	7900 / 7999
server_tcp_port	6701	number_of_tcp_pooled_workers	80	Active Applet TCP Connection	2
server_udp_runtime_port	6901	number_of_udp_runtime_workers	10	number_of_udp_runtime_queue	0 / 0
server_udp_listen_port	6902	number_of_udp_listen_workers	10	number_of_udp_summary_queue	0 / 0
server_udp_lwst_call_stack_port	6703	number_of_udp_callstack_workers	30	number_of_udp_callstack_queue	0 / 0

Java Heap Memory [GC]

Total	75 MB (79,360,000 byte)
Used	44 MB (46,752,456 byte)
Free	31 MB (32,607,544 byte)

Thinktime Meter

Thinktime Hashbox Size	0
Thinktime Hashbox Entry Size	7,003
Thinktime Hashbox Used Memory	0 MB

X-View Configuration

X-View Profile Cache Queue	xview_profile_cache_queue_size	512
	Current X-View Profile Cached Count	0
	Current Count	0
	Memory Used	0.002 MB (2,048 byte)
	Average Size	0 KB (0 byte)
X-View Profile ISAM File Index	Index Count	15,992
	Memory Used	0.002 MB
X-View Profile ISAM File Size	File Size(../../data/file/xprof.isam)	511.999 MB
	xview_profile_isam_file_max_size	512mb

Server Control Center has 9 tap menus as follows:

12.14.1.Default

The Default menu has the data as follows:

- You can see the default data about the JENNIFER server, Java, OS, etc.
- You can see the JENNIFER server network configurations.
- You can see the Java heap memory utilization of the JENNIFER server. If you wan to execute the Java garbage collection for the JENNIFER server, click the **[GC]** button.
- You can see the data used in calculating the think time.
- Configurations and processing status of the X-View graph.

12.14.2.System Environment Variables

The System Environment Variables menu has the data as follows:

- Java environment variables
- JENNIFER server option
- Servlet context data

12.14.3.Log

The Log menu displays log file lists of the JENNIFER server. You can download and delete the log file in the list .

Notice: You can not delete the today's log file.

12.14.4.JMX

The JMX menu has the data as follows:.

- Java JMX data
- Apache Tomcat JMX data

If you click the JMT name in the list, you can see the details.

Notice: This JMX tap menu is provided when you only install the JENNIFER on the Java 1.5 or higher.

12.14.5.File

The File menu has the data as follows:

- Option configurations of the JENNIFER server(Related to the data file)
- File list and size of each directory
- You can see the total table size at the botton of the window.

12.14.6.Performance Database

The Performance Database menu has the data as follows:

- DBMS and JDBC driver data related to the performance database
- The name of java.sql.Connection class method having return value and its method return value

12.14.7.Apache Derby

The Apache Derby menu has the data as follows:

- Configurations related to the Apache Derby and the file directory path of the database
- You can see the every table size and the number of data except for the system table. However, as for the daily table, you will see the only the data about yesterday's table in order to prevent performance degradation. If you check the certain data about the certain day's table, click the **[View]** button.
- You can see the total table size at the botton of the window.

Notice: This Apache Derby menu is provided when you only use the Apache Derby as a performance database.

12.14.8.Backup

You can backup and recover the data in the Backup menu. If you want to see more details, refer to the Backup and Recover the Data part.

12.14.9.Event

The Event menu displays the event list. If you want to see more details, refer to the Event Log Record part.

17

Performance Status and Reports

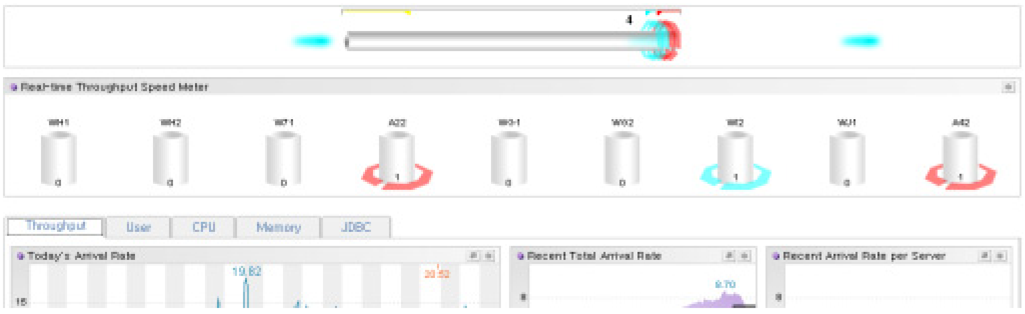
This chapter describes methods for monitoring and analyzing the diverse performance data collected by the JENNIFER agent through the [**Real-time Monitoring | Real-time Status], [Real-time Monitoring | Application], [Statistics | Statistics Status]** and **[Statistics | Application]** menus.

In addition, it also describes various report templates that are provided in the [Statistics | Report] menu.

13.1. Performance Status

From the **[Real-time Monitoring | Real-time Status]** menu, you can monitor today's general performance data using various charts.

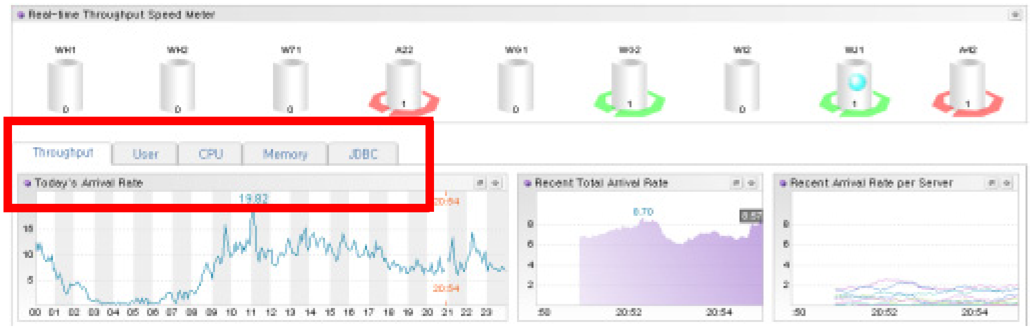
Figure 13-1:Real-time Status (1)


Real-time Throughput Speed Meter
WH1
WH2
W71
A22
W91
W92
W2
WJ1
A42
Throughput
User
CPU
Memory
JDBC
Today's Arrival Rate
Recent Total Arrival Rate
Recent Arrival Rate per Server

Using the speedbar and the speedmeter at the top of the screen, you can intuitively monitor the real-time load of the Java application.

By clicking the various tabs in the middle of the screen, you can check today's general performance data, including throughput, users, CPU, memory and JDBC.

Figure 13-2:Real-time Status (2)

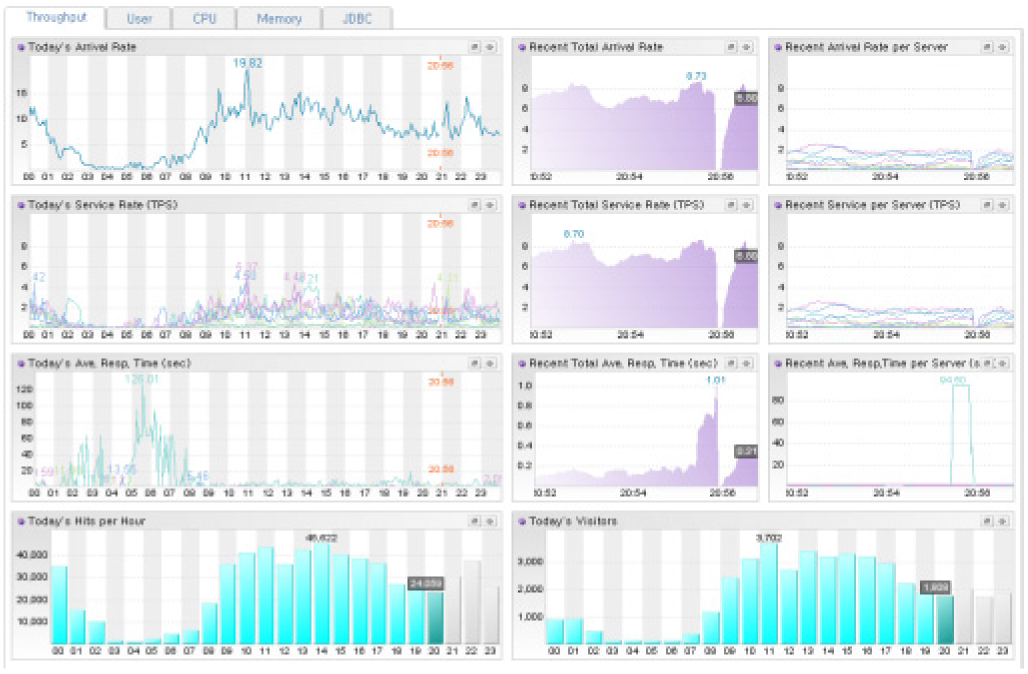

Real-time Throughput Speed Meter
WH1
WH2
W71
A22
W91
W92
W2
WJ1
A42
Throughput
User
CPU
Memory
JDBC
Today's Arrival Rate
Recent Total Arrival Rate
Recent Arrival Rate per Server

Notice: The real-time status mostly uses equalizer and runtime line charts..

13.1.1.Throughput

The throughput tab shows today's arrival rate, service rate, average response time, hits per hour and visitors per hour in the form of a runtime line chart and a bar chart.

Figure 13-3:Real-time Status - Throughput Tab

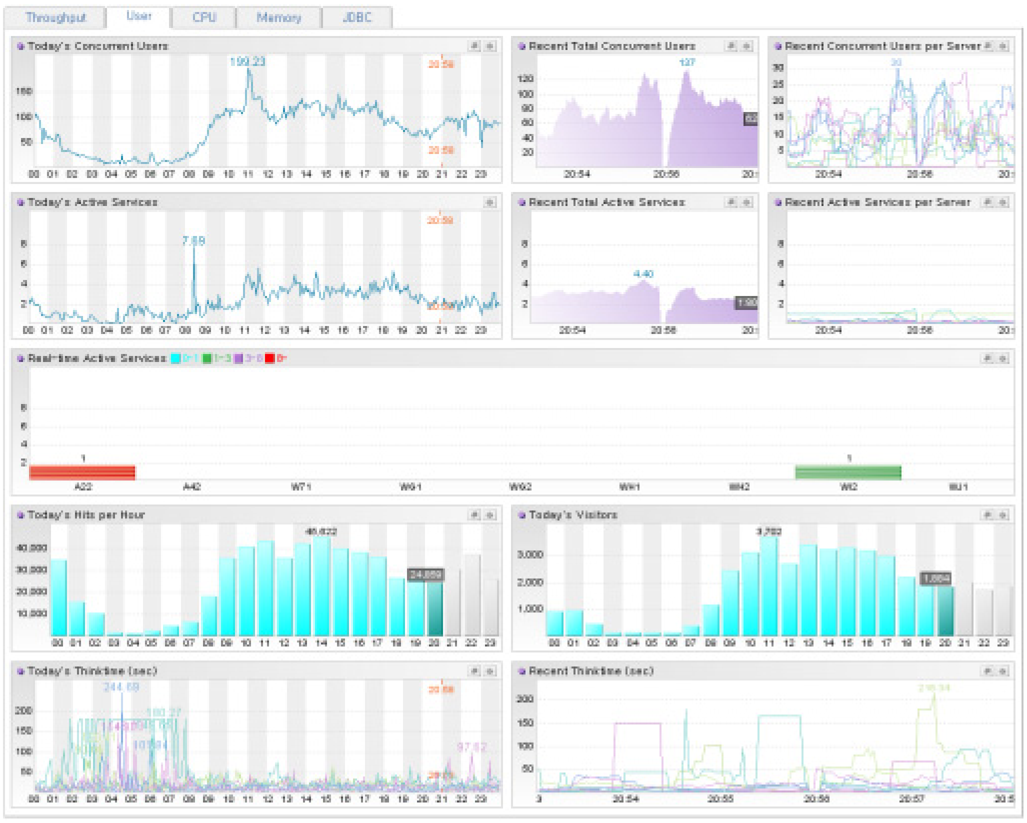

Throughput
User
CPU
Memory
JDBC
Today's Arrival Rate
Recent Total Arrival Rate
Recent Arrival Rate per Server
Today's Service Rate (TPS)
Recent Total Service Rate (TPS)
Recent Service per Server (TPS)
Today's Ave. Resp. Time (sec)
Recent Total Ave. Resp. Time (sec)
Recent Ave. Resp.Time per Server
Today's Hits per Hour
Today's Visitors

13.1.2.User

The user tab shows today's concurrent users, active services, think time, visitors per hour, hits per hour in the form of a equilizer chart, a runtime line chart, a line chart and a bar chart.

Notice: The hourly hits are related to the throughput, while the number of hourly visitors is related to the number of users. However, to balance the horizontal configuration of the bar chart, the hourly hits and the hourly visitors are shown in both the throughput and user tabs.

Figure 13-4:Real-time Status - User Tab

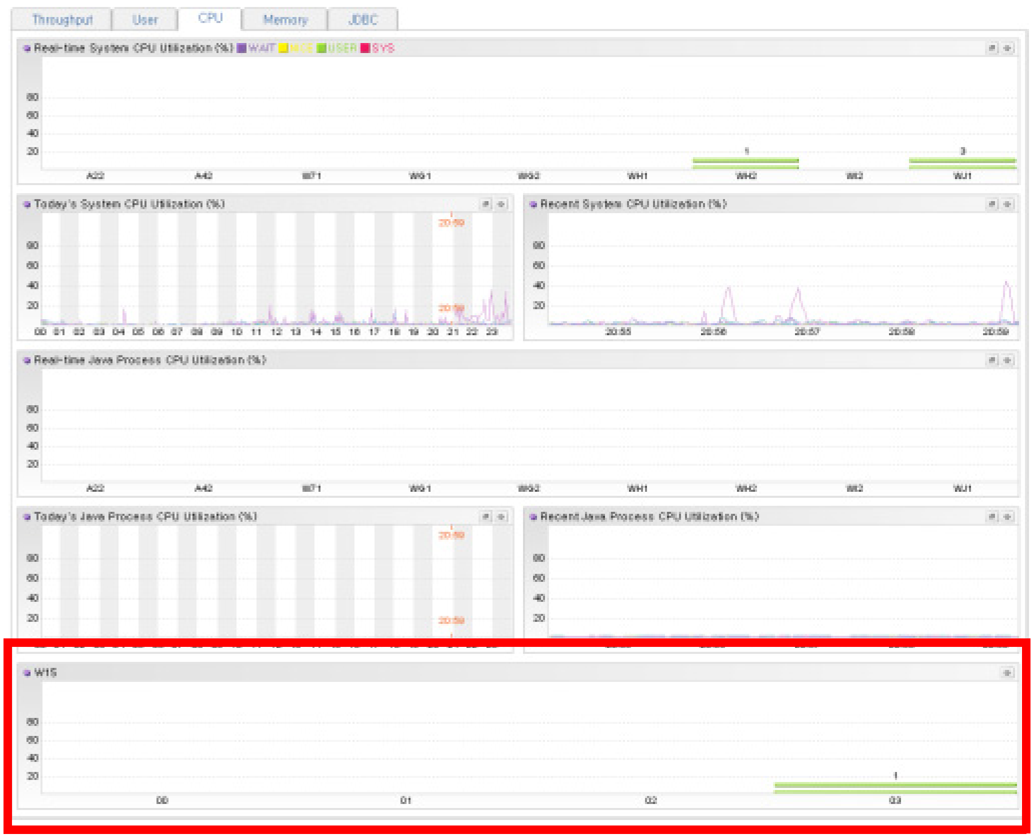


13.1.3.CPU

The CPU tab shows today's system CPU utilization, Java process CPU utilization in the form of a equilizer chart, a runtime line chart, a line chart and a bar chart

When the WMOND module is used, the system CPU utilization will be displayed in the form of an equalizer chart in the bottom of the CPU tab.

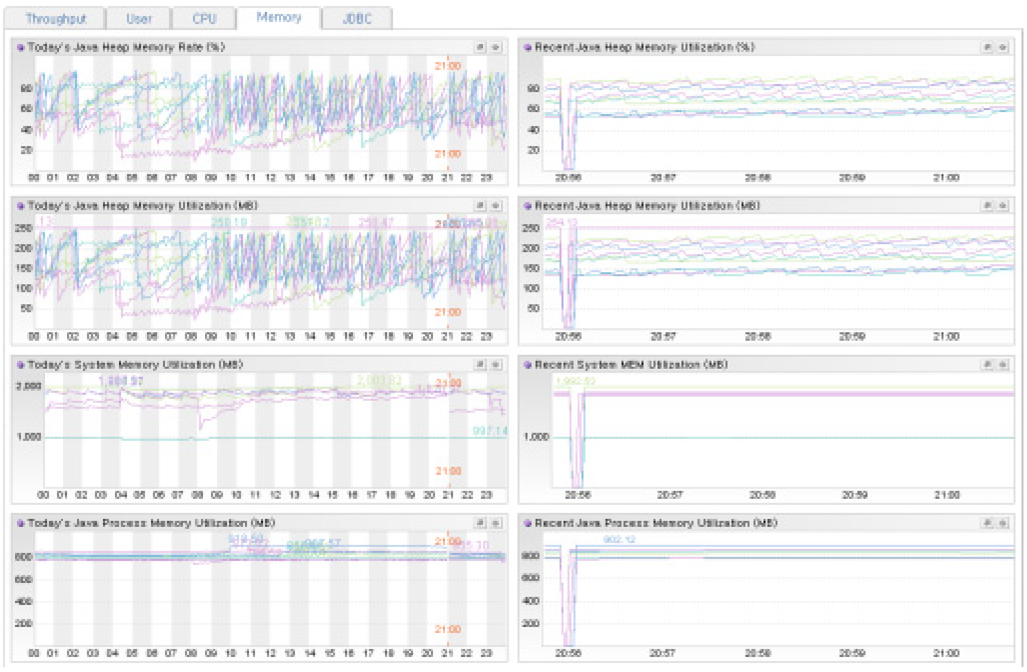
Figure 13-5:Real-time Status - CPU Tab


Throughput
User
CPU
Memory
JDBC
Real-time System CPU Utilization (%)
WAIT
USER
SYS
Today's System CPU Utilization (%)
Recent System CPU Utilization (%)
Real-time Java Process CPU Utilization (%)
Today's Java Process CPU Utilization (%)
Recent Java Process CPU Utilization (%)
W15

13.1.4.Memory

The memory tab shows today's Java heap memory utilization, system memory utilization, Java process memory utilization in the form of a runtime line chart and a line chart.

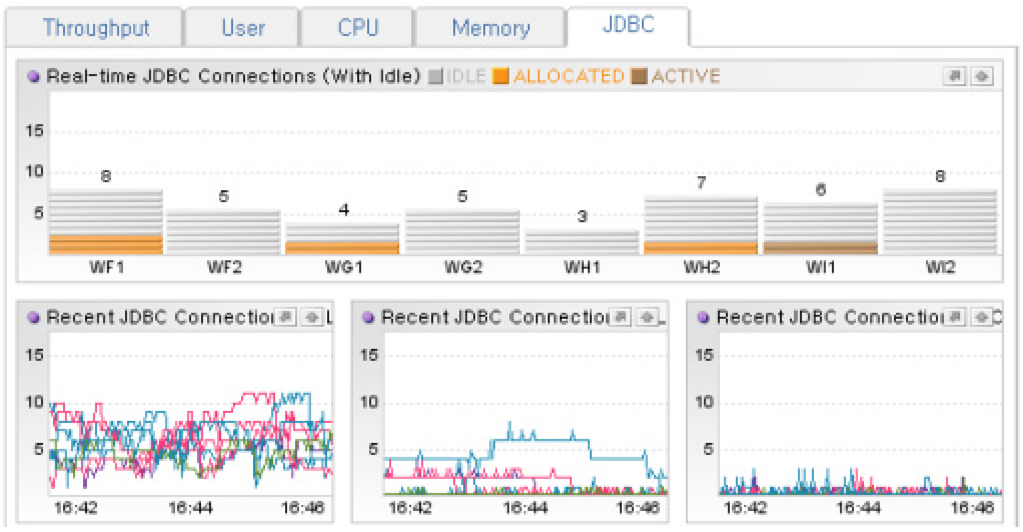
Figure 13-6:Real-time Status - Memory Tab


Throughput
User
CPU
Memory
JDBC
Today's Java Heap Memory Rate (%)
Recent Java Heap Memory Utilization (%)
Today's Java Heap Memory Utilization (MB)
Recent Java Heap Memory Utilization (MB)
Today's System Memory Utilization (MB)
Recent System MEM Utilization (MB)
Today's Java Process Memory Utilization (MB)
Recent Java Process Memory Utilization (MB)

13.1.5.DB

The DB tab shows today's DB connections in the form of a equilizer chart and a runtime line chart.

Figure 13-7:Real-time Status - DB Tab



13.2. Real-time Monitoring - Application

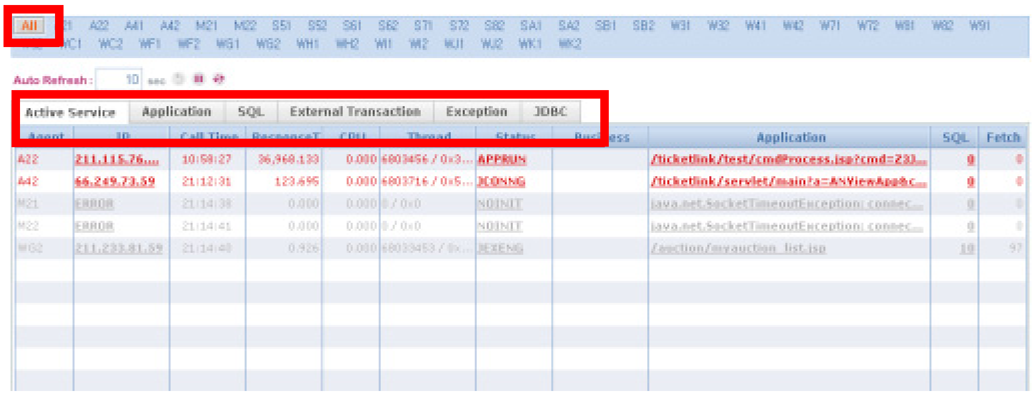
The JENNIFER agent collects the processing statistical data for active services, application, SQL, external transactions, exceptions and JDBC. The user can view the processing statistical data for applications in real time through the **[Real-Time Monitoring | Application]** menu. Internally, when requested by the user, the JENNIFER server retrieves the data by making a reverse TCP call to the JENNIFER agent.

Notice: The processing statistical data for active services and JDBC can be viewed in real time only.

The data is a 10-minute statistical summary. That is, it provides the total hits and the average response time for the login.jsp application over a 10-minute period, but does not provide the arrival and response times for individual login.jsp transactions. If you want to analyze individual transactions, refer to **[X-view and Profiling]**.

Notice: “Real-time” processing statistical data does not refer to the current 10-minute segment. It simply refers to the most recent of the 10-minute segments starting at 00, 10, 20 minutes, etc. So, if the current time is 09:18, it displays the processing statistical data from 09:10 to 09:20.

Figure 13-8:Real-time Monitoring | Application


All
A21 A22 A41 A42 M21 M22 S51 S52 S61 S62 S71 S72 S82 SA1 SA2 SB1 SB2 W31 W32 W41 W42 W71 W72 W81 W82 W91
W92 WC1 WC2 WF1 WF2 WG1 WG2 WH1 WH2 WI1 WI2 WJ1 WJ2 WK1 WK2
Auto Refresh : 10 sec
Active Service
Application
SQL
External Transaction
Exception
JDBC

Agent	IP	Call Time	ResponseT	CPU	Thread	Status	Business	Application	SQL	Fetch
A22	211.115.76....	10:58:27	36,968.133	0.000	6803456 / 0x3...	APPRUN		/ticketlink/test/cmdProcess.jsp?cmd=233...	0	0
A42	66.249.73.59	21:12:31	123.695	0.000	6803716 / 0x5...	JCONNG		/ticketlink/servlet/main?a=ANViewApp&c...	0	0
M21	ERROR	21:14:38	0.000	0.000	0 / 0x0	NOINIT		java.net.SocketTimeoutException: connec...	0	0
M22	ERROR	21:14:41	0.000	0.000	0 / 0x0	NOINIT		java.net.SocketTimeoutException: connec...	0	0
WG2	211.233.81.59	21:14:40	0.926	0.000	68033453 / 0x...	JEXENG		/auction/myauction_list.jsp	10	97

You can check application, SQL, external transactions, exceptions, and JDBC for certain JENNIFER agents only. Therefore, if you select [All] in the JENNIFER agent selection area, then the application, SQL, external transactions, exceptions and JDBC tabs will be deactivated. However, you can check the active services of all JENNIFER agents. If there are too many JENNIFER agents involved, it may take a very long time to view the active service list.

13.2.1.Active Service Processing Statistics

The following is a description of the active service list.

Table 13-1: Active Service List

Item	Description
Agent	JENNIFER agent ID
IP	The IP address of the user who requested the service. If you click on the column, the WHOIS function will provide the detailed information on the IP address.
Arrival time	The time at which the service request arrives
Elapsed time	The time elapsed while processing the service request. It is expressed in seconds.
CPU	The CPU utilization time used to process the service requests until now.
Thread	The Java thread ID processing the service request
Status	Displays the status of active service. If you click on the column, you can check the status of active service.
Task name	The name of the task processed by the application
Application	The application name. If you click on the column, you can check the detailed information of the active service.

Table 13-1: Active Service List

Item	Description
SQL	The number of SQL executed by the active service until now. If you click on the column, you can check the SQL executed by the active service.
Fetch	The number of SQL fetches that are processed

The following is a description of the active service status code.

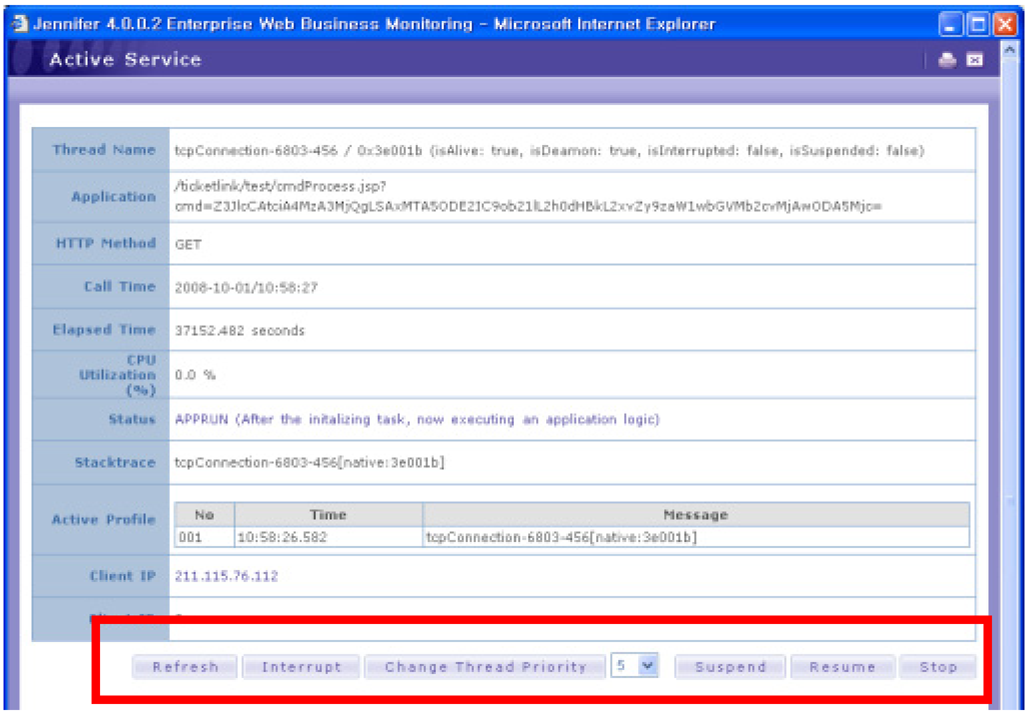
Table 13-2: Active Service Status Code

Status code	Description
NOINIT	During initialization, the basic information such as URI, remote IP address, Java thread ID is being extracted.
INITZG	Registering the active service and increasing the hits
REQST1	Basic information (user ID) is being extracted from the cookies
ARRSND	The service request data is being transmitted to the JENNIFER server via UDP
REQST2	Setting the character set and the initial processing of the HTTP GET/POST data
REQEND	After HTTP GET/POST processing, it is calculating the thinktime and the number of visitors.
INITED	The initialization is complete. Checking for use of PLC (Peak Load Control).
RJCTNG	A message is being sent to a user while the PLC is causing congestion.
RJCTED	Finished sending a message while PLC is causing congestion.
APPRUN	The initialization is finished, and the application logic is being executed now.
JCONNG	Making a JDBC connection (ds.getConection())
JCONND	After the JDBC connection, the next application logic is being executed.
JPRENG	Running the con.prepareStatement("...") code
JPREPD	After running the con.prepareStatement("...") code, the next logic is being executed.
JEXENG	Performing a JDBC SQL query
JEXETD	After performing a JDBC SQL query, the next logic is being executed.
JRSNXT	The rs.next() code is being executed repeatedly.
JRSFLS	After the rs.next() code processing results turns out to be false, the next logic is being executed.
JRSCLD	After executing the rs.close() code, the next logic is being executed.
JSTCLD	After executing the stat.close() and pstmt.close() codes, the next logic is being executed.
JCONCD	After executing the con.close() code, the next logic is being executed.
TXEXEC	Performing external transactions with CTG/Jolt/JLink/WebT.

Table 13-2: Active Service Status Code

Status code	Description
TXEXED	External transaction execution is finished. The next logic is being executed.
ENDAPP	All application logics have been executed.
ENDLNG	The response time for processing the user request exceeds the threshold.
ERREXP	Some exceptions are not processed while executing the application logic.
ERRIOE	A java.io.IOException has occurred.
ERRRCS	Process was forced to stop due to the detection of repetitive calls to the service method of the sublet object

Figure 13-9:Details of Active Services



Jennifer 4.0.0.2 Enterprise Web Business Monitoring - Microsoft Internet Explorer
Active Service
Refresh
Interrupt
Change Thread Priority
5
Suspend
Resume
Stop

Thread Name	tcpConnection-6803-456 / 0x3e001b (isAlive: true, isDeamon: true, isInterrupted: false, isSuspended: false)
Application	/ticketlink/test/cmdProcess.jsp? cmd=Z3JlcCAtciA4MzA3MjQgLSAxMTA5ODE2IC9ob21lL2h0dHBkL2xvZy9zaW1wbGVMb2cvMjAwODA5Mjc=
HTTP Method	GET
Call Time	2008-10-01/10:58:27
Elapsed Time	37152.482 seconds
CPU Utilization (%)	0.0 %
Status	APPRUN (After the initalizing task, now executing an application logic)
Stacktrace	tcpConnection-6803-456[native:3e001b]
Active Profile	
Client IP	211.115.76.112

No	Time	Message
001	10:58:26.582	tcpConnection-6803-456[native:3e001b]

By clicking the buttons at the bottom of the detail of the active service screen, you can control the Java threads that process the active service.

- Interrupt - Call the interrupt method of the Java thread object.
- Change thread priority - Call the setPriority method of the Java thread to change the priority for the threads.
- Suspend - Call the suspend method of the Java thread object to pause the Java thread.

- Resume - Call the resume method of the Java thread to restart the Java thread.
- Stop - Call the stop method of the Java thread to stop the Java thread.

However, most of the java.lang.Thread methods called by clicking the button are deprecated. If you click on the stop button while a JDBC connection is open, then it could lead to a JDBC connection leakage.

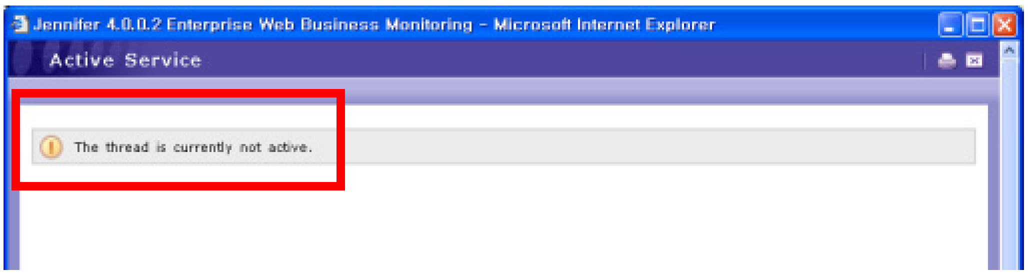
In addition, for performance improvement, the WAS manages the threads using the thread pool. If the WAS checks for abnormal threads before assigning them to the application, it will be okay to stop a specific thread through JENNIFER. However, if the WAS does not check for abnormal threads, then an abnormal thread could be assigned to the application, and some unforeseen problems could occur while processing other services. Therefore, if active services that are not likely to be completed are using the CPU excessively, you are recommended to suspend them, rather than stop them.

Using the enable_active_thread_kill option of the JENNIFER agent, you can prevent the Java thread from being stopped.

```
enable_active_thread_kill = false
```

If the active service is terminated, the following message will appear in a pop-up window:

Figure 13-10:When the Active Service is terminated


Jennifer 4.0.0.2 Enterprise Web Business Monitoring - Microsoft Internet Explorer
Active Service

The thread is currently not active.

13.2.2.Application Processing Statistics

The following table describes the application list:

Table 13-3: Application List

Items	Description
Hits	The number of executed applications
Success counts	The number of successfully executed applications
Failure counts	The number of failed applications

Table 13-3: Application List

Items	Description
Total response time	The total response time for executing the application
Average response time	The total response time divided by the number of hits
Standard deviation	The standard deviation for the response time for executing the application
Minimum response	The minimum response time for executing the application
Maximum response	The maximum response time for executing the application
Average CPU	The average CPU use-time used to execute the application
CPU(tpmC)	The CPU(tpmC) value used to execute the application
CPU sum	The total CPU use-time to execute the application
Application	The application name. In general, it refers to a URI.

If you click on an application in the application list, the list of SQL and external transactions executed by the application will be displayed in the bottom. The following table describes the list.

Table 13-4: Detailed Items related to Application

Item	Description
Type	SQL stands for the SQL, while TX stands for external transactions.
Occupancy ratio	The proportion of the response time for the application that is occupied by the SQL or the external transaction. It is expressed as a %.
Hits	The number of SQL or external transactions called by the application
Total response time	The total response time to execute the SQL or the external transaction called by the application
Average response time	The response time divided by the number of hits
Standard deviation	The standard deviation for the response time to execute the SQL or the external transaction called by the application
Minimum response	The minimum response time to execute the SQL or the external transaction called by the application
Maximum response	The maximum response time to execute the SQL or the external transaction called by the application
SQL/transaction	The name of the SQL or external transaction

13.2.3.SQL Processing Statistics

The following table describes the SQL list:

Table 13-5: SQL List

Item	Description
Hits	The number of executed SQL
Total response time	The total response time for executing the SQL
Average response time	The total response time divided by the number of hits
Standard deviation	The standard deviation for the response time for executing the SQL
Minimum response	The minimum response time for executing the SQL
Maximum response	The maximum response time for executing the SQL
SQL	SQL

If you click on an SQL in the SQL list, the list of application that executed the SQL will be displayed in the bottom. The following table describes the list.

Table 13-6: Detailed Items Related to SQL

Item	Description
Occupancy ratio	The proportion of the response time for the application that is occupied by the SQL. It is expressed as a %.
Hits	The number of SQL called by the application
Total response time	The total response time to execute the SQL called by the application
Average response time	The response time divided by the number of hits
Standard deviation	The standard deviation for the response time to execute the SQL called by the application
Minimum response	The minimum response time to execute the SQL called by the application
Maximum response	The maximum response time to execute the SQL called by the application
Application	Application name

13.2.4.External Transaction Processing Statistics

The following table describes the external transaction list:

Table 13-7: External Transaction List

Item	Description
Hits	The number of executed external transaction
Total response time	The total response time for executing the external transaction
Average response time	The total response time divided by the number of hits

Table 13-7: External Transaction List

Item	Description
Standard deviation	The standard deviation for the response time for executing the external transaction
Minimum response	The minimum response time for executing the external transaction
Maximum response	The maximum response time for executing the external transaction
Transaction	External transaction name

If you click on an external transaction in the external transaction list, the list of application that executed the external transaction will be displayed in the bottom. The following table describes the list.

Table 13-8: Detailed Items Related to External Transaction List

Item	Description
Occupancy ratio	The proportion of the response time for the application that is occupied by the external transaction. It is expressed as a %.
Hits	The number of external transaction called by the application
Total response time	The total response time to execute the external transaction called by the application
Average response time	The response time divided by the number of hits
Standard deviation	The standard deviation for the response time to execute the external transaction called by the application
Minimum response	The minimum response time to execute the external transaction called by the application
Maximum response	The maximum response time to execute the external transaction called by the application
Application	Application name

13.2.5.Exception Statistics

The following table describes the exception list:

Table 13-9: Exception List

Items	Description
Occurrence counts	The number of exceptions occurring in the application
Occurrence ratio	The ratio of the number of exceptions in the application to the total number of exceptions in every application. It is expressed asa %.
Exception item	Exception name

If you click on an exception in the exception list, then the list of applications that include the exception will be displayed. The following table describes this list.

Table 13-10: Detailed Items related to Exceptions

Item	Description
Occurrence counts	The number of exceptions occurring in the application
Occurrence ratio	The ratio of the number of exceptions in the application to the total number of exceptions in every application. It is expressed asa %.
Application	The name of the application in which the exception occurs
Content	The content related to exceptions. Shows the exception messages and the stacktrace.

13.2.6.DB Statistics

The following is the description of the DB list.

Table 13-11: DB List

Items	Description
Hash code	The hash code for the JDBC connection object
DB SID	Oracle database SID (displayed only when the Oracle database is used). If you want to display SID, additional onfiguration is required.
Class	The implementation class for the java.sql.Connection interface. It varies depending on the JDBC driver used.
Name	The name registered in the JNDI
IP	The IP address of the user who requested the service processed by the Java thread that uses the JDBC connection
Arrival time	Arrival time of the request to the service processed by the Java thread that uses the JDBC connection
Elapsed time	The elapsed time of the service processed by the Java thread that uses the JDBC connection
Thread	The ID of the Java thread using the JDBC connection
Status	Same as the status code for the active service
Application	The application name of the service processed by the Java thread that uses the JDBC connection. You can click on the column to view the details of the active service.

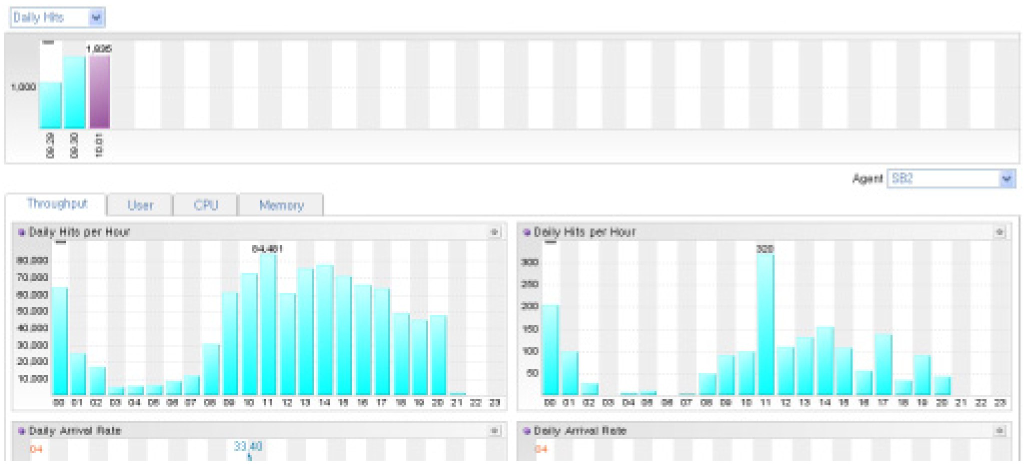
The items listed from “IP” through to “Application” describe the active service processed by the Java thread that uses the JDBC connection. If the JDBC connection is in standby mode, then no content is displayed.

Notice: If there is a programming error, the same JDBC connection object could be assigned two or more Java threads at the same point. In this case, you will see the [Duplicated allocation] message.

13.3. Statistics - Statistics Status

In the **[Real-time Monitoring | Statistics Status]** menu, you can monitor the general performance data for the selected date in the form of various charts.

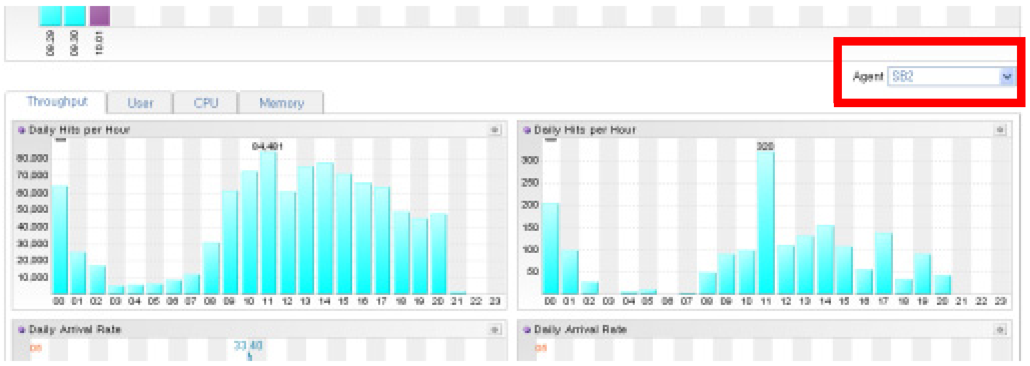
Figure 13-11:Statistics Status (1)



First, you can use the bar chart at the top of the screen to check the daily hits and visitors. Basically, this will display the number of hits for each day. If you click on the bar corresponding to the day, you can see the general performance data, which appears in multiple tabs in the middle of the screen.

Using these, you can view the general performance data for the day, including throughput, users, CPU and memory.

Figure 13-12:Statistics Status (2)

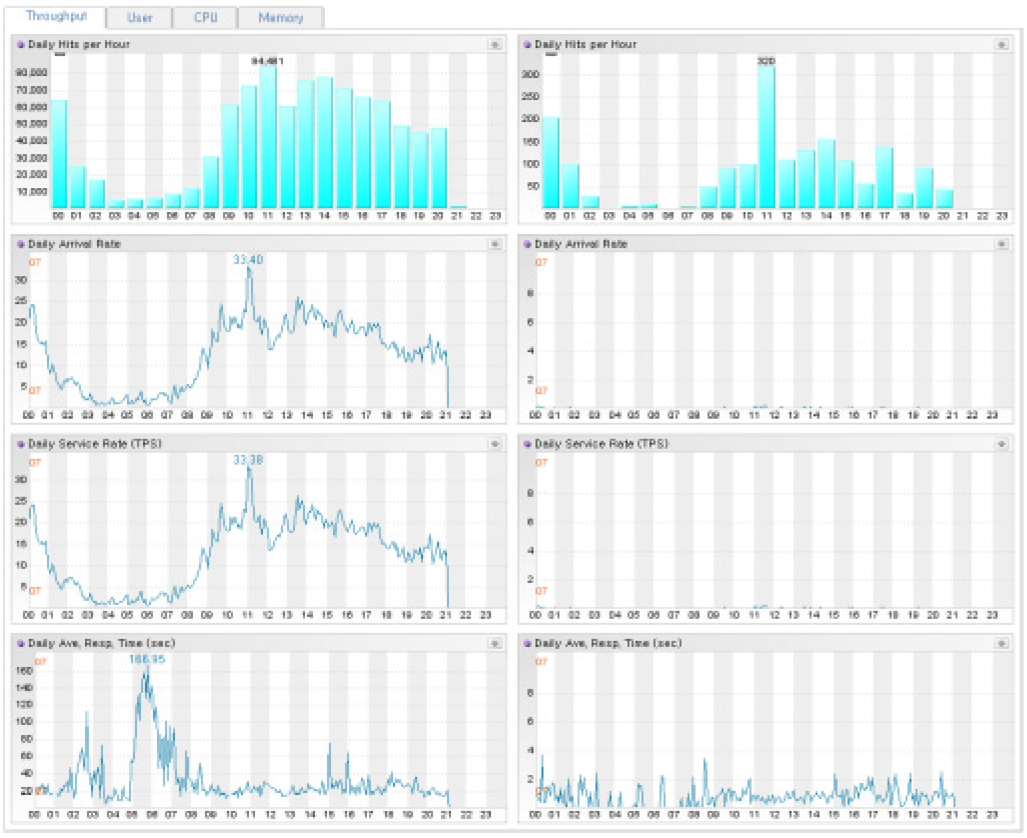


Notice: The statistics status mostly uses line charts.

13.3.1.Throughput

The throughput tab shows the selected day's arrival rate, a service rate, a average response time and hits per hour in the form of a runtime line chart and a bar chart.

Figure 13-13:Statistics Status - Throughput Tab

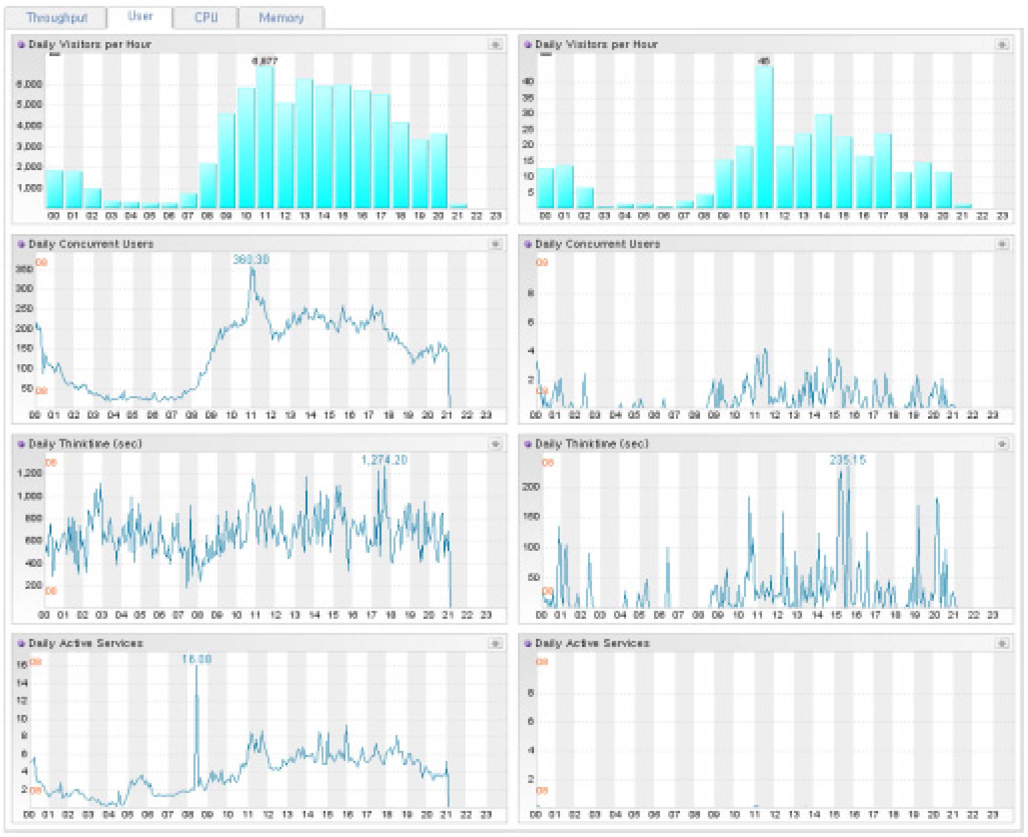


The chart to the left of the throughput tab displays the cumulative value or the average of all the values collected by all of the monitored JENNIFER agents, not just specific JENNIFER agents. If the chart at the right is a bar chart, then no content is displayed, while if it is a line chart, it will display the performance data collected for individual JENNIFER agents as lines. However, if you click on an individual JENNIFER agent in the list at the upper right corner of the throughput tab, the bar chart will display the performance data collected from that JENNIFER agent, and the line chart will only display the performance data collected from that specific JENNIFER agent.

13.3.2.User

The user tab shows the selected day's concurrent users, active services, think time, and visitors per hour in the form of a line chart and a bar chart

Figure 13-14:Statistics Status - User Tab

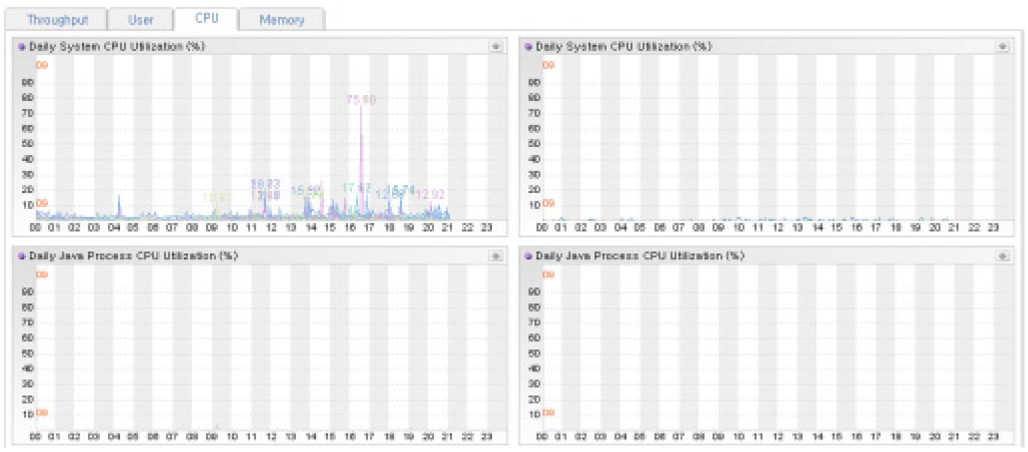


The chart to the left of the user tab displays the cumulative value or the average of all the values collected by all of the monitored JENNIFER agents, not just specific JENNIFER agents. If the chart at the right is a bar chart, then no content is displayed, while if it is a line chart, it will display the performance data collected for individual JENNIFER agents as lines. However, if you click on an individual JENNIFER agent in the list at the upper right corner of the throughput tab, the bar chart will display the performance data collected from that JENNIFER agent, and the line chart will only display the performance data collected from that specific JENNIFER agent.

13.3.3.CPU

The CPU tab shows the selected dat's system CPU utilization, Java process CPU utilization in the form of a line chart.

Figure 13-15:Statistics Status - CPU Tab

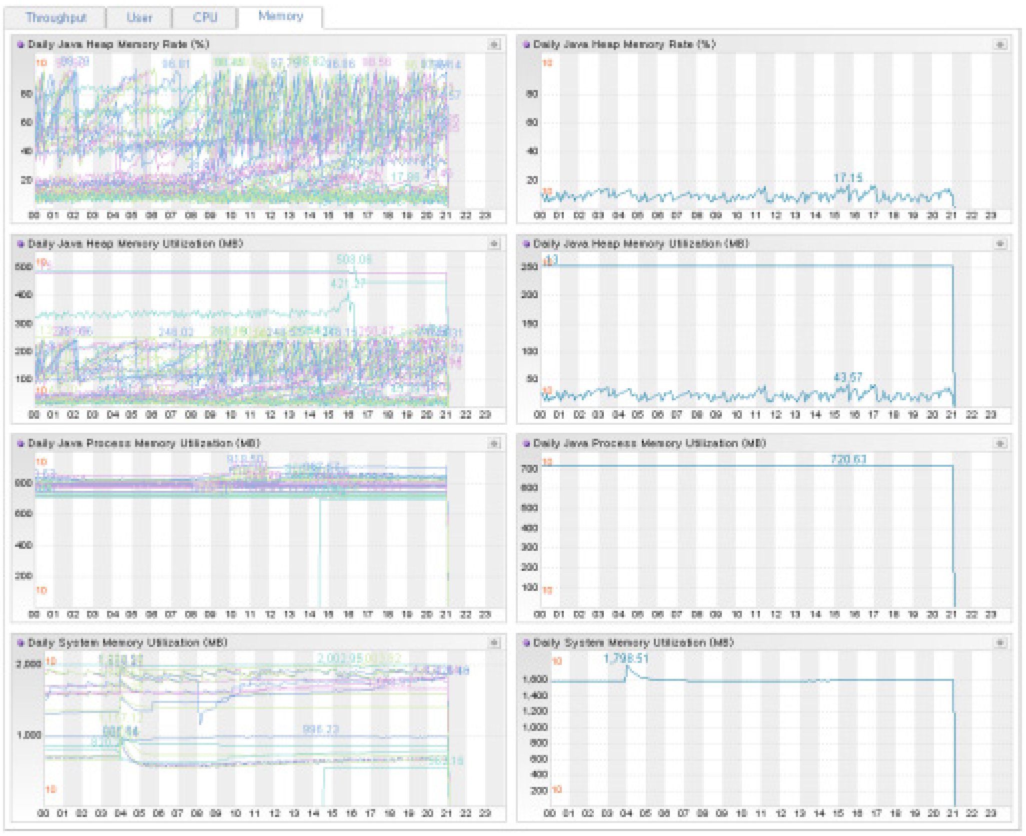

Throughput
User
CPU
Memory
Daily System CPU Utilization (%)
Daily System CPU Utilization (%)
Daily Java Process CPU Utilization (%)
Daily Java Process CPU Utilization (%)

Unlike the throughput or user-related performance data, the cumulative value or average has no bearing on the CPU-related performance data. Therefore, the line chart at the left of the CPU tab displays the performance data for individual JENNIFER agents as separate lines.

13.3.4.Memory

The memory tab shows the selected day's Java heap memory utilization, system memory utilization, Java process memory utilization in the form of a line chart.

Figure 13-16:Statistics Status - Memory Tab


Throughput
User
CPU
Memory
Daily Java Heap Memory Rate (%)
Daily Java Heap Memory Utilization (MB)
Daily Java Process Memory Utilization (MB)
Daily System Memory Utilization (MB)

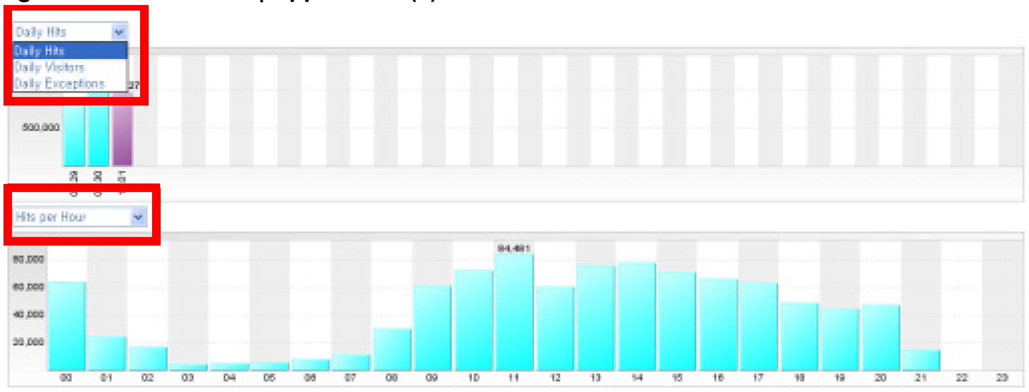
Unlike the throughput or user-related performance data, the cumulative value or average has no bearing on the Memory-related performance data. Therefore, the line chart at the left of the CPU tab displays the performance data for individual JENNIFER agents as separate lines.

13.4. Statistics - Application

The JENNIFER server stores the processing statistical data for applications, SQL, external transactions and exceptions collected from the JENNIFER agent in the database once every 10 minutes. The user can check the data using the **[Statistics | Application]** menu.

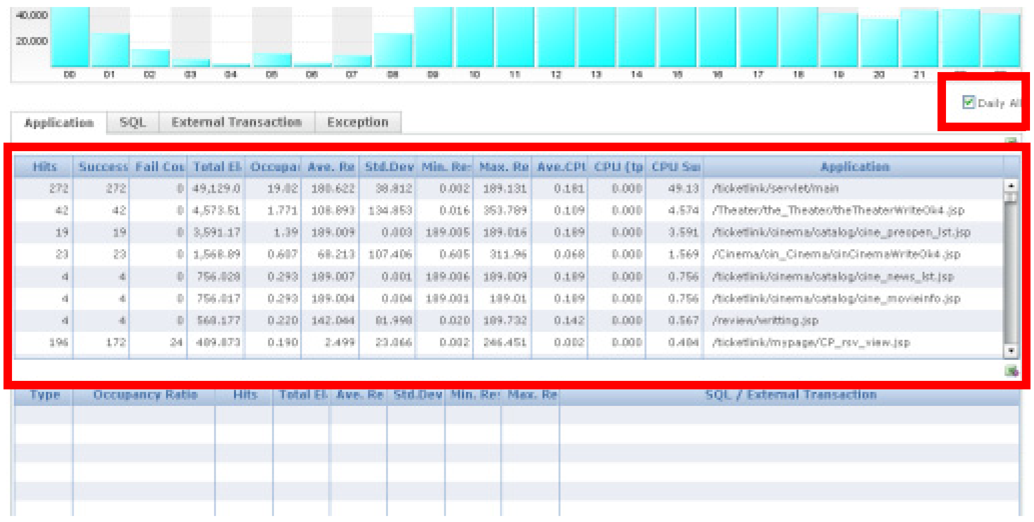
The data is a summary of statistics over a 10-minute period. For instance, it provides the total hits and the average response time for the login.jsp application over a 10-minute period, but does not provide the arrival and response time for individual login.jsp transactions. If you want to analyze individual transactions, you should refer to **[X-view Screen Analysis]**.

Figure 13-17:Statistics | Application (1)



The first bar graph displays performance data such as the hits per day, the visitors per day and the exceptions per day. The default setting is the hits per day. If you click a certain day in the first bar graph, you can see this day's various performance data such as the hits per hour, the visitors per hour, the concurrent users, the think time, the active services, the arrival rates, the service rates, the average response time, and the exceptions per hour in the second bar graph. The default setting is the hits per hour. The hits per hour, the visitors per hour, and the exceptions per hour is displayed through a bar graph for an hour and other data such as the concurrent users, the think time, the active services, the arrival rates, the service rates, and the average response time is displayed through a line chart for 5 minutes.

Figure 13-18:Statistics | Application (2)



Hits	Success	Fail Cou	Total El	Occupa	Ave. Re	Std.Dev	Min. Re	Max. Re	Ave.CPU	CPU (tp	CPU Su	Application
272	272	0	49,129.0	19.02	180.622	38.812	0.002	189.131	0.181	0.000	49.13	/ticketlink/servlet/main
42	42	0	4,573.51	1.771	108.893	134.853	0.016	353.789	0.109	0.000	4.574	/Theater/the_Theater/theTheaterWriteOk4.jsp
19	19	0	3,591.17	1.39	189.009	0.003	189.005	189.016	0.189	0.000	3.591	/ticketlink/cinema/catalog/cine_prespen_lst.jsp
23	23	0	1,568.89	0.607	68.213	107.406	0.605	311.96	0.068	0.000	1.569	/Cinema/cin_Cinema/cinCinemaWriteOk4.jsp
4	4	0	756.028	0.293	189.007	0.001	189.006	189.009	0.189	0.000	0.756	/ticketlink/cinema/catalog/cine_news_lst.jsp
4	4	0	756.017	0.293	189.004	0.004	189.001	189.01	0.189	0.000	0.756	/ticketlink/cinema/catalog/cine_movieinfo.jsp
4	4	0	568.177	0.220	142.044	81.990	0.020	189.732	0.142	0.000	0.567	/review/writting.jsp
196	172	24	409.873	0.190	2.499	23.066	0.002	246.451	0.002	0.000	0.404	/ticketlink/mypage/CP_rsv_view.jsp

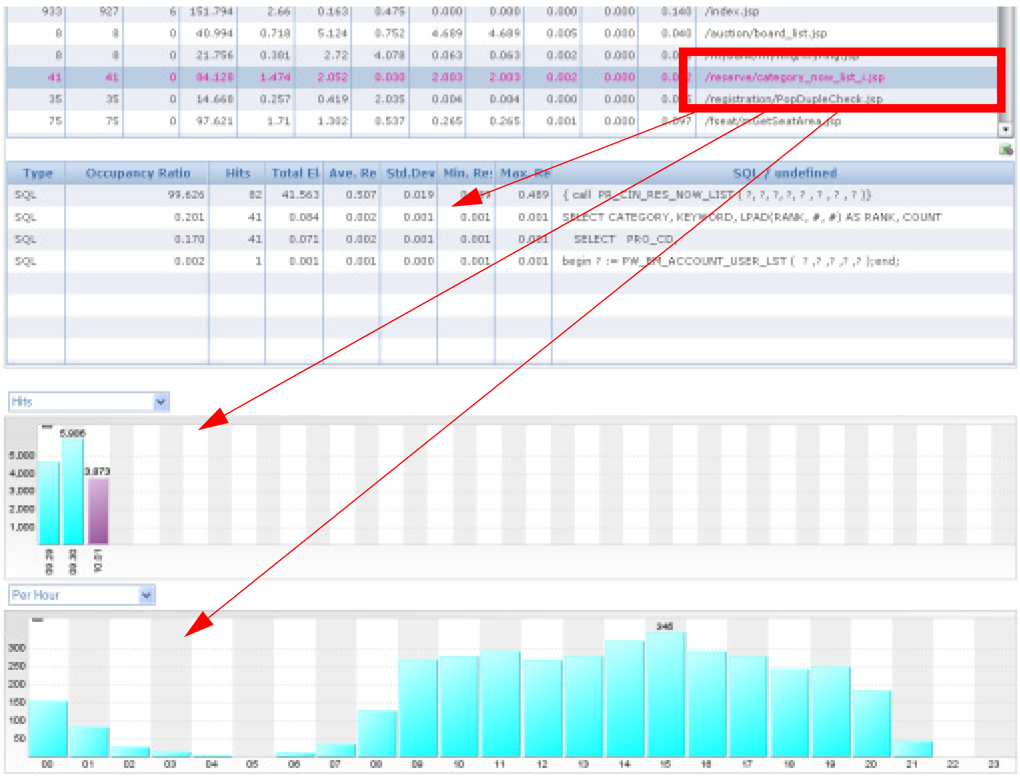
Type	Occupancy Ratio	Hits	Total El	Ave. Re	Std.Dev	Min. Re	Max. Re	SQL / External Transaction

If you click a certain time in the second graph after selecting a chart type among the hits per hour, the visitors per hour, and exceptions per hour, you can see this time's data such as application, SQL, external transaction, and processing status data for exceptions.

When you select the check box of **[Daily All]** on the right, today's total statistical data of application, SQL, external transaction, exception, etc processing status is displayed.

Notice: If you click **[Daily All]** on the right, you can see the processing statistical data such as application, SQL, external transaction, exception, etc of the selected date regardless of time zone.

Figure 13-19:Statistics | Application (3)



933	927	6	151.794	2.66	0.163	0.475	0.000	0.000	0.000	0.000	0.140	/index.jsp
8	8	0	40.994	0.718	5.124	0.752	4.689	4.689	0.005	0.000	0.040	/auction/board_list.jsp
8	8	0	21.756	0.381	2.72	4.078	0.063	0.063	0.002	0.000	[illegible]	[illegible]
41	41	0	84.128	1.474	2.052	0.030	2.003	2.003	0.002	0.000	[illegible]	/reserve/category_now_list_i.jsp
35	35	0	14.668	0.257	0.419	2.035	0.004	0.004	0.000	0.000	[illegible]	/registration/PopDupleCheck.jsp
75	75	0	97.621	1.71	1.302	0.537	0.265	0.265	0.001	0.000	[illegible]	/fseat/mGetSeatArea.jsp

Type	Occupancy Ratio	Hits	Total El	Ave. Re	Std.Dev	Min. Re:	Max. Re	SQL / undefined
SQL	99.626	82	41.563	0.507	0.019	[illegible]	0.489	{ call PB_CIN_RES_NOW_LIST (?, ?, ?, ?, ? , ? , ? , ?)}
SQL	0.201	41	0.084	0.002	0.001	0.001	0.001	SELECT CATEGORY, KEYWORD, LPAD(RANK, #, #) AS RANK, COUNT
SQL	0.170	41	0.071	0.002	0.001	0.001	0.001	SELECT PRO_CD,
SQL	0.002	1	0.001	0.001	0.000	0.001	0.001	begin ? := PW_BM_ACCOUNT_USER_LST (? ,? ,? ,? ,?);end;

If you select a specific application, SQL, external transaction or exception in the first grid, then all of the relevant information will be displayed in the second grid, and in the third and the fourth charts.

For instance, if you select a specific application in the first grid, then the second grid will display the list of SQL and external transactions executed by the application during the same time period. The third chart will display the number of hits, the failure counts, the total, average, maximum and minimum response times, the CPU time and CPU(tpmC) for the application on each day. If you click on a certain day in the third chart, the data for the selected day will be displayed in the form of a bar or line chart.

13.4.1.Statistics of Application Processing Status

The following table describes the application list:

Table 13-12: Application List

Items	Description
Hits	The number of executed applications
Success counts	The number of successfully executed applications
Failure counts	The number of failed applications
Total response time	The total response time for executing the application
Average response time	The total response time divided by the number of hits
Standard deviation	The standard deviation for the response time for executing the application
Minimum response	The minimum response time for executing the application
Maximum response	The maximum response time for executing the application
Average CPU	The average CPU use-time used to execute the application
CPU(tpmC)	The CPU(tpmC) value used to execute the application
CPU sum	The total CPU use-time to execute the application
Application	The application name. In general, it refers to a URI.

If you click on an application in the application list, the list of SQL and external transactions executed by the application will be displayed in the bottom. The following table describes the list.

Table 13-13: Detailed Items related to Application

Item	Description
Type	SQL stands for the SQL, while TX stands for external transactions.
Occupancy ratio	The proportion of the response time for the application that is occupied by the SQL or the external transaction. It is expressed as a %.
Hits	The number of SQL or external transactions called by the application
Total response time	The total response time to execute the SQL or the external transaction called by the application
Average response time	The response time divided by the number of hits
Standard deviation	The standard deviation for the response time to execute the SQL or the external transaction called by the application
Minimum response	The minimum response time to execute the SQL or the external transaction called by the application

Table 13-13: Detailed Items related to Application

Item	Description
Maximum response	The maximum response time to execute the SQL or the external transaction called by the application
SQL/transaction	The name of the SQL or external transaction

13.4.2.Statistics of SQL Processing Status

The following table describes the SQL list:

Table 13-14: SQL List

Item	Description
Hits	The number of executed SQL
Total response time	The total response time for executing the SQL
Average response time	The total response time divided by the number of hits
Standard deviation	The standard deviation for the response time for executing the SQL
Minimum response	The minimum response time for executing the SQL
Maximum response	The maximum response time for executing the SQL
SQL	SQL

If you click on an SQL in the SQL list, the list of application that executed the SQL will be displayed in the bottom. The following table describes the list.

Table 13-15: Detailed Items Related to SQL

Item	Description
Occupancy ratio	The proportion of the response time for the application that is occupied by the SQL. It is expressed as a %.
Hits	The number of SQL called by the application
Total response time	The total response time to execute the SQL called by the application
Average response time	The response time divided by the number of hits
Standard deviation	The standard deviation for the response time to execute the SQL called by the application
Minimum response	The minimum response time to execute the SQL called by the application
Maximum response	The maximum response time to execute the SQL called by the application
Application	Application name

13.4.3.Statistics of External transaction Processing Status

The following table describes the external transaction list:

Table 13-16: External Transaction List

Item	Description
Hits	The number of executed external transaction
Total response time	The total response time for executing the external transaction
Average response time	The total response time divided by the number of hits
Standard deviation	The standard deviation for the response time for executing the external transaction
Minimum response	The minimum response time for executing the external transaction
Maximum response	The maximum response time for executing the external transaction
Transaction	External transaction name

If you click on an external transaction in the external transaction list, the list of application that executed the external transaction will be displayed in the bottom. The following table describes the list.

Table 13-17: Detailed Items Related to External Transaction List

Item	Description
Occupancy ratio	The proportion of the response time for the application that is occupied by the external transaction. It is expressed as a %.
Hits	The number of external transaction called by the application
Total response time	The total response time to execute the external transaction called by the application
Average response time	The response time divided by the number of hits
Standard deviation	The standard deviation for the response time to execute the external transaction called by the application
Minimum response	The minimum response time to execute the external transaction called by the application
Maximum response	The maximum response time to execute the external transaction called by the application
Application	Application name

13.4.4.Statistics of Exception Processing Status

The following table describes the exception list:

Table 13-18: Exception List

Items	Description
Occurrence counts	The number of exceptions occurring in the application
Occurrence ratio	The ratio of the number of exceptions in the application to the total number of exceptions in every application. It is expressed asa %.
Exception item	Exception name

If you click on an exception in the exception list, then the list of applications that include the exception will be displayed. The following table describes this list.

Table 13-19: Detailed Items related to Exceptions

Item	Description
Occurrence counts	The number of exceptions occurring in the application
Occurrence ratio	The ratio of the number of exceptions in the application to the total number of exceptions in every application. It is expressed asa %.
Application	The name of the application in which the exception occurs
Content	The content related to exceptions. Shows the exception messages and the stacktrace.

If you click the application field on the detailed error list, you will see the detailed error data in the pop-up window.

13.5. Perfomance Data Trend Analysis(PTA)

Perfomance Data Trend Analysis (PTA) is newly added on the JENNIFER 4.1 and it helps user to analyze the various performance data.

As comparing the different performance statistics on an arbitrary date, PTA analyzes the current running system’s interconnections and its variations. Through this function, user is able to find out what is the most suitable performance management way in the system.

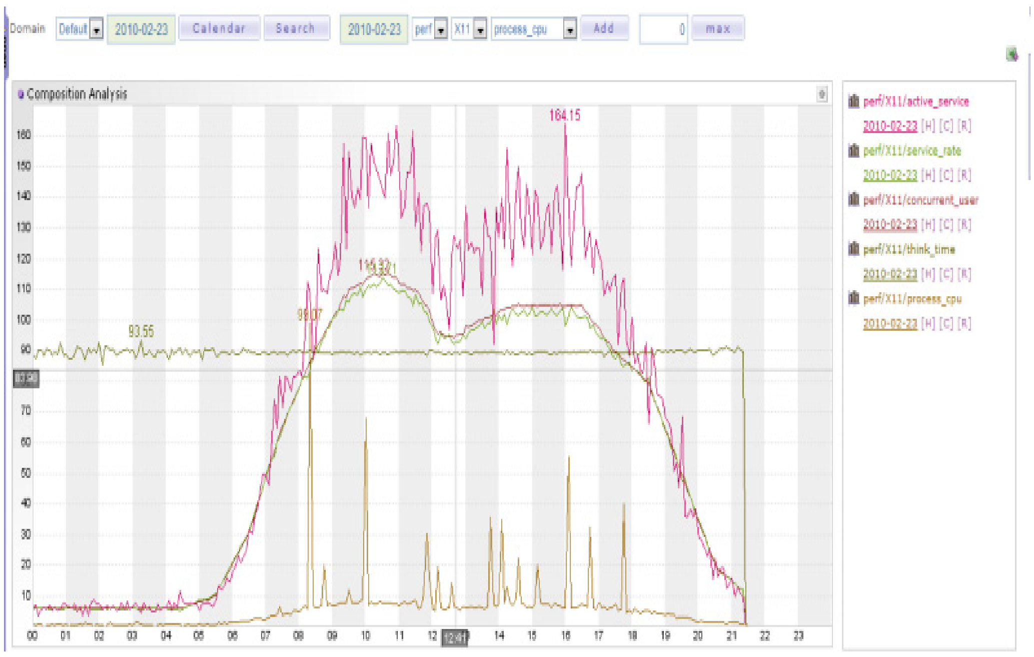
The data which is needed in managing application performance can be divided into three parts such as resource, user and service. As more users use the application services, the service requests and system resource increase.

However, the interconnections or mutual relevancy on those data is not simple and could not be a mathematical explanation. Somebody calculates resource per request, but this way is not reasonable one. Therefore, the industry does capacity planning by combing traditional overhead calculation method and existed experiences.

While the experience is one of the meaningful parts in the performance management, an efficient analysis tool is also needed to emphasize and understand more clear the experience.

PTA of JENNIFER compares the variations of collected performance data on a daily basis and analyzes them. As comparing mutual relevancy of today's/yesterday's throughput variations and CPU utilization, user can recognize the interconnections between performance data in order to improve web system performance.

Figure 13-20:PTA Data



PTA data is separately managed and has a different storing period comparing to the other collected performance data.

13.5.1.Utilization of the Statistical PTA

If you install the JENNIFER 4.1.0 successfully, FTA is automatically enabled on the JENNIFER.

- Menu : [Statistics|PTA]

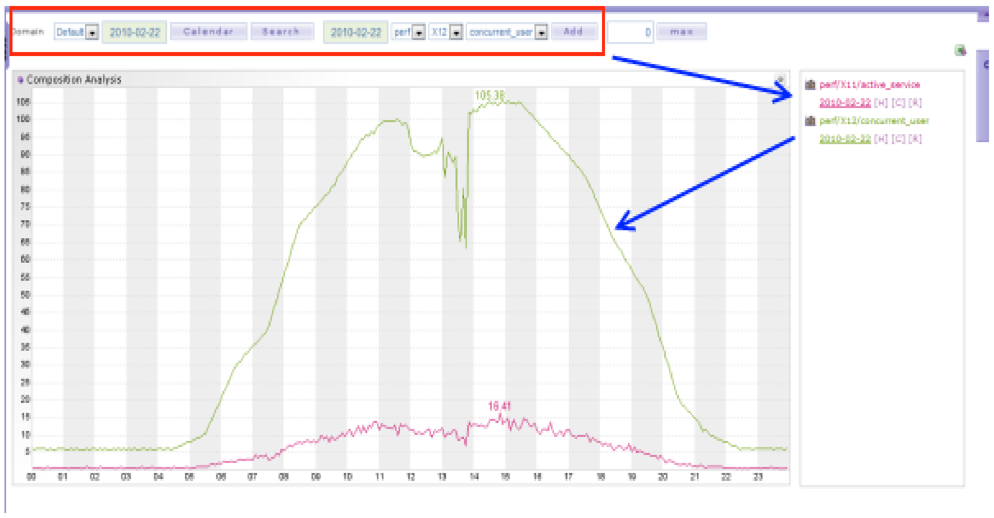
However, if you upgrade the JENNIFER older version than 4.1.0, register the PTA on the Statistics menu.

```
stat_pta.jsp
```

The PTA provides insightful dashboard. At first, input the upper drop-down menus by selecting date and application information. Click the [Add] button and you will see the PTA graph.

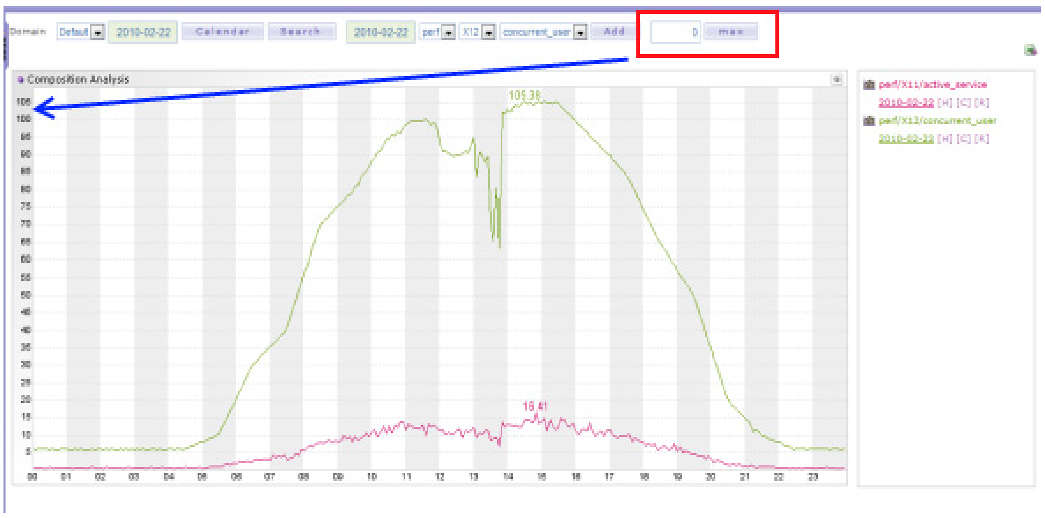
```
Calendar --> Search --> Select Box --> Add
```

Figure 13-21:PTA Menu



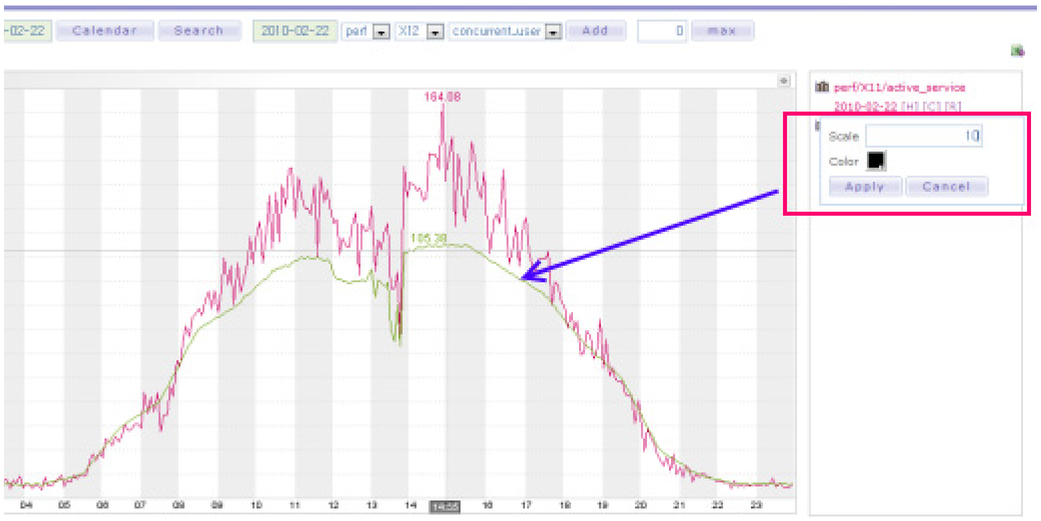
The graph size is automatically adjusted. However, you can fix the scale of the graph by setting “MAX” value as well.

Figure 13-22:Graph Scale



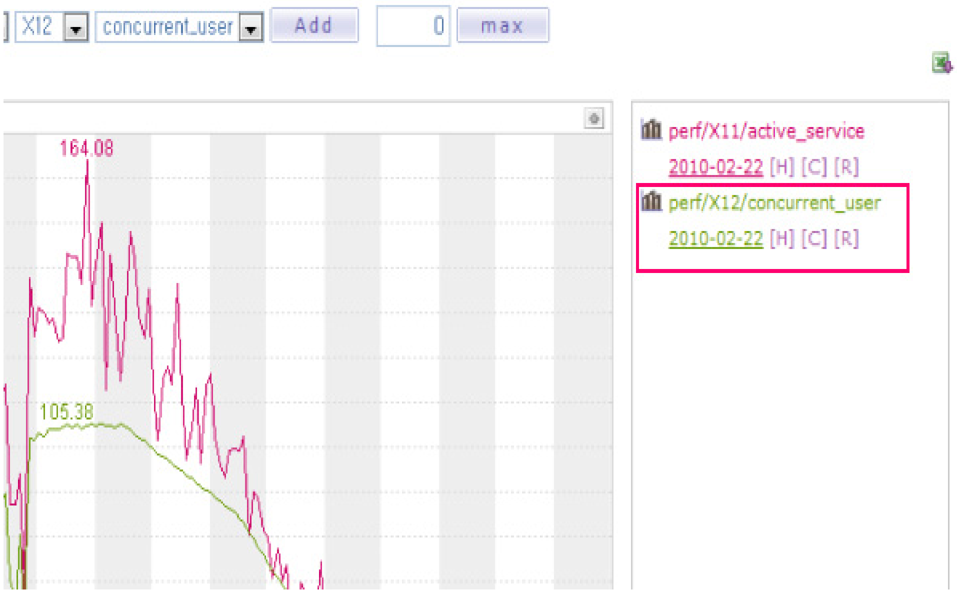
The PTA aims to compare and analyze the each different performance data in a large scope. Therefore, the certain graph could have unrelated values. In this case, user can modify the graph scale on the graph characteristics.

Figure 13-23:Modity the Scale



You can hide the each graph and even delete the unnecessary graph.

Figure 13-24:Downloading the Data

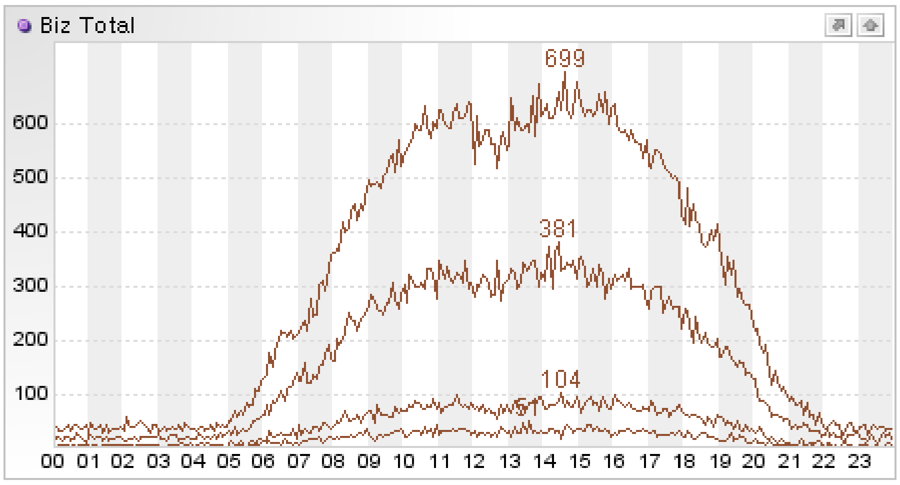


Lastly, if you click the button on the right upper area, you can download the data on each graph as an excel file.

13.5.2.Utilizing Real-time PTA

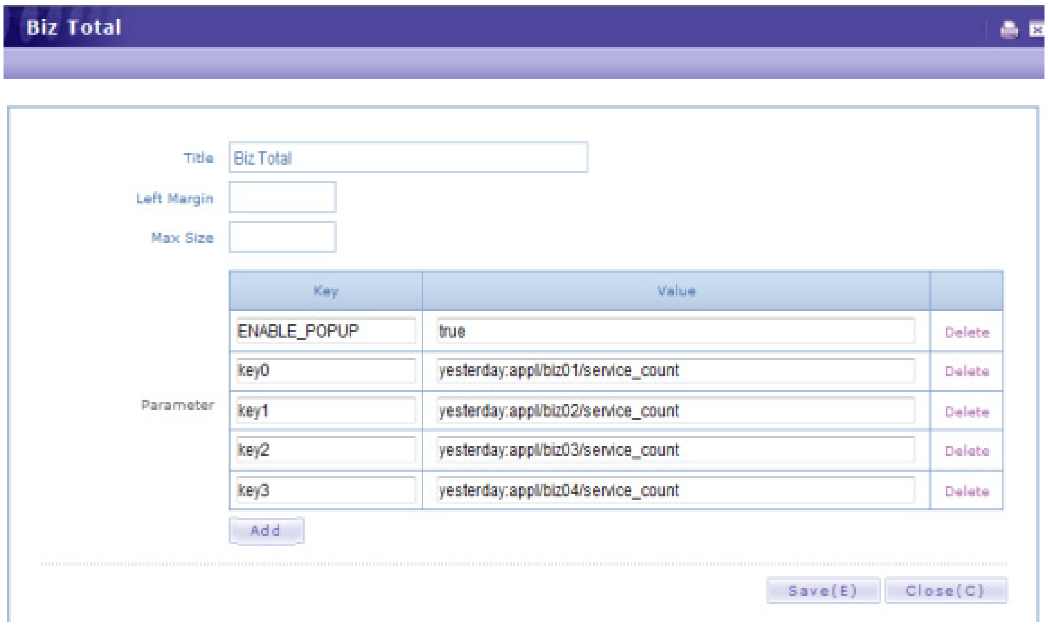
Use this function when you want to monitor the daily variations of the performance data in real-time.

Figure 13-25:PTA Graph



User can compare the each different performance data on a single screen.

Figure 13-26:Biz Total



Title	Biz Total
Left Margin	
Max Size	

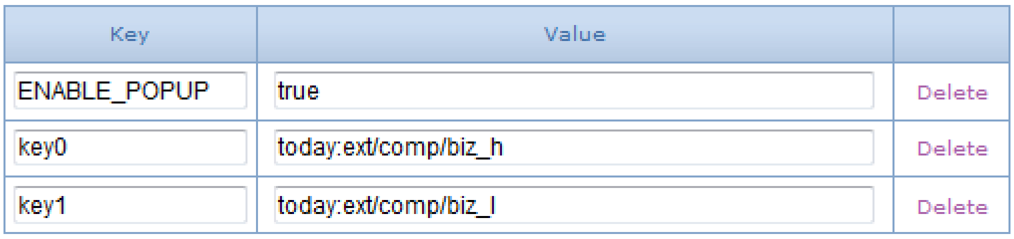
Parameter

Key	Value	
ENABLE_POPUP	true	Delete
key0	yesterday:appl/biz01/service_count	Delete
key1	yesterday:appl/biz02/service_count	Delete
key2	yesterday:appl/biz03/service_count	Delete
key3	yesterday:appl/biz04/service_count	Delete

Set the data naming in a consecutive name from key0 to key99. For instance, set like key0, key1,,,. If the name is not linked consecutively, the performance data is displayed by the consecutive names.

```
key0 = yesterday:appl/biz01/service_count
key1 = today:appl/biz01/service_count
```

Figure 13-27:Input Data



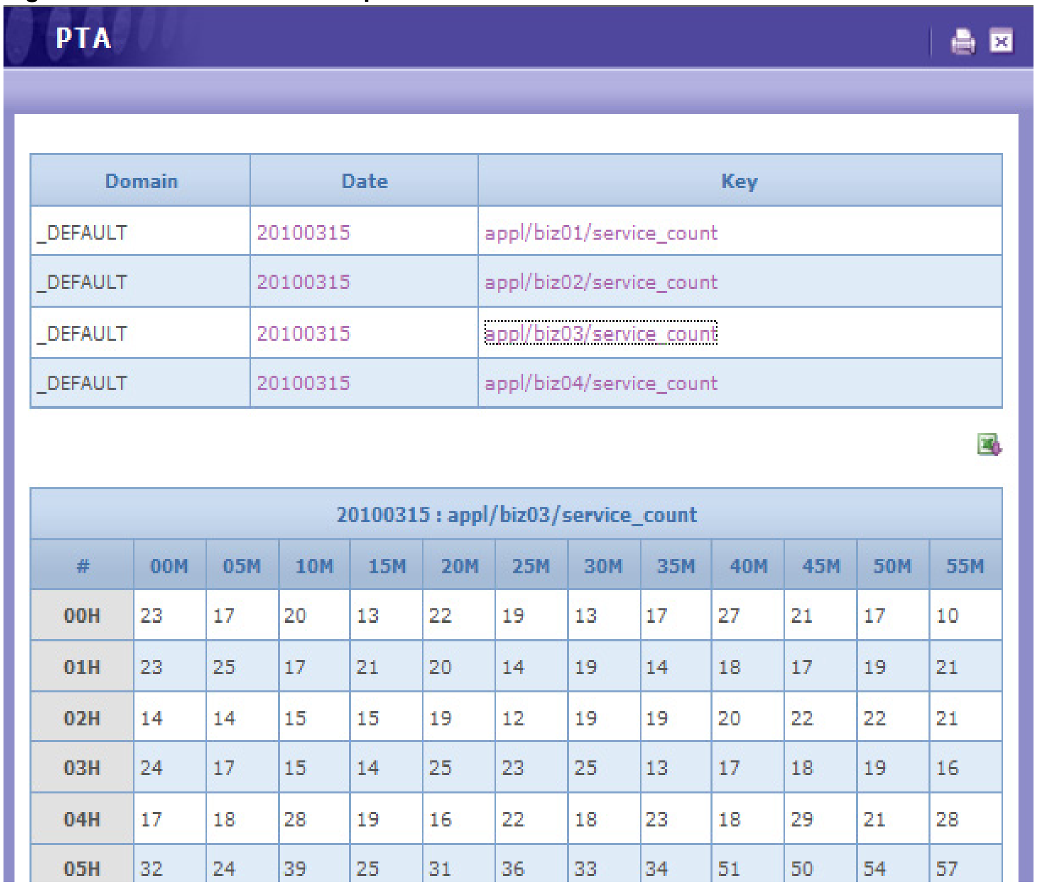
Key	Value	
ENABLE_POPUP	true	Delete
key0	today:ext/comp/biz_h	Delete
key1	today:ext/comp/biz_l	Delete

If you set the following option and input the title of the graph,

```
ENABLE_POPUP = true
```

the detailed view button appears on the right upper area on the real-time PTA graph. If you click this button, you can see the data in real-time and download the data on each graph as an excel file.

Figure 13-28:Real-time PTA Graph



PTA

Domain	Date	Key
_DEFAULT	20100315	appl/biz01/service_count
_DEFAULT	20100315	appl/biz02/service_count
_DEFAULT	20100315	appl/biz03/service_count
_DEFAULT	20100315	appl/biz04/service_count

20100315 : appl/biz03/service_count

#	00M	05M	10M	15M	20M	25M	30M	35M	40M	45M	50M	55M
00H	23	17	20	13	22	19	13	17	27	21	17	10
01H	23	25	17	21	20	14	19	14	18	17	19	21
02H	14	14	15	15	19	12	19	19	20	22	22	21
03H	24	17	15	14	25	23	25	13	17	18	19	16
04H	17	18	28	19	16	22	18	23	18	29	21	28
05H	32	24	39	25	31	36	33	34	51	50	54	57

13.5.3.Creating PTA Data

You can compose the miltiple performance figures and then, crate them into an single performance index in rea-time monitoring environment. Set the server options as follows;

```
pta_items.a1=today:appl/biz01/service_count
pta_items.a2=today:appl/biz02/service_count
pta_items.a3=today:appl/biz03/service_count

pta_items.a4=today:appl/biz04/service_count
pta_items.a5=today:appl/biz05/service_count
pta_items.a6=today:appl/biz06/service_count

pta_items.a7=today:appl/biz07/service_count
pta_items.a8=today:appl/biz08/service_count
pta_items.a9=today:appl/biz09/service_count

pta_compose.biz_h= (a7+a8+a9)/(a1+a2+a3)
pta_compose.biz_l = (a4+a5+a6)/(a1+a2+a3)
```

In order to create composed PTA data, set the item by using a pta_items.xxx option. Next, register the operation through a new key name using pta_compose.xxx.

When you monitor this value, input the following value in the chart option of the user-defined window.

```
key0 = today:ext/comp/biz_h
key1 = today:ext/comp/biz_l
```

13.5.4.PTA Data Management

The JENNIFER implements a unique storing way to analyze various performance trends effectively.

13.5.4.1. Storing Path

The location of data storing can be set by JENNIFER server option.

```
pta_dir=../../data/pta/
```

Each data is managed like the following file based on the path to “pta_dir”.

13.5.4.2. Backup and Restoring

For PTA data, you can execute a backup/restoring process on the JENNIFER's Backup and Restoring page. However, you can also execute a backup/restoring process by copying directory.

13.5.4.3. Storage Period

The default of storage period of JENNIFER server configuration file is one year as follows; However, the period should be reset based on disk utilization.

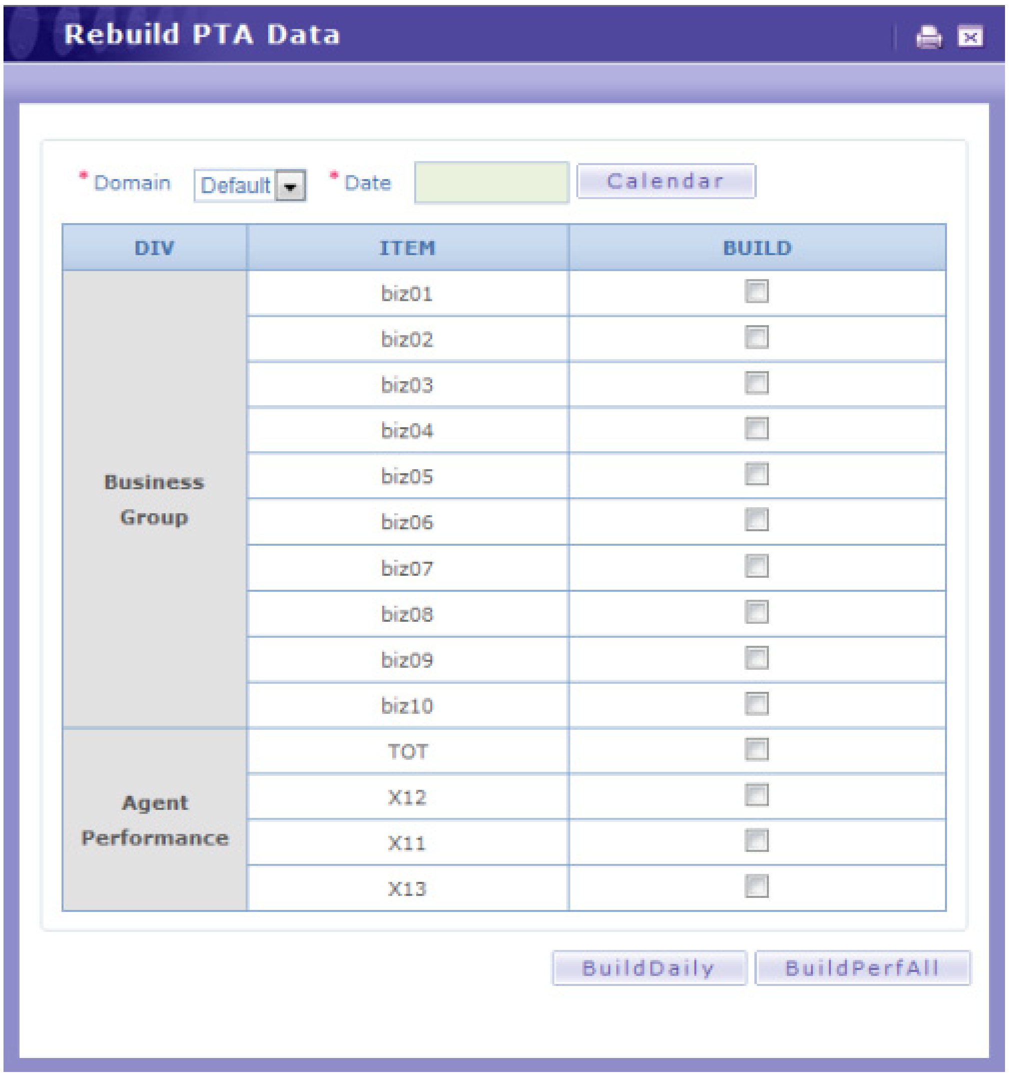
```
time_actor_16 = com.javaservice.jennifer.server.timeactor.CleanerActor
02   PTA MONTH 12
```

You can delete directly the data in the directory.

13.6. PTA Data Build

JENNIFER 4.5 provides PTA rebuild function for the user who wants to upgrade the JENNIFER into v4.5 or recreate the PTA data.To use PTA rebuild, move to the [Tools]-[Rebuild PTA Data] menu.

Figure 13-29:Rebuild PTA Data



DIV	ITEM	BUILD
Business Group	biz01	☐
	biz02	☐
	biz03	☐
	biz04	☐
	biz05	☐
	biz06	☐
	biz07	☐
	biz08	☐
	biz09	☐
	biz10	☐
Agent Performance	TOT	☐
	X12	☐
	X11	☐
	X13	☐

Build Daily

If you select the data and click [BuildDaily], JENNIFER creates the PTA data about the selected date. It copies the data per agent from the PERF_X table of the database and copies the business data from X-View file data.

Build PerfX All

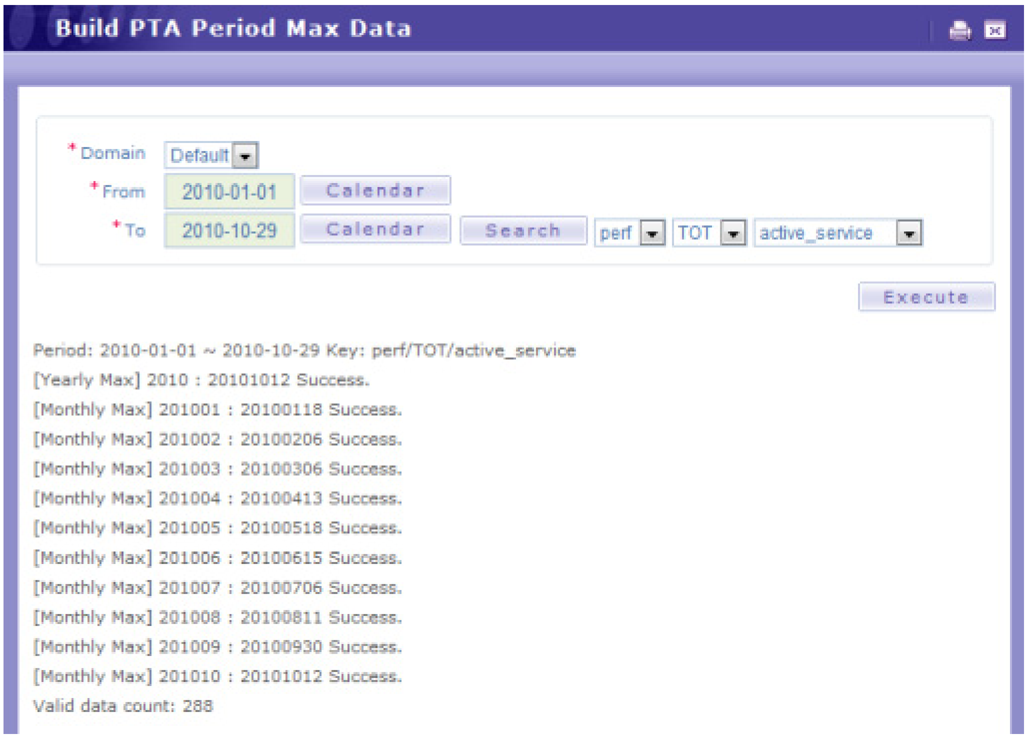
If you do not want to rebuild the PTA data per day, execute [BuildPerfAll] and rebuild the data at once.

If you click this button, JENNIFER copies all performance data from PerfX(performance table per agent) about the agent that you selected and create the PTA data.

13.6.1.Build PTA Period Max Data

When you compare the performance data, you may need not only the data about the certain date but also the maximum value of the month or year. In this case, execute [Build PTA Period MAX Data].

Figure 13-30:Build PTA Period Max Data



* Domain: Default
* From: 2010-01-01
* To: 2010-10-29
perf / TOT / active_service

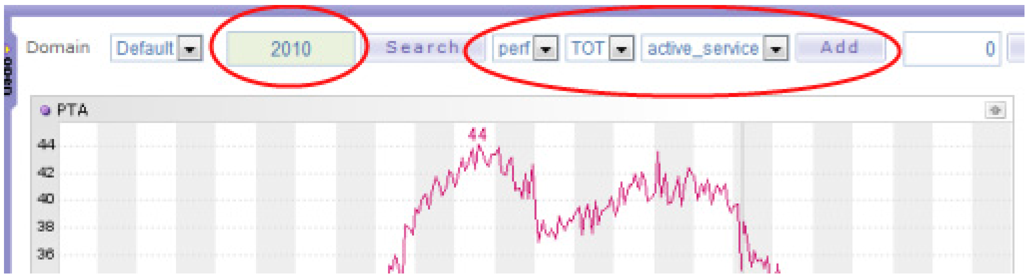
```
Period: 2010-01-01 ~ 2010-10-29 Key: perf/TOT/active_service
[Yearly Max] 2010 : 20101012 Success.
[Monthly Max] 201001 : 20100118 Success.
[Monthly Max] 201002 : 20100206 Success.
[Monthly Max] 201003 : 20100306 Success.
[Monthly Max] 201004 : 20100413 Success.
[Monthly Max] 201005 : 20100518 Success.
[Monthly Max] 201006 : 20100615 Success.
[Monthly Max] 201007 : 20100706 Success.
[Monthly Max] 201008 : 20100811 Success.
[Monthly Max] 201009 : 20100930 Success.
[Monthly Max] 201010 : 20101012 Success.
Valid data count: 288
```

Select the period to search the maximum value and click the data.

```
Period: 2010-01-01 ~ 2010-10-29 Key: perf/TOT/active_service
[Yearly Max] 2010 : 20101012 Success.
[Monthly Max] 201001 : 20100118 Success.
[Monthly Max] 201002 : 20100206 Success.
[Monthly Max] 201003 : 20100306 Success.
[Monthly Max] 201004 : 20100413 Success.
[Monthly Max] 201005 : 20100518 Success.
[Monthly Max] 201006 : 20100615 Success.
[Monthly Max] 201007 : 20100706 Success.
[Monthly Max] 201008 : 20100811 Success.
[Monthly Max] 201009 : 20100930 Success.
[Monthly Max] 201010 : 20101012 Success.
Valid data count: 288
```

In the above example, search the max data about perf/TOT/active_service and input the "2010". After that, search the max data about each month and copy it into "2010XX".

You can seach this data if you input "2010"on the PTA screen.



13.6.2.Scheduler for PTA Period Max Data

You can update the max value per month everyday using TimeActor.

```
time_actor_22 = com.javaservice.jennifer.server.timeactor.PtaFindMax
02   perf/TOT/active_service
```

As you input like above in the JENNIFER server configurations, JENNIFER copies the daily data about max data of this month/year of perf/TOT/active_service at every day of 2:00 am.

13.7. J-SCORE

J-SCORE creates the standard performance index by composing the each different performance related data and converting into a single integrated figure.

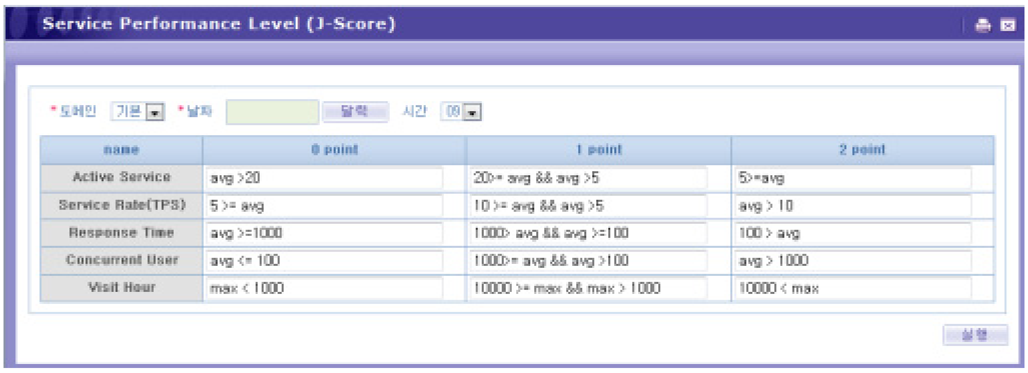
Converting the different figures into one single standard performance data is so much meaningful work. Under the heterogeneous multi-system environment, JENNIFER creates the J-SCORE by understanding the differences between systems and revising them appropriately. J-SCORE let all system stakeholders manage the system performance with a consistent view.

13.8. How to Execute

J-SCORE is simple to execute. Open the window and select the date and time. You can see the result after executing.

- Click the [Tools | J-Score] menu on the left of JENNIFER dashboard and you will see the belowing window.

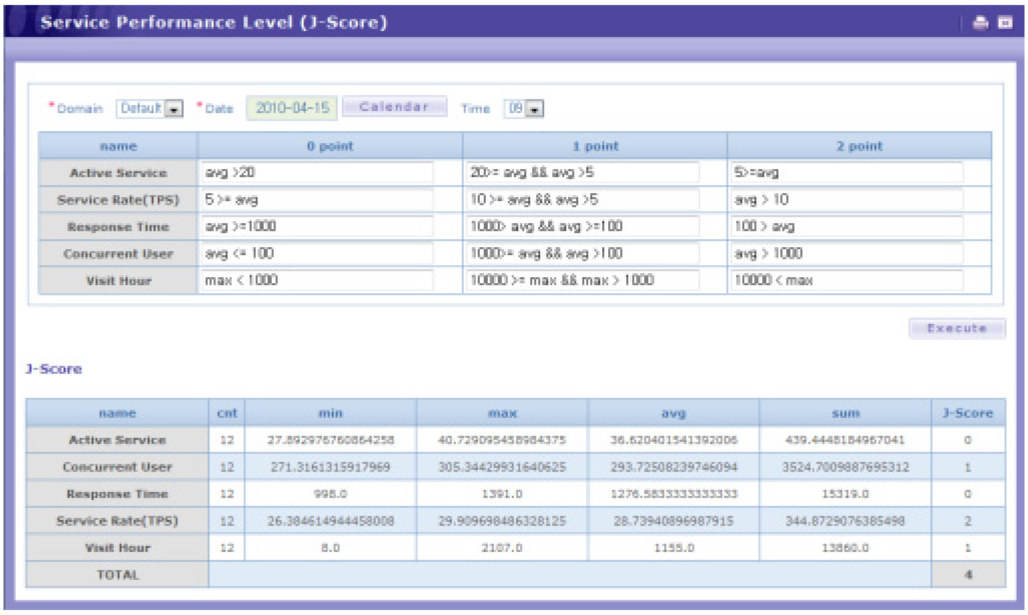
Figure 13-31:Executing J-Score 1



name	0 point	1 point	2 point
Active Service	avg >20	20>= avg && avg >5	5>=avg
Service Rate(TPS)	5 >= avg	10 >= avg && avg >5	avg > 10
Response Time	avg >=1000	1000> avg && avg >=100	100 > avg
Concurrent User	avg <= 100	1000>= avg && avg >100	avg > 1000
Visit Hour	max < 1000	10000 >= max && max > 1000	10000 < max

- Input the date and click the “Execute” button. Now, you will see the result.

Figure 13-32:Executing J-Score 2


Service Performance Level (J-Score)

*Domain Default *Date 2010-04-15 Time 09

name	0 point	1 point	2 point
Active Service	avg >20	20>= avg && avg >5	5>=avg
Service Rate(TPS)	5 >= avg	10 >= avg && avg >5	avg > 10
Response Time	avg >=1000	1000> avg && avg >=100	100 > avg
Concurrent User	avg <= 100	1000>= avg && avg >100	avg > 1000
Visit Hour	max < 1000	10000 >= max && max > 1000	10000 < max

J-Score

name	cnt	min	max	avg	sum	J-Score
Active Service	12	27.892976760864258	40.729095458984375	36.620401541392006	439.4448184967041	0
Concurrent User	12	271.3161315917969	305.34429931640625	293.72508239746094	3524.7009887695312	1
Response Time	12	998.0	1391.0	1276.5833333333333	15319.0	0
Service Rate(TPS)	12	26.384614944458008	29.909698486328125	28.73940896987915	344.8729076385498	2
Visit Hour	12	8.0	2107.0	1155.0	13860.0	1
TOTAL						4

13.8.1.How to Creating the Configuration File and Registration

Register the performance index and operation that are be used in J-SCORE.

```
JENNIFER_SERVER/bin/score.xml
```

The sample configuration way is as follows;

```
<?xml version="1.0" encoding="utf-8" ?>
<jscore>
<rule>
  <id>perf/TOT/active_service</id>
  <name>Active Service</name>
  <score  point="0"><![CDATA[   avg >20             ]]></score>
  <score  point="1"><![CDATA[   20>= avg && avg >5 ]]></score>
  <score  point="2"><![CDATA[   5>=avg              ]]></score>
</rule>

...

<rule>
  <id>perf/TOT/visit_hour</id>
  <name>Visit Hour</name>
  <score  point="0"><![CDATA[   max < 1000            ]]></score>
  <score  point="1"><![CDATA[   10000 >= max && max > 1000 ]]></score>
  <score  point="2"><![CDATA[   10000 < max              ]]></score>
</rule>

</jscore>
```

Notice: Every data registered in <color blue> PTA is can be used in J-SCORE.

To creating the configuration file is as follows;

- Several <rule> can be defined in <jscore>.
- <id> and <name> tags are must be defined per <rule>.
- More than one must be exist in <score>.
- Point must be set in <score> as a property, but you can not set the <score> which has same points in a rule.
- One formula is created for each <score> and the result value from the formula should be Boolean.

- Use the parameter when you create the formula and the available parameter is as follows;

Table 13-20: Parameter Data

Name	Description
cnt	The # of 0 among the selected time-zone, the number of data in an hour is 12.(fixed)
max	Maximum value among the selected time-zone
min	Minimum value among the selected time-zone
avg	Average value among the selected time-zone
sum	Sum value among the selected time-zone

If you do not want to place the configuration file in JENNIFER_SERVER/bin/score.xml, set the following option in the JENNIFER SERVER.

```
score_filename=/jennifer/server/myscore.xml
```

13.9. Report(WebReport)

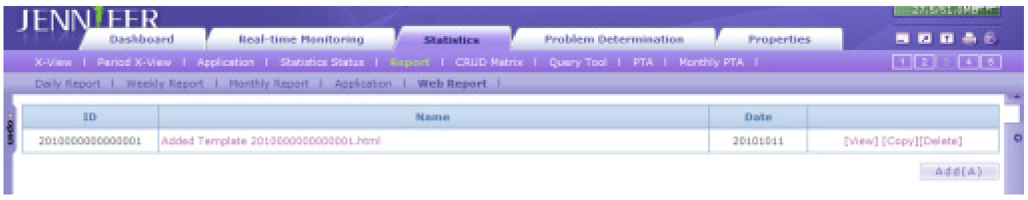
JENNIFER's WebReport is a pure web-based reporting module. It helps you to output the various performance data collected by JENNIFER easily.

- Support web-baed WYSIWYG, design and creat a report without additional solution.
- To design a report, JENNIFER 4.5 adopted Component concept. It means the report item can be separately modified and distributed. In addition, the report can be created and modified by user.
- Create and modify a report only using html/css/javascript. Server module controls the chart contents, but the every output materials are processed based on web. Newly created JENNIFER report is much more flexible and simple to use. The report is output only through a web browser.

13.9.1.Quick Start

After installing the JENNIFER 4.5, select the [Statistics/Report/WebReport] menu. Then, you will see a list as below.

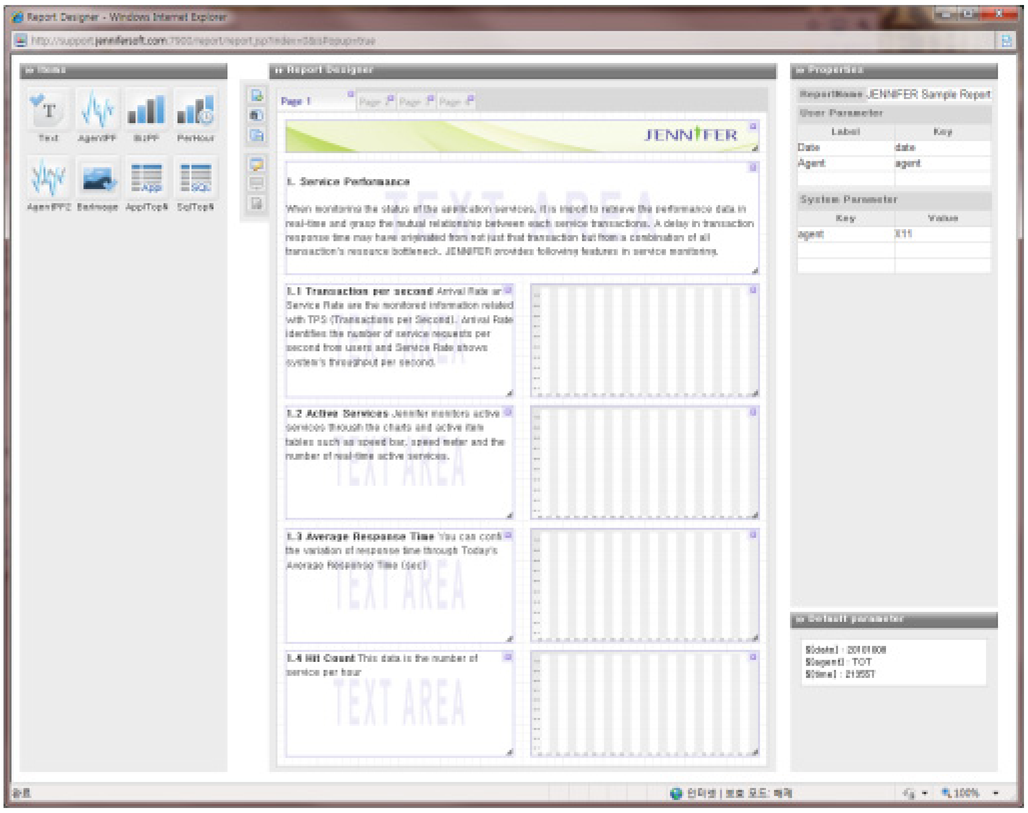
Figure 13-33:List


JENNIFER
Dashboard
Real-time Monitoring
Statistics
Problem Determination
Properties
X-View | Period X-View | Application | Statistics Status | Report | CRUD Matrix | Query Tool | PTA | Monthly PTA
Daily Report | Weekly Report | Monthly Report | Application | Web Report
Add(A)

ID	Name	Date	
2010000000000001	Added Template 2010000000000001.html	20101011	[View] [Copy][Delete]

A sample report is included in JENNIFER 4.5. If you open a smaple on the list, you can see a report template.

Figure 13-34:Report Template



1. Service Performance

When monitoring the status of the application services, it is import to retrieve the performance data in real-time and grasp the mutual relationship between each service transactions. A delay in transaction response time may have originated from not just that transaction but from a combination of all transaction's resource bottleneck. JENNIFER provides following features in service monitoring.

1.1 Transaction per second Arrival Rate and Service Rate are the monitored information related with TPS (Transactions per Second). Arrival Rate identifies the number of service requests per second from users and Service Rate shows system's throughput per second.

1.2 Active Services Jennifer monitors active services through the charts and active item tables such as speed bar, speed meter and the number of real-time active services.

1.3 Average Response Time You can confirm the variation of response time through Today's Average Response Time (sec)

1.4 Hit Count This data is the number of service per hour

ReportName JENNIFER Sample Report

User Parameter

Label	Key
Date	date
Agent	agent

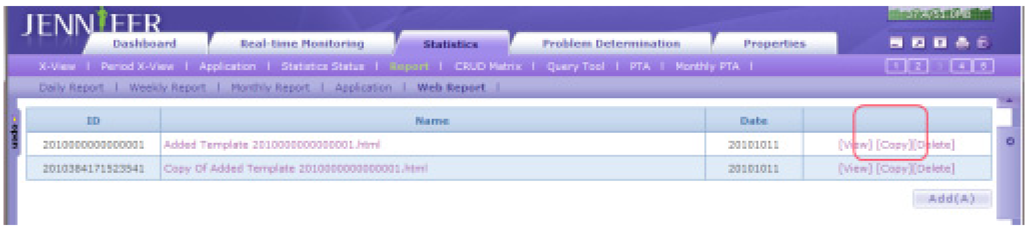
System Parameter

Key	Value
agent	X11

$[date] : 20101008
$[agent] : TOT
$[time] : 213557

After modifying the original template, you can not restore the template. So, it is safe to copy the original one and modify the copy one. If you click the [copy] on the list, you can copy the template.

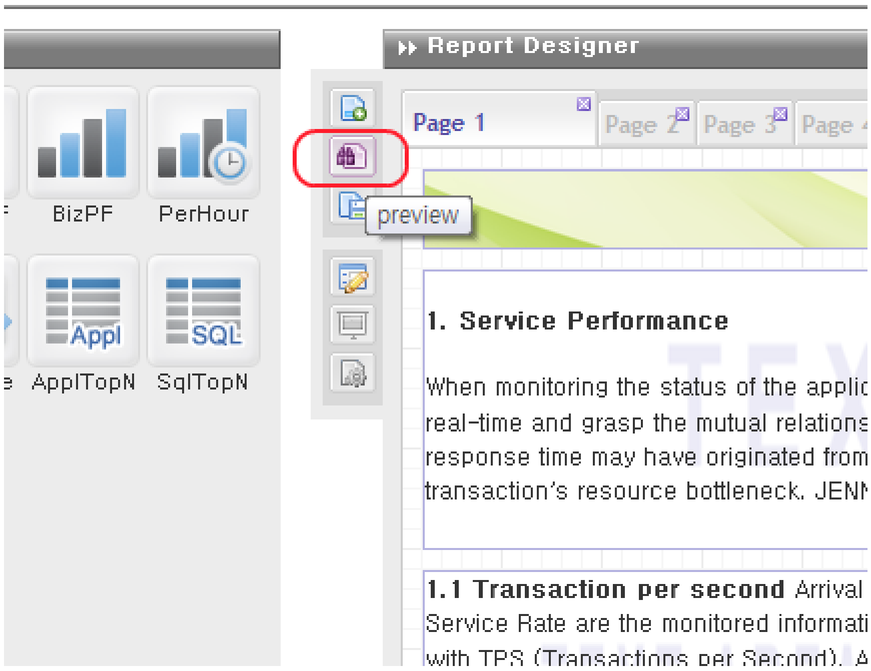
Figure 13-35:Copy the Template


JENNIFER
Dashboard
Real-time Monitoring
Statistics
Problem Determination
Properties
X-View | Period X-View | Application | Statistics Status | Report | CRUD Matrix | Query Tool | PTA | Monthly PTA |
Daily Report | Weekly Report | Monthly Report | Application | Web Report |
Add(A)

ID	Name	Date	
2010000000000001	Added Template 2010000000000001.html	20101011	[View] [Copy][Delete]
2010384171523541	Copy Of Added Template 2010000000000001.html	20101011	[View] [Copy][Delete]

Click the copied template and open it.

Figure 13-36:Template Open



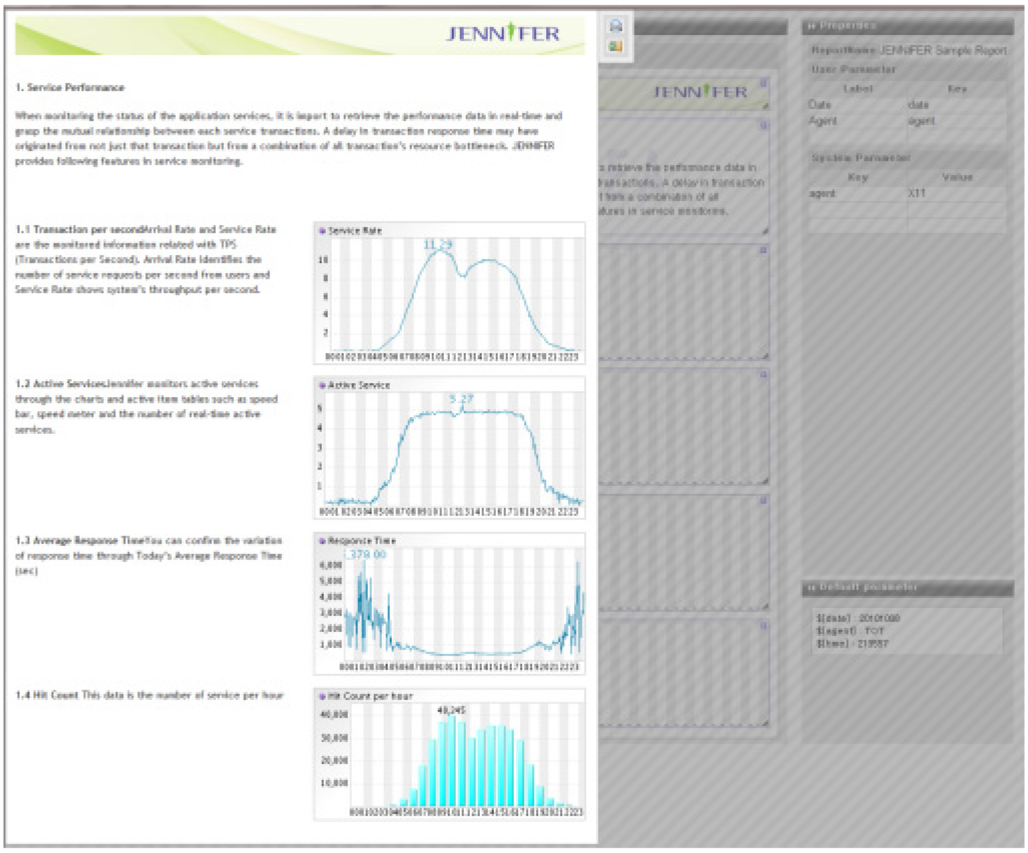
1. Service Performance

When monitoring the status of the applic
real-time and grasp the mutual relations
response time may have originated from
transaction's resource bottleneck. JENN

1.1 Transaction per second Arrival
Service Rate are the monitored informati
with TPS (Transactions per Second). A

If you click the [Preview] button, you can see the output image about the current page.

Figure 13-37:Report Output



1. Service Performance

When monitoring the status of the application services, it is import to retrieve the performance data in real-time and grasp the mutual relationship between each service transactions. A delay in transaction response time may have originated from not just that transaction but from a combination of all transaction's resource bottleneck. JENNIFER provides following features in service monitoring.

1.1 Transaction per secondArrival Rate and Service Rate are the monitored information related with TPS (Transactions per Second). Arrival Rate identifies the number of service requests per second from users and Service Rate shows system's throughput per second.

1.2 Active ServicesJennifer monitors active services through the charts and active item tables such as speed bar, speed meter and the number of real-time active services.

1.3 Average Response TimeYou can confirm the variation of response time through Today's Average Response Time (sec)

1.4 Hit Count This data is the number of service per hour

Properties

ReportName JENNIFER Sample Report

User Parameter

Label	Key
Date	date
Agent	agent

System Parameter

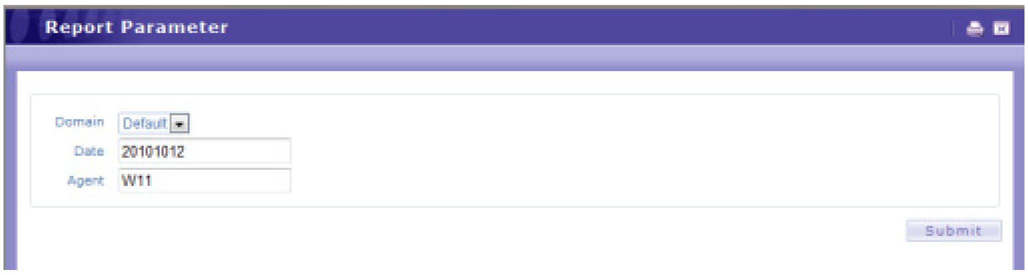
Key	Value
agent	X11

Default parameter

$[date] : 20101030
$[agent] : TOT
$[time] : 213537

If you click the [View] button on the previous page, you can see the output image as follows.

Figure 13-38:Input Window

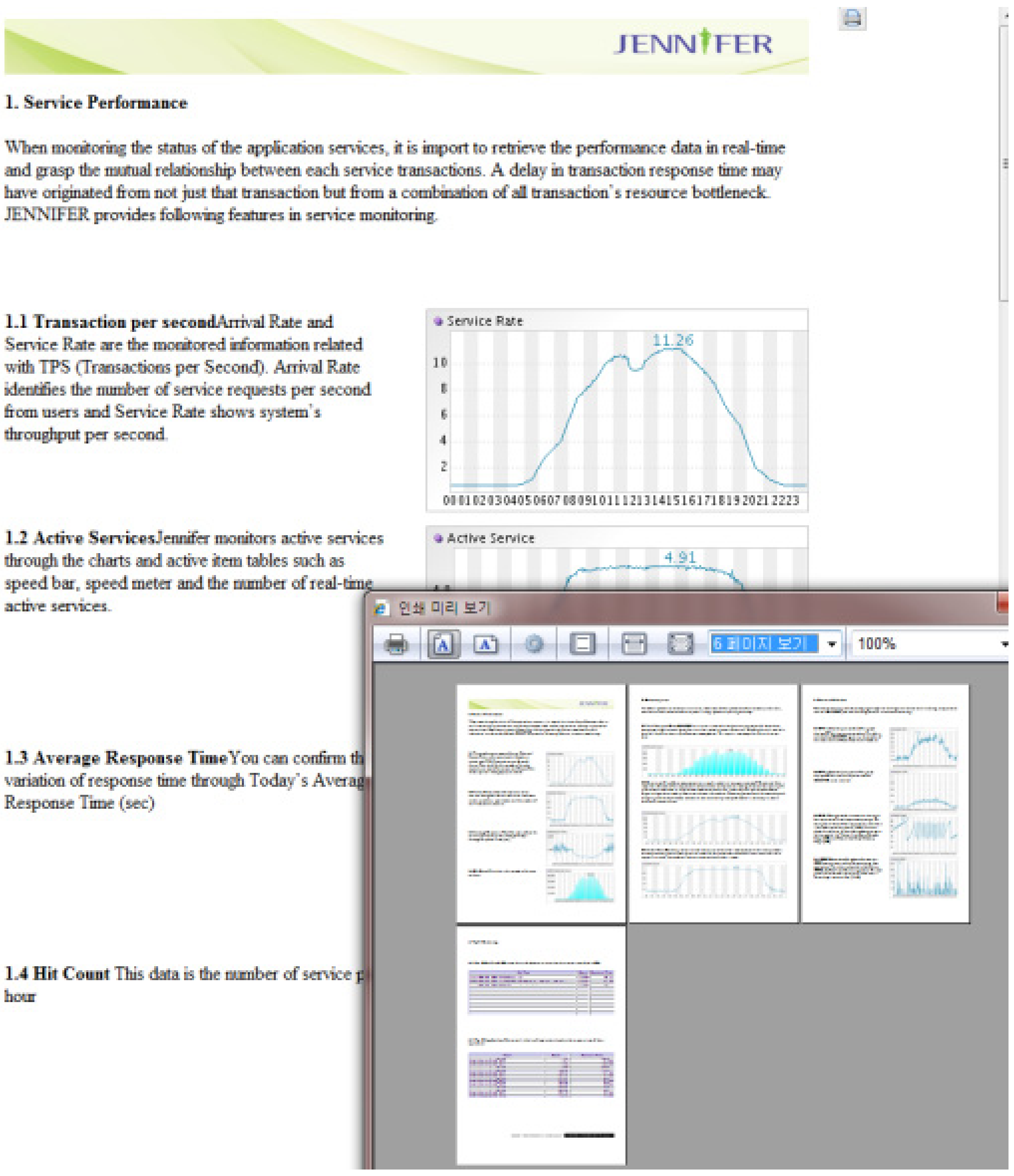


Report Parameter

Domain	Default
Date	20101012
Agent	W11

If you click the [Submit] button after input a parameter, you can see the print screen as follows.

Figure 13-39:Print Window



JENNIFER

1. Service Performance

When monitoring the status of the application services, it is import to retrieve the performance data in real-time and grasp the mutual relationship between each service transactions. A delay in transaction response time may have originated from not just that transaction but from a combination of all transaction's resource bottleneck. JENNIFER provides following features in service monitoring.

1.1 Transaction per secondArrival Rate and Service Rate are the monitored information related with TPS (Transactions per Second). Arrival Rate identifies the number of service requests per second from users and Service Rate shows system's throughput per second.

1.2 Active ServicesJennifer monitors active services through the charts and active item tables such as speed bar, speed meter and the number of real-time active services.

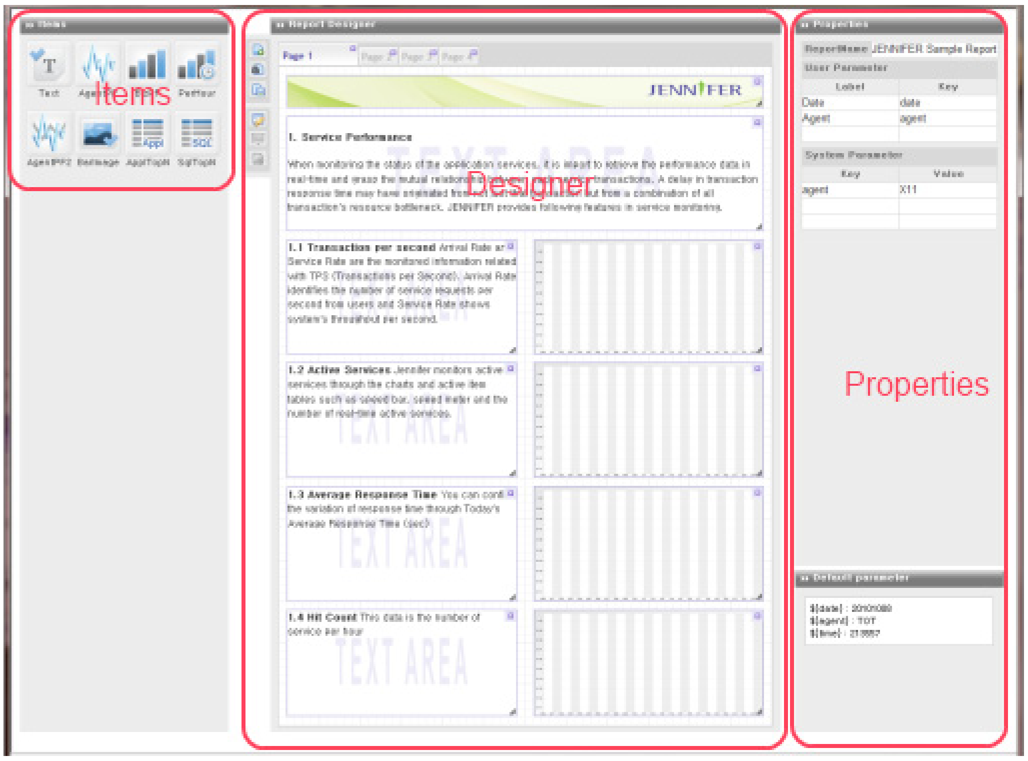
1.3 Average Response TimeYou can confirm th variation of response time through Today's Averag Response Time (sec)

1.4 Hit Count This data is the number of service p hour

13.9.2.Screen Description

Design area of JENNIFER's WebReport template can be divided into three main areas.

Figure 13-40:Design Window



13.9.2.1. Items

This area is a report component that creates charts and tables. The report item can be created and distributed separately. The report template can be added arbiturally when the JENNIFER is upgraded. Also, user can create and add an item, if he want.

13.9.2.2. Report Designer

In this area, user creates report template. The template is designed by page. Drag an item and edit it on the blank page. If you click one item, you can modify the properties about the component in the reight part of Properties area.

13.9.2.3. Properties

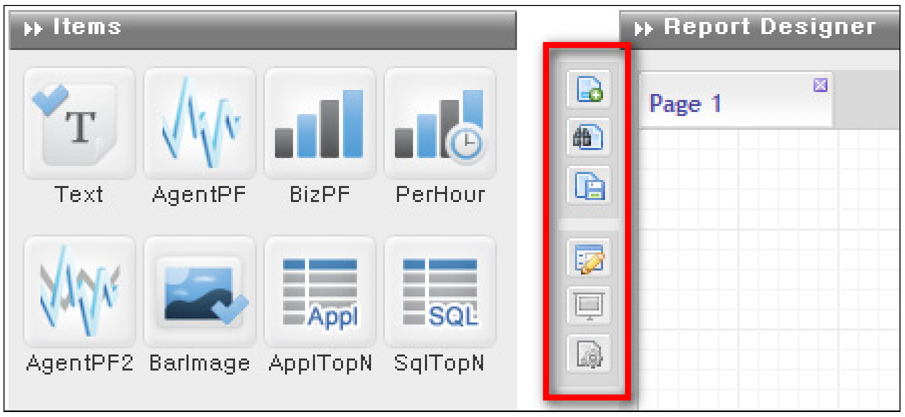
In this area, user can set the properties value about each component.

- Report Properties : It can be used in all report types and set the parameter additionally. Set the properties by clicking the tap area indicating page name.
- Page Properties : It can be used in one page and set the properties by clicking the tap area indicating page name.

- Item Properties : Input and modify the properties used by one item. Set the properties by clicking each item.

13.9.2.4. WebReport Menu

Figure 13-41:WebReport Menu

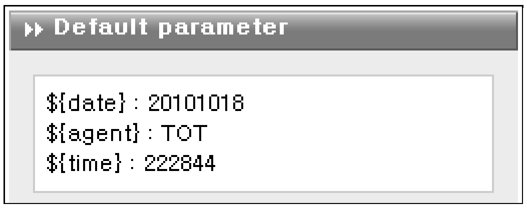


- Add page : Add a new page.
- Preview : Apply the output image on the current page and preview the output image.
- Save : Save the created report template.
- Output : Excute the currently created report template as a previous step(before output). If you set the user input parameter, input page will be executed for the input parameter.
- Presentation(PT) : Transit the report template into PT(presentation) mode.(To be implemented)
- Revision : Using report designer, revise the proportion between template and output material that is to be released.(To be implemented)

13.9.3.Setting the Input Parameter

The WebReport can process parameter for the properties. Parameter is mostly used to designate the properties which are variable. You can set the parameter value by clicking the desinger area.

Figure 13-42:Setting for the Input Parameter


Default parameter

${date} : 20101018
${agent} : TOT
${time} : 222844

There are three types of parameters which is used in WebReport.

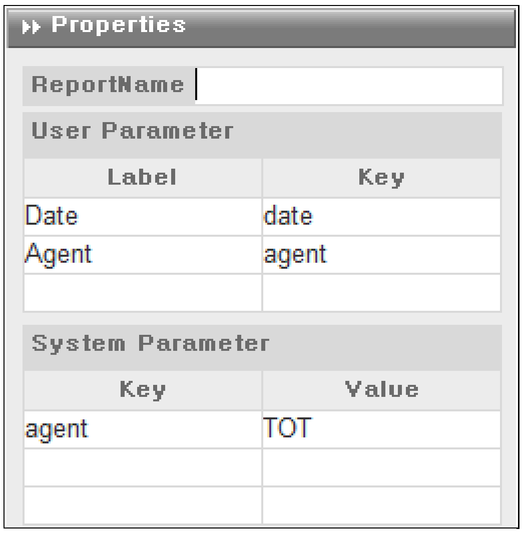
- Default Parameter : Defined default parameters are ${agent}, ${date} and ${time}.
- System Parameter : Defined System parameter is used in executing “Output”.
- User Parameter : User parameter is can be input and modified by user in proecessing “Output”.

If you set the same names of parameters, the parameter priority is as [User parameter→ System parameter→Default parameter].

To define parameter, input the lable name in the Label area for user parameter and define the key(figure or english letter)which is to be used in item in the Key area. (If you define it like the default parameter or system parameter in the key, the default paremeter is defined as a key value.)

You will see the not value but label in the user parameter because it is processed by user.

To define the system parameter, define the key and input the value which is ablut key. (Mostly invariable.)


Properties

ReportName

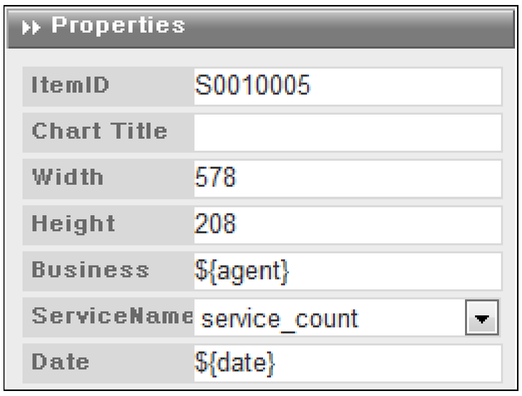
User Parameter

Label	Key
Date	date
Agent	agent

System Parameter

Key	Value
agent	TOT

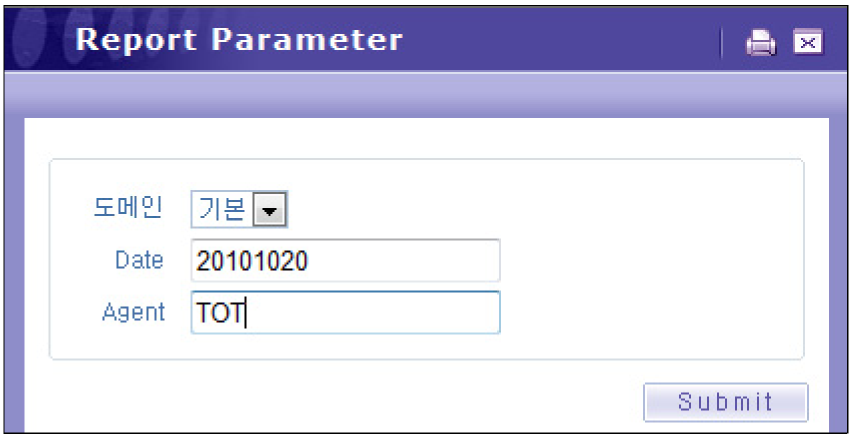
The defined parameter can be used as “${key}” type in the item properties.



ItemID	S0010005
Chart Title	
Width	578
Height	208
Business	${agent}
ServiceName	service_count
Date	${date}

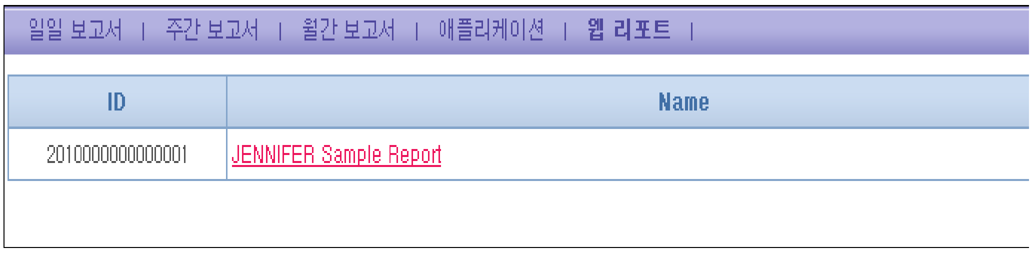
The setting parameter - is displayed by clicking “Output” button as follows.(You have to click this button after saving the report.)

In case of user parameter, the value can be input from user because the input form has changed suitable for “Output” area.



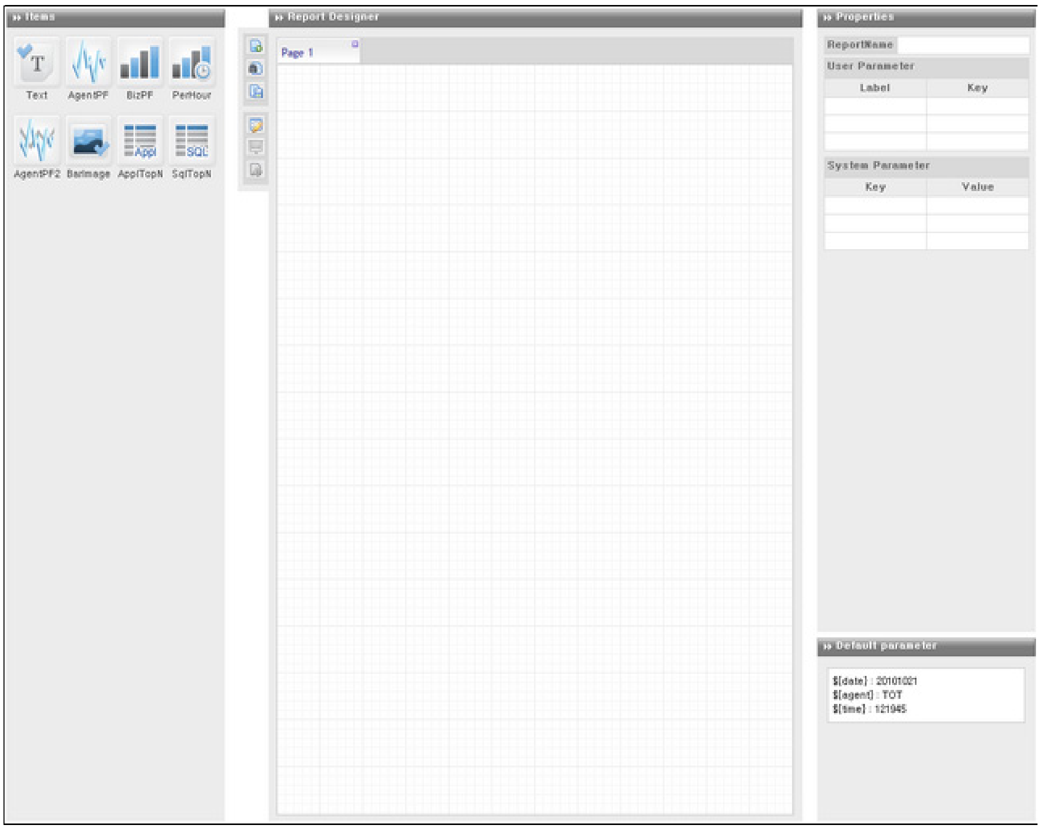
도메인	기본
Date	20101020
Agent	TOT

13.9.4.Create Report


일일 보고서 | 주간 보고서 | 월간 보고서 | 애플리케이션 | 웹 리포트 |

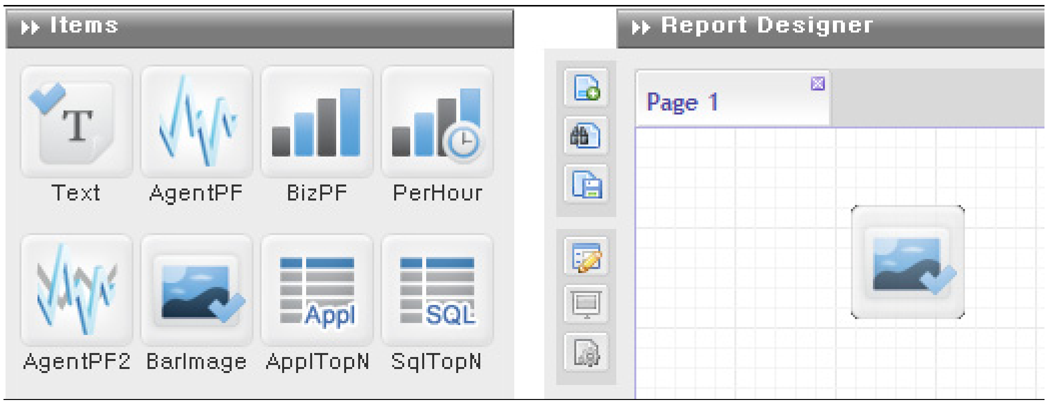
ID	Name
2010000000000001	JENNIFER Sample Report

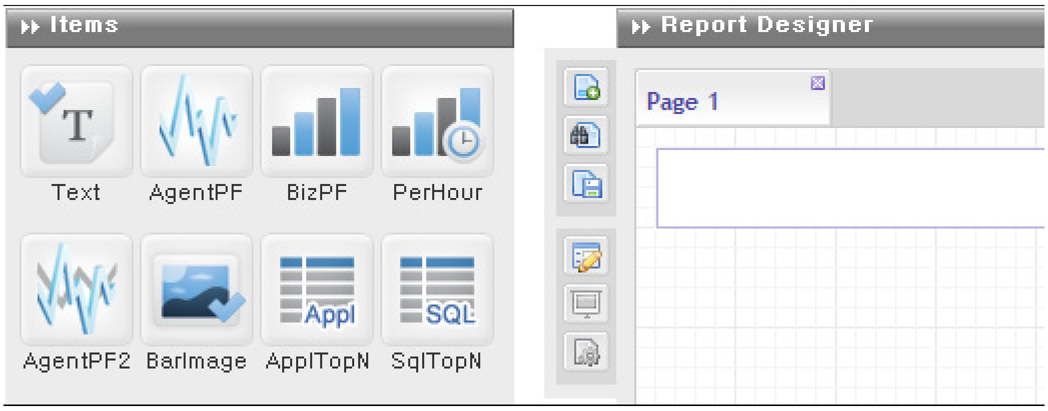
Select the [Add] on the report template list and start to create a template through designer.



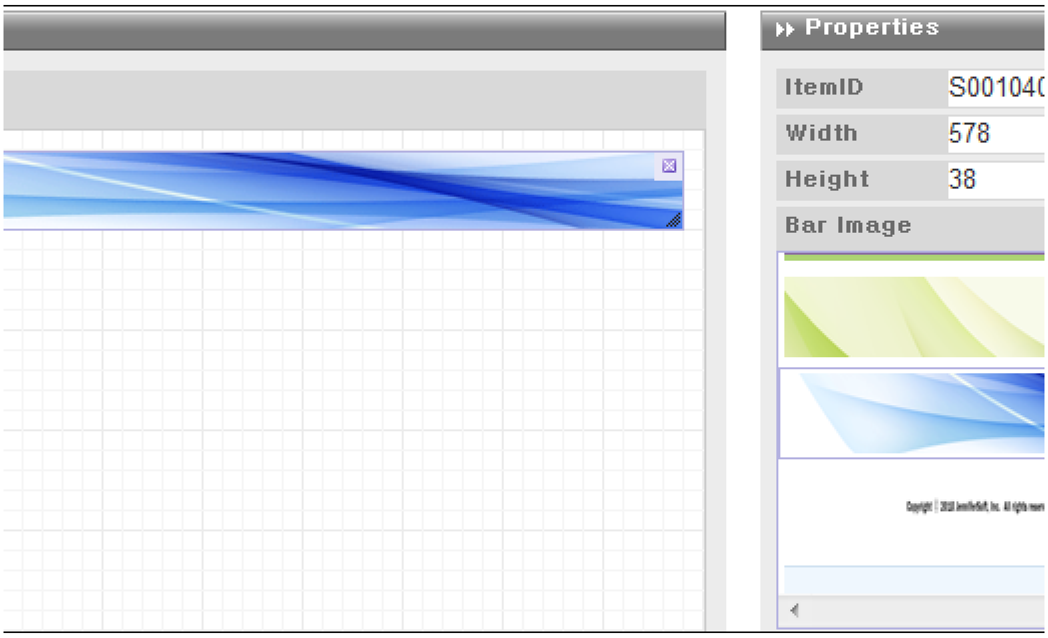
$[date] : 20101021
$[agent] : TOT
$[time] : 121945

Drag a item in the design area(page). On the below example, BarImage item is deployed. .



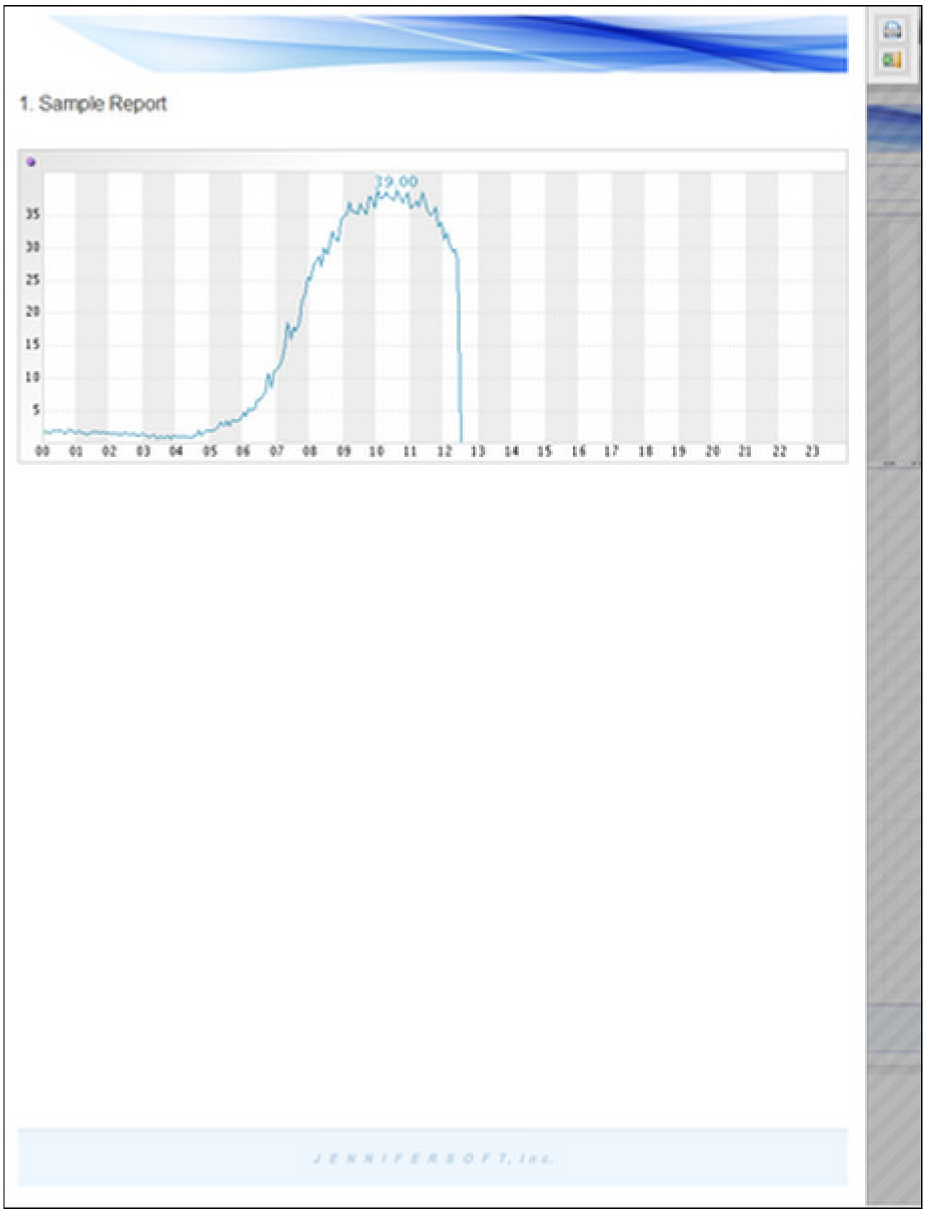


Click the deployed item and set the properties..



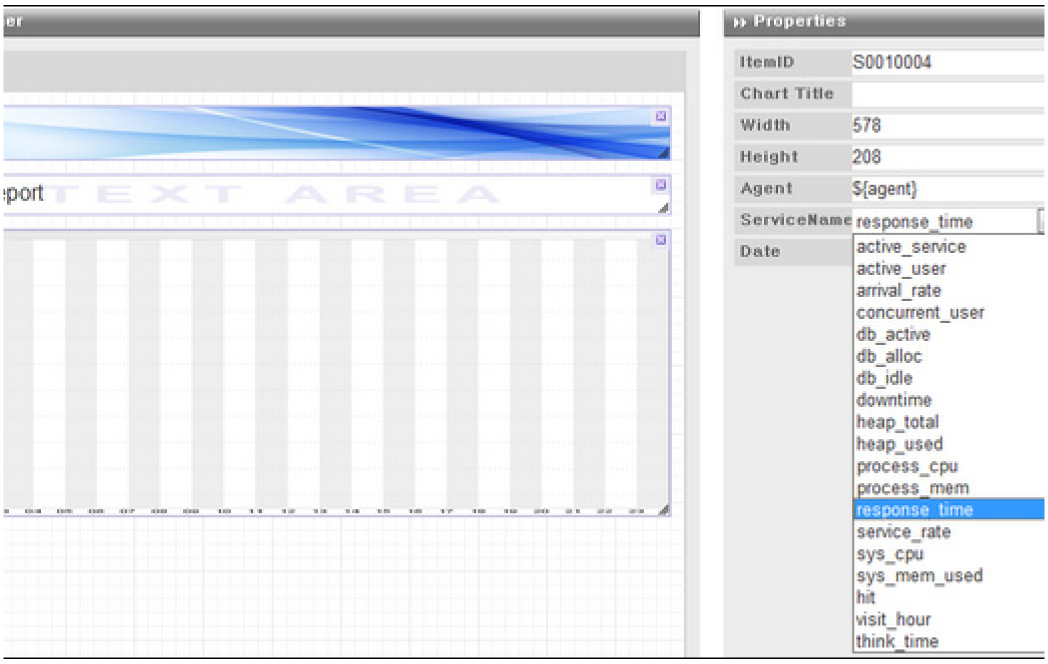
ItemID	S00104(
Width	578
Height	38
Bar Image	

Add more items, then click the (Preview) button. Check the items are well deployed upon your favor.



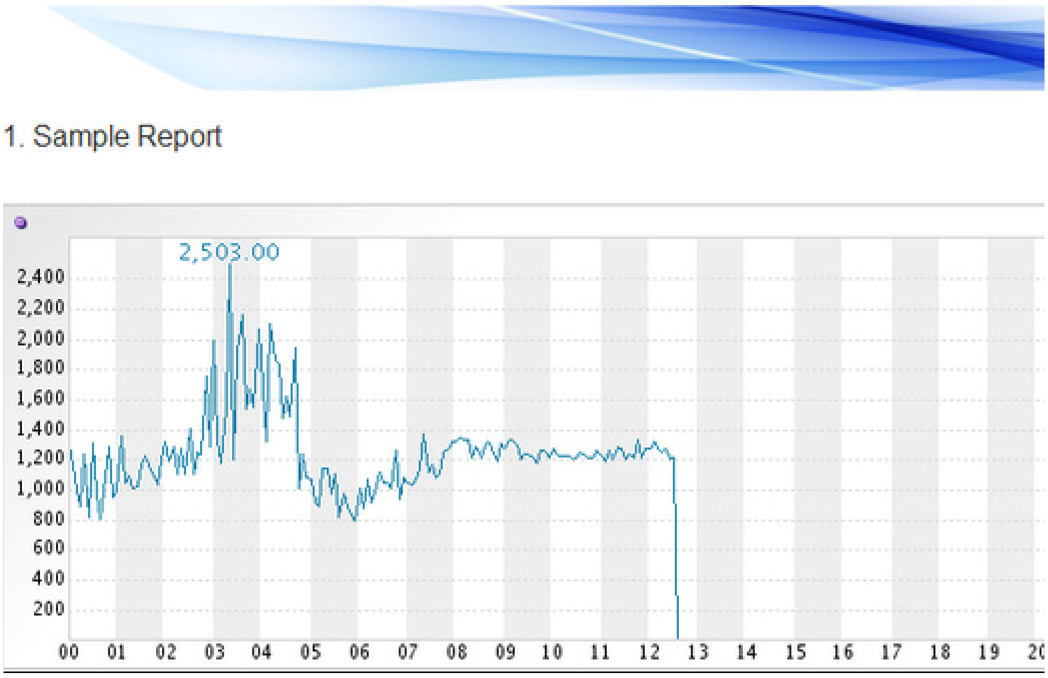
If you try to preview right after adding the item, the default values of the item properties are applied. Select the added item (AgentPF) and change the properties.

(In the preview screen, you can output the template by page. You can print out current selected page.)



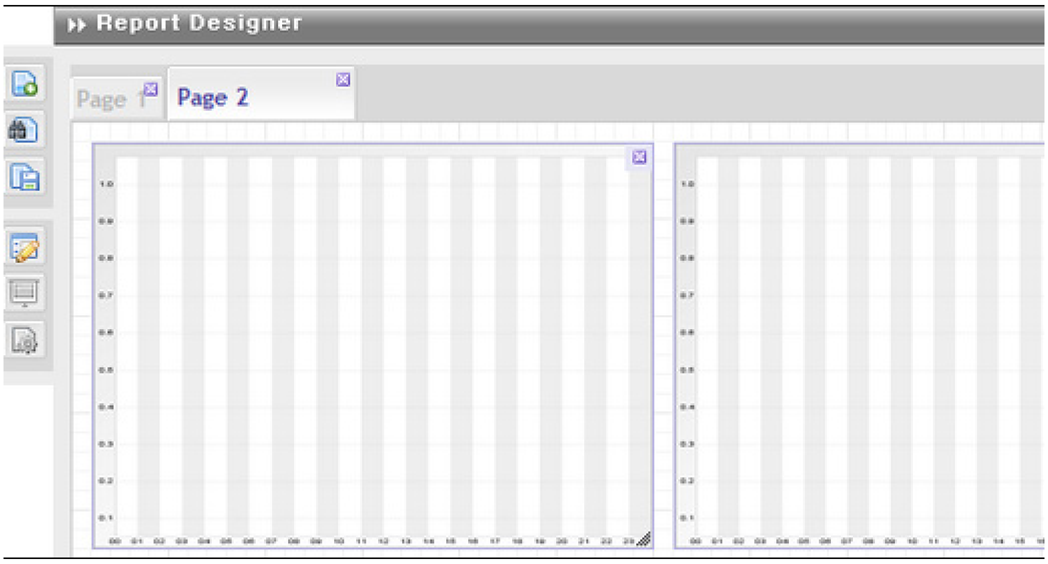
ItemID	S0010004
Chart Title	
Width	578
Height	208
Agent	${agent}
ServiceName	response_time
Date	

Check the changed value is applied properly.

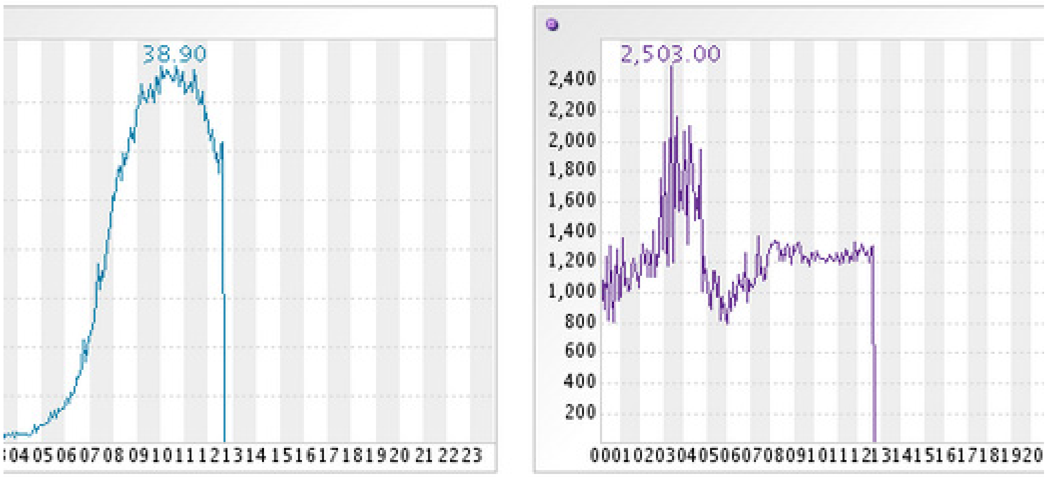


Click the [Add] button and add a page.

(You can add pages by 10 as a maximum.)

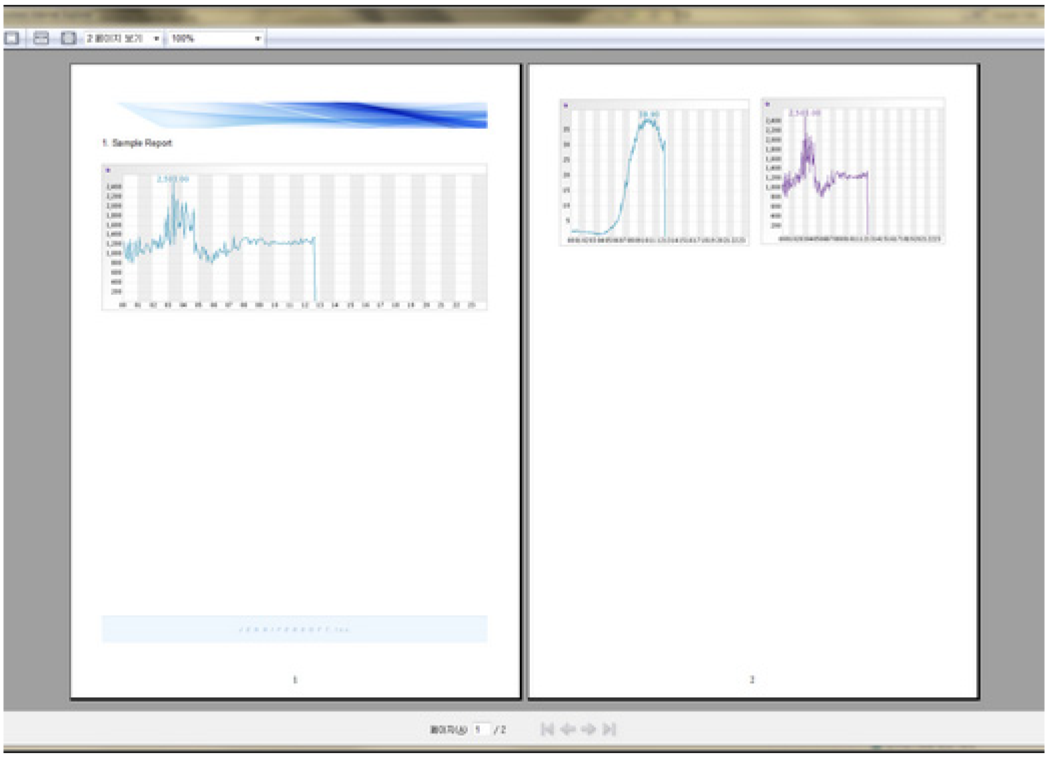


As the previous page, deploy items that you want and create a page by setting properties.



After creating report template, click the [Save] button and save the template.

Execute the created template by clicking the [Output] button. Check the result of template and print out.



13.9.5.Item Description

Added items are 8 as a default.(The item would be added continuously according to the properties.)

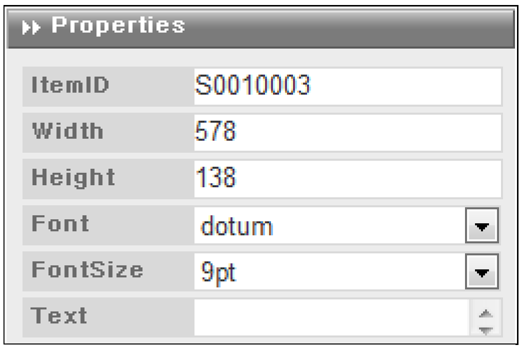
Items have common properties such as ItemID, Width and Height.

ItemID matches current properties to the item and Width and Height control the item size to the screen.

13.9.5.1. Text

Text item is used to input the text on the template. You can input the Text and Html, but it is recommended to use simple Html in case you input Html.

Text item has features as follows.



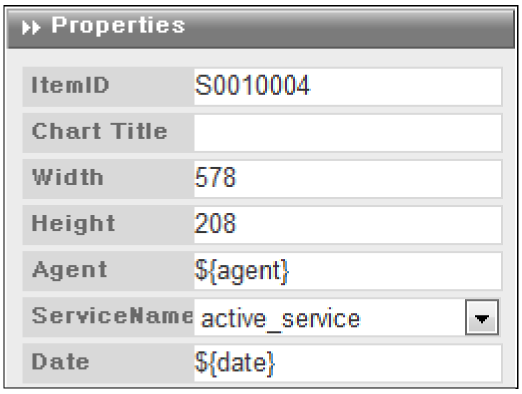
ItemID	S0010003
Width	578
Height	138
Font	dotum
FontSize	9pt
Text	

- Font : Set the Text font.
- FontSize : Set the Text size.
- Text : Input the output text on the page.

13.9.5.2. AgentPF

The performance data per agent is displayed as a chart.

AgentPF items has features as follows.



ItemID	S0010004
Chart Title	
Width	578
Height	208
Agent	${agent}
ServiceName	active_service
Date	${date}

- ChartTitle : This is title of chart.
- Agent : This is agent of data to be searched.
- ServiceName : Select the data to be searched.
- Date : Input the date of data to be searched.

Selectable ServiceName

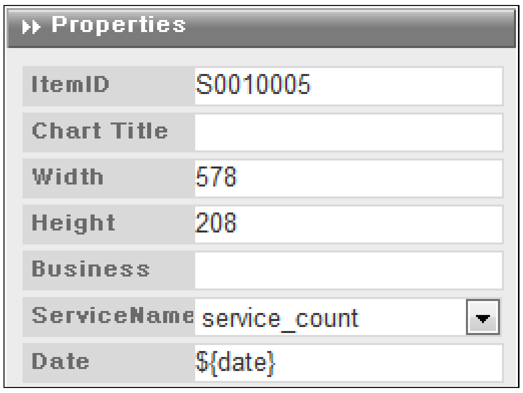
- active_service : Active server per agent
- active_user : Active user per agent
- arrival_rate : Arrival Rate per agent

- concurrent_user : Concurrent user per agent
- db_active : Active DB connection per agent(Except TOT)
- db_alloc : Active DB connection per agent
- db_idle : Idle DB connection per agent
- downtime : Downtime per 5 mins/agent(Max 300 sec)
- heap_total : Maximum HEAP memory per agent
- heap_used : HEAP memory utilization per agent
- process_cpu : Process CPU utilization per agent(%)
- process_mem : Process memory utilization per agent(MB)
- response_time : Average response time per agent(ms)
- service_rate : TPS per agent(TPS)_
- sys_cpu : Total CPU utilization executed by agent
- sys_mem_used : Total system memory utilization executed by agent
- hit : Hits per 5 mins/agent
- visit_hour : Visitors per hour/agent
- think_time : Think Time per agent

13.9.5.3. BizPF

BizPF item outputs the business performance data through bar chart.

BizPF item has features as follows.



ItemID	S0010005
Chart Title	
Width	578
Height	208
Business	
ServiceName	service_count
Date	${date}

- ChartTitle : This is a title of chart.
- Business : Set the business of data to be searched.

- ServiceName : Select the data to be searched.
- Date : The date of data to be searched.

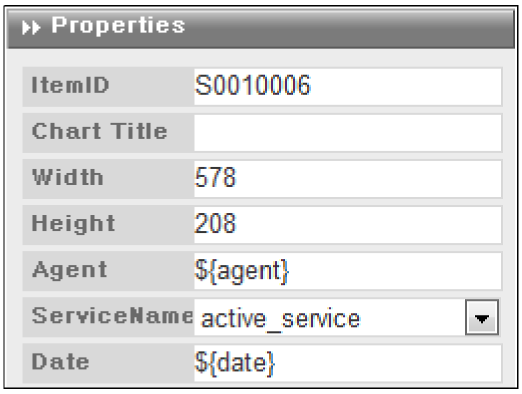
Selectable ServiceName

- service_count : (SUM) Hits per 5 mins/business
- service_time : (Average)Response time per business(ms)
- sql_time : (Average)SQL execution time per business(ms)
- error_count : (SUM)Errors per 5 mins/business
- etx_time : (Average)Execution time of external transaction per business(ms)
- sla_fail : (SUM)Exceeded services from SLA data(set in business group) per 5 mins/ business
- client_time : (Average)Final user response time(only useable in setting browser response time)
- turnaround_count ; (SUM)Final user response cases (only useable in setting browser response time)

13.9.5.4. PerHour

PerHour item outputs the performance data per agent through bar chart.

PerHour item has features as follows.


Properties

ItemID	S0010006
Chart Title	
Width	578
Height	208
Agent	${agent}
ServiceName	active_service
Date	${date}

- ChartTitle : This is a title of chart.
- Agent : This is an agent of data to be searched.
- ServiceName : This is a service name of data to be searched.

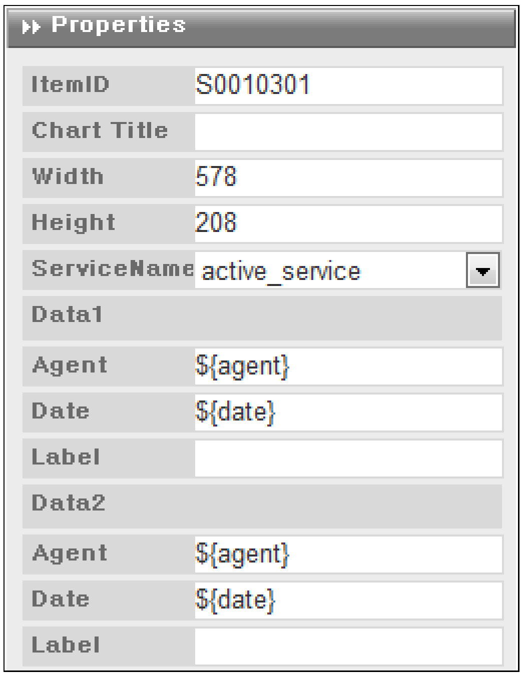
- Date : The date of data to be searched..

Notice: The selectable ServiceName item is same as AgentPF. However, if you change it into data per hour, “hit” and “down_time” use the SUM value per hour and visit_hour uses the maximum value of the time.

13.9.5.5. AgentPF2

AgentPF2 outputs a chart that compares both performance data per agent.

AgentPF2 item has features as follows



Properties

ItemID	S0010301
Chart Title	
Width	578
Height	208
ServiceName	active_service
Data1	
Agent	${agent}
Date	${date}
Label	
Data2	
Agent	${agent}
Date	${date}
Label	

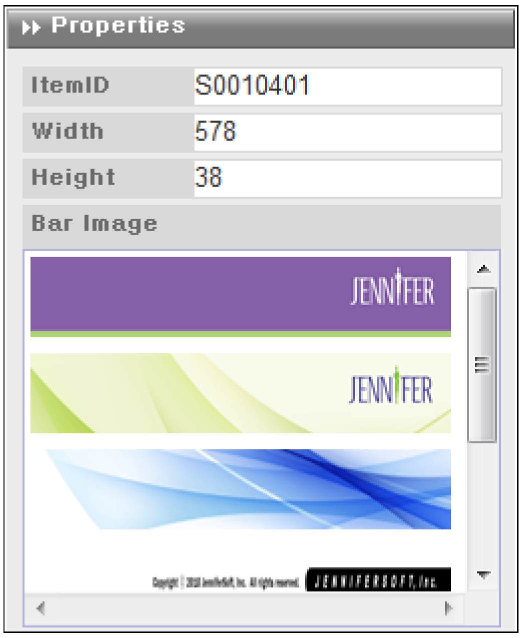
- ChartTitle : This is title of chart.
- ServiceName : Select the data to be searched.
- Agent : This is an agent of data to be searched.
- Label : This is a label to be displayed on the chart..

Notice: Selectable ServiceName is same as AgentPF. Input the data for “Data1” and “Data2”.

13.9.5.6. BarImage

BarImage item is an image item mostly used in title, header and footer of a report.

BarImage item has features as follows..

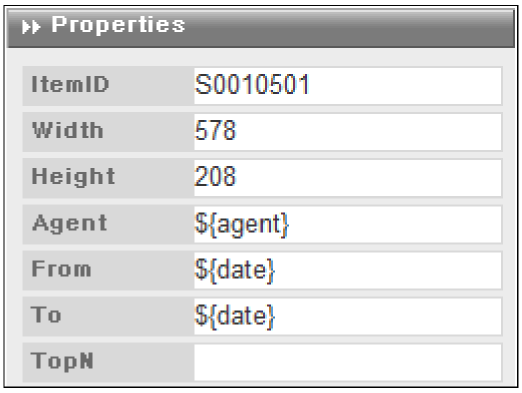


ItemID	S0010401
Width	578
Height	38
Bar Image	

- Bar Image : If you select an image, registered item is appied on the page.

13.9.5.7. ApplTopN

ApplTopN item outputs N of lastly delayed services. ApplTopN item has features as follows.



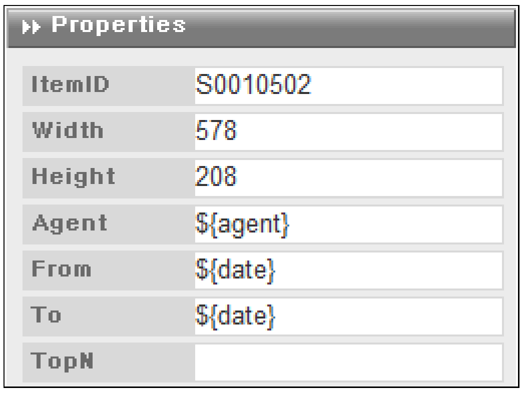
ItemID	S0010501
Width	578
Height	208
Agent	${agent}
From	${date}
To	${date}
TopN	

- Agent : Input the agent of the data to be searched.(TOT is total)
- From : Input the start date of data to be searched.
- To : Input the end date of data to be searched.
- TopN : Input the number that is to be searched from the top.

13.9.5.8. SqlTopN

SqlTopN item outputs N of lastly delayed services among executed SQL.

SqlTopN item has features as follows.

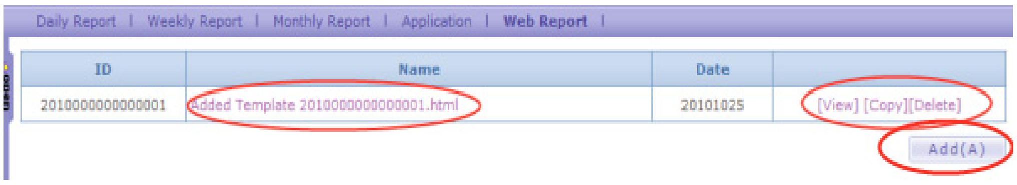


ItemID	S0010502
Width	578
Height	208
Agent	${agent}
From	${date}
To	${date}
TopN	

- Agent : Input the agent of the data to be searched.(TOT is total)
- From : Input the start date of data to be searched.
- To : Input the end date of data to be searched.
- TopN : Input the number that is to be searched from the top.

13.9.6.Management of Report Template

The report can be copied and modified by template unit.


Daily Report | Weekly Report | Monthly Report | Application | Web Report |
Add(A)

ID	Name	Date	
2010000000000001	Added Template 2010000000000001.html	20101025	[View] [Copy][Delete]

- Report name : The screen shows an edit page of report.
- [View] : The screen shows a parameter screen for report printing. If ypu input a parameter, you can print a report.
- [Copy] : Copy a report.
- [Delete] : Delete a report.
- [Add] : Create a new report.

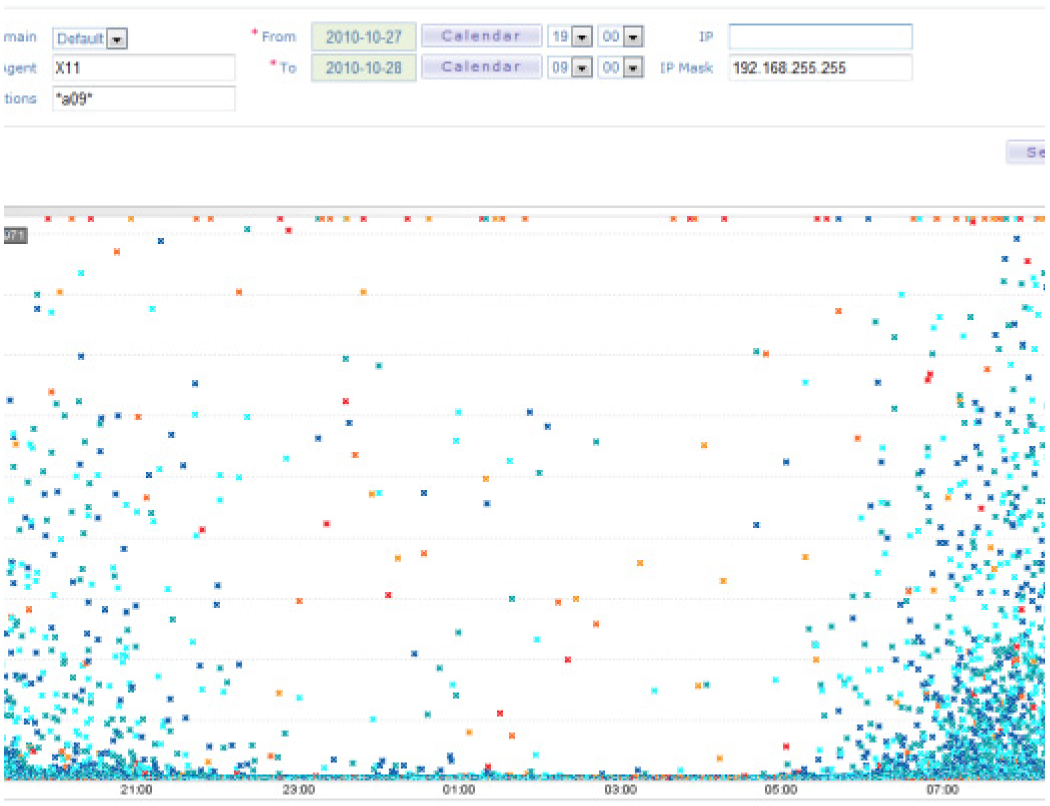
13.10. Advanced Statistics

In this chapter, newly added statistics features will be discribed since JENNIFER4.5.

13.10.1. Periodic X-view

You can search the X-View data for the certain period of time. Use this graph when you want to analyze the application service data over along-period of time.

Figure 13-43:Periodic X-View



Notice: But, if you search the heavy amount of data, the memory error may occur. Set the period appropriately.

13.10.1.1. Setting of searching condition

- From/To: Set the time that you want to search.

- Agent : Set the agent.
- Applications : Set the application service name to be filtered. Refer to the configurations as follows.

```
*AAA* : Service name including AAA
   BBB* : Service name including BBB
   *CCC : Service name ended with CCC
  AA*DD : Service name started with AA and ended with DD
```

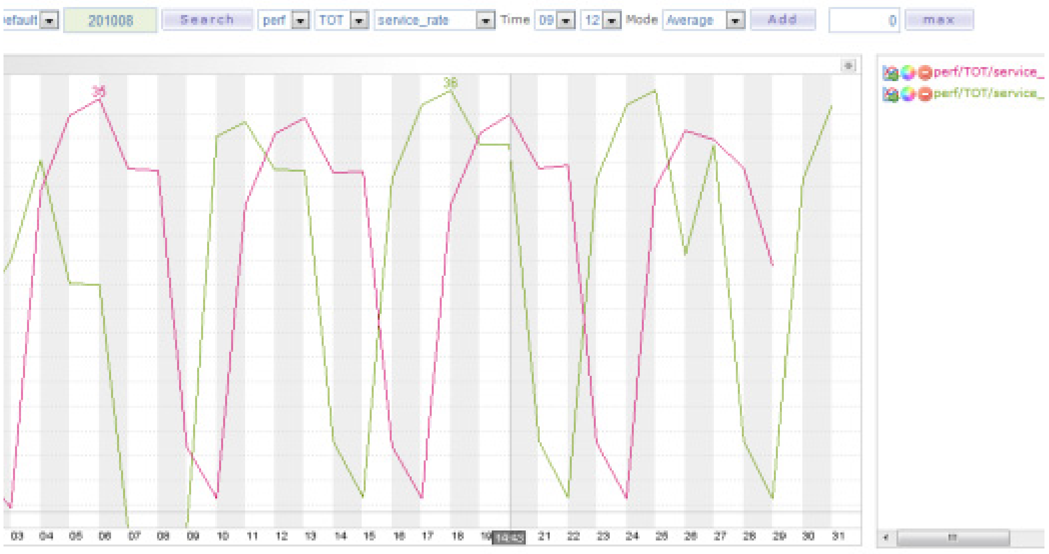
- IP : Client IP to be filtered.
- IP Mask : Refer to the configurations when you set the client IP that is to be filtered.

```
192.168.0.255 : All IP started with 192.168.0
 192.168.255.255 : 1All IP started with 92.168
 192.255.255.255 : All IP started with 192
```

13.10.2.Monthly PTA

JENNIFER analyze the system performance per month. It shows the pattern of data about a certain time zone during a month. The date from 1 to 31 are indicated on the X-axis.

Figure 13-44:Monthly PTA



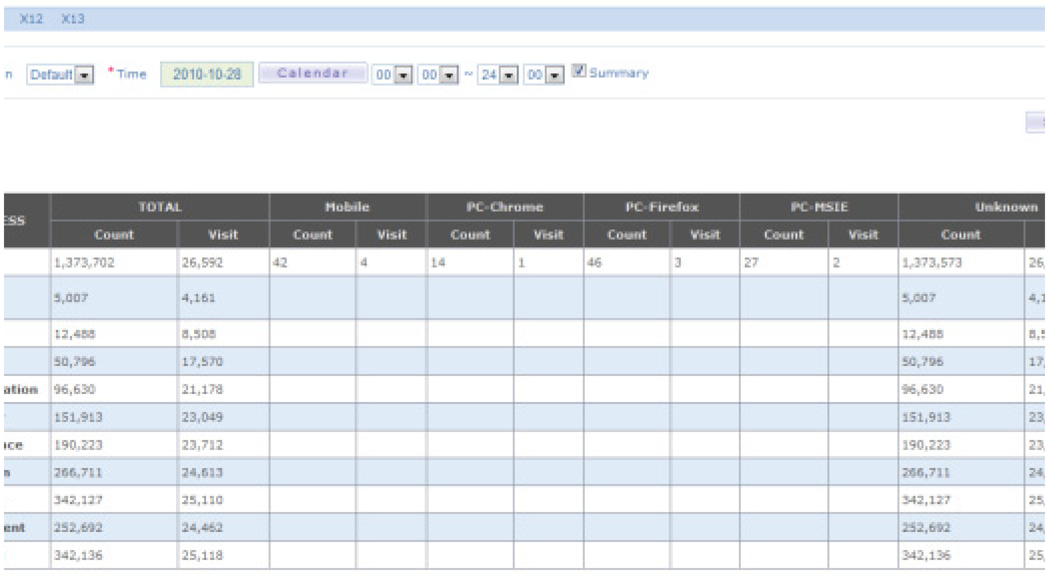
If you click the [Search] button after input the month, you can see the check box for the available value on the right. Select the value like perf/TOT/service_rate and select the time period.

For instance, if you want to see the average value of TPS between 9:00 and 12:00 during a month, select from 9:00 to 12:00 and Average mode. Then, click [Add] and you can see the added graph on the screen.

13.10.3.Browsers

JENNIFER outputs the statistics about the browser information about which browser such as PC or mobile was used when a user accesses to the JENNIFER dashboard.

Figure 13-45:Browsers



ESS	TOTAL		Mobile		PC-Chrome		PC-Firefox		PC-MSIE		Unknown	
	Count	Visit	Count	Visit	Count	Visit	Count	Visit	Count	Visit	Count	
	1,373,702	26,592	42	4	14	1	46	3	27	2	1,373,573	26,
	5,007	4,161									5,007	4,1
	12,488	8,508									12,488	8,5
	50,796	17,570									50,796	17,
ation	96,630	21,178									96,630	21,
	151,913	23,049									151,913	23,
ice	190,223	23,712									190,223	23,
n	266,711	24,613									266,711	24,
	342,127	25,110									342,127	25,
ent	252,692	24,462									252,692	24,
	342,136	25,118									342,136	25,

When you select the date and time, JENNIFER displays the number of hits per business and visitors per browser based on its executed services.

Notice: If you clear the [Summary] Check, JENNIFER outputs the statistics not parsing Http User-Agent.

18

Java Application Advanced Monitoring

This chapter describes methods for expanding the JENNIFER agent by programming and interconnecting the performance data collected by the JENNIFER server with other applications. In addition, it describes the method used to monitor the solution for implementing the SOA (Service Oriented Architecture).

14.1. Expanding the JENNIFER Agent

JENNIFER can collect various performance data without modifying the source code of the Java application by using the JENNIFER agent. You can expand the monitoring function by using the API and the adaptors provided by the JENNIFER agent.

Such an expansion requires simple programming. To compile the source code, you must first register the jennifer.jar file of the JENNIFER_HOME/agent directory in the class path.

14.1.1.ActiveTraceUtil

Using the ActiveTraceUtil class, you can modify the transaction data on your own. In addition to using the ExtraAgentUtil class, you will need to modify the application source code in order to expand the JENNIFER agent. Therefore, if the JENNIFER agent is removed, you must modify all of the source code that uses the ActiveTraceUtil class. The API for the ActiveTraceUtil class is as follows:

```
package com.javaservice.jennifer.agent;

public class ActiveTraceUtil {

    public ActiveTraceUtil();
    public ActiveTraceUtil(ActiveObject activeObject);

    public String getGUID();
    public void setGUID(String guid);

    public long getPUID();
    public void setPUID(long puid);

    public long getTUID();

    public void setAppException(String message);
    public void setAppException(int exceptionCode, String message);

    public void setApplicationName(String name);
    public void addActiveServiceName(String name);

    public void setExternalTransactionName(String name);

    public static void addProfile(String message);

}
```

The ActiveTraceUtil object should be used exclusively for the Java thread that processes the transaction. In other words, you must not use it as a global variable or as a member field of the javax.servlet.Servlet object. For this reason, you are recommended to create this object as a local variable in the method.

14.1.1.1. Modifying the Application Name

By default, the application name of a transaction is set using the option of the JENNIFER agent. For more details, refer to [Application Service Naming]. However, you can set the application name using the setApplicationName method of the ActiveTraceUtil class.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.setApplicationName("APP_NAME_001");
```

When the above code is executed by the transaction, the application name becomes APP_NAME_001.

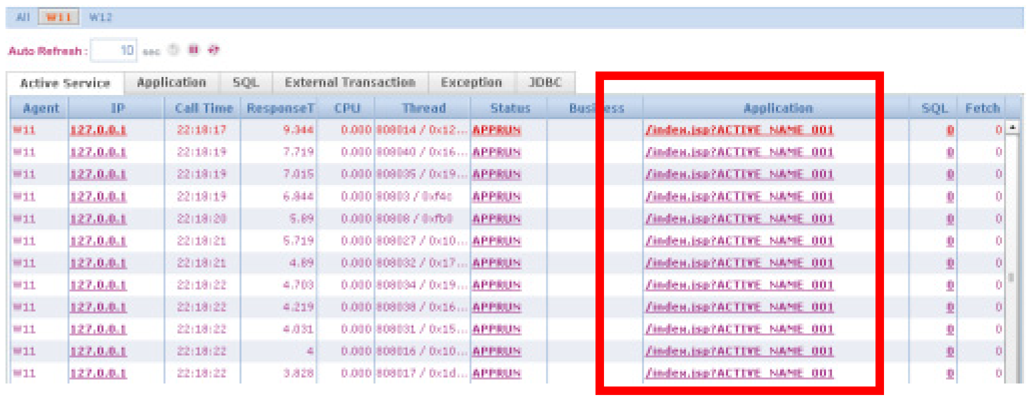
If you do not want to change the application name itself, but want to add arbitrary content to the application name that appears in the active service tab of the **[Real-time Monitoring | Application]** menu, use the addActiveServiceName method of the ActiveTraceUtil class.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.addActiveServiceName("ACTIVE_NAME_001");
```

If the above code is executed by the transaction that has the application name, /index.jsp, then you will see the following message in the active service tab:

```
/index.jsp?ACTIVE_NAME_001
```

Figure 14-1:Active Service Tab



Agent	IP	Call Time	ResponseT	CPU	Thread	Status	Business	Application	SQL	Fetch
W11	127.0.0.1	22:18:17	9.344	0.000	808014 / 0x12...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:19	7.719	0.000	808040 / 0x16...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:19	7.015	0.000	808035 / 0x19...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:19	6.844	0.000	80803 / 0xf4c	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:20	5.89	0.000	80808 / 0xfb0	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:21	5.719	0.000	808027 / 0x10...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:21	4.89	0.000	808032 / 0x17...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:22	4.703	0.000	808034 / 0x19...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:22	4.219	0.000	808038 / 0x16...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:22	4.031	0.000	808031 / 0x15...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:22	4	0.000	808016 / 0x10...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0
W11	127.0.0.1	22:18:22	3.828	0.000	808017 / 0x1d...	APPRUN		/index.jsp?ACTIVE_NAME_001	0	0

You can call the addActiveServiceName method of the ActiveTraceUtil class more than once.

```
activeTraceUtil.addActiveServiceName("ACTIVE_NAME_001");
...
activeTraceUtil.addActiveServiceName("ACTIVE_NAME_002");
```

If the above code is executed by the transaction that has the application name, /index.jsp, you will see the following message in the active service tab:

```
/index.jsp?ACTIVE_NAME_001+ACTIVE_NAME_002
```

14.1.1.2. Modifying the External Transaction Name

By default, an external transaction name is set using the option of the JENNIFER agent. For more details, please refer to [External Transaction Naming]. However, you can set an external transaction name using the setExternalTransactionName method of the ActiveTraceUtil class.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.setExternalTransactionName("TX_NAME_001");
```

When the above code is executed by the transaction, the external transaction name becomes TX_NAME_001.

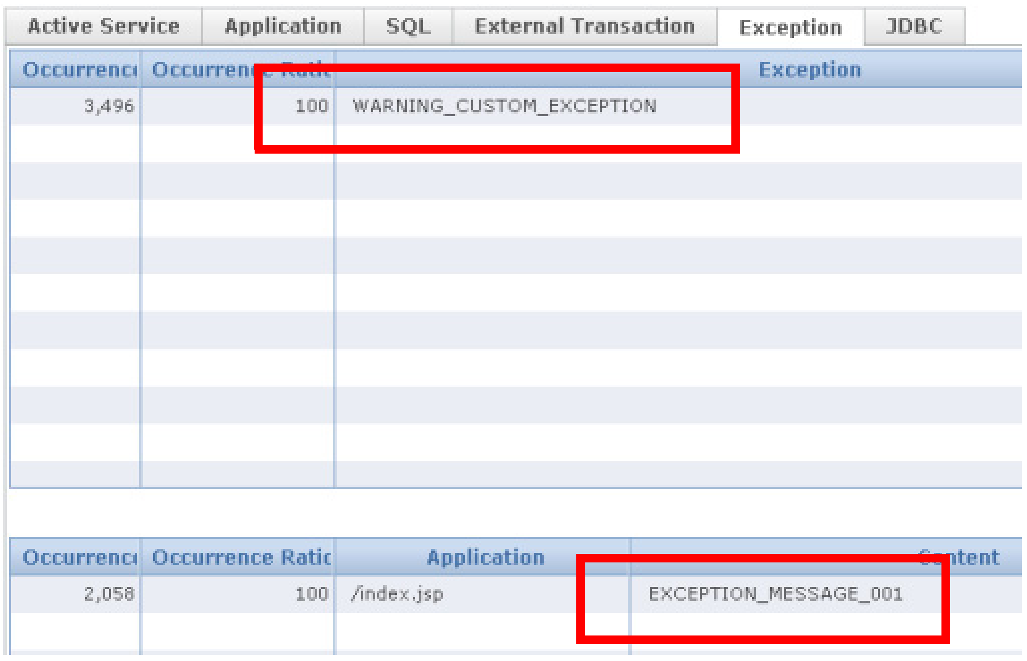
14.1.1.3. Generating an Arbitrary Exception

Using the setAppException method of the ActiveTraceUtil class, you can generate an arbitrary exception while executing the transaction.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.setAppException("EXCEPTION_MESSAGE_001");
```

If the above code is executed by the transaction, the transaction concludes that the WARNING_CUSTOM_EXCEPTION type of exception has occurred. The parameter of the setAppException method is an exception message.

Figure 14-2:Exception Tab



Occurrence	Occurrence Ratio	Exception
3,496	100	WARNING_CUSTOM_EXCEPTION

Occurrence	Occurrence Ratio	Application	Content
2,058	100	/index.jsp	EXCEPTION_MESSAGE_001

For more details on exceptions, refer to [Understanding Alerts and Exceptions].

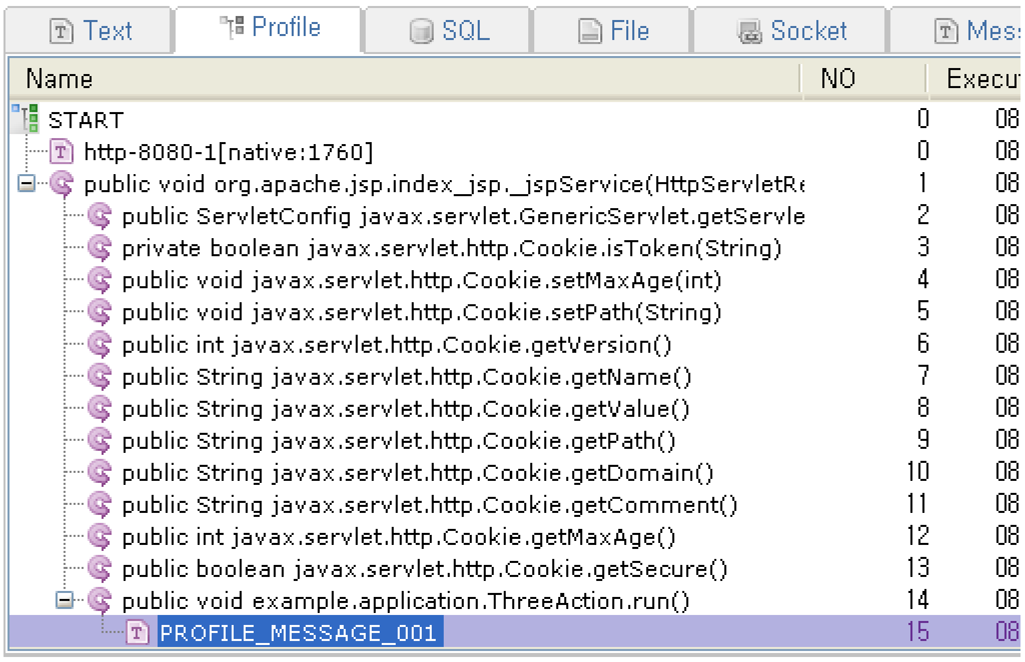
14.1.1.4. Adding a Message to the Profile

Using the addProfile method of the ActiveTraceUtil class, you can add an arbitrary message to the X-View profile data. Unlike other methods, this method is a static method of the ActiveTraceUtil class.

```
ActiveTraceUtil.addProfile("PROFILE_MESSAGE_001");
```

When the above code is executed while the transaction is in progress, the PROFILE_MESSAGE_001 message will appear in the X-View profile data.

Figure 14-3:X-View Profile



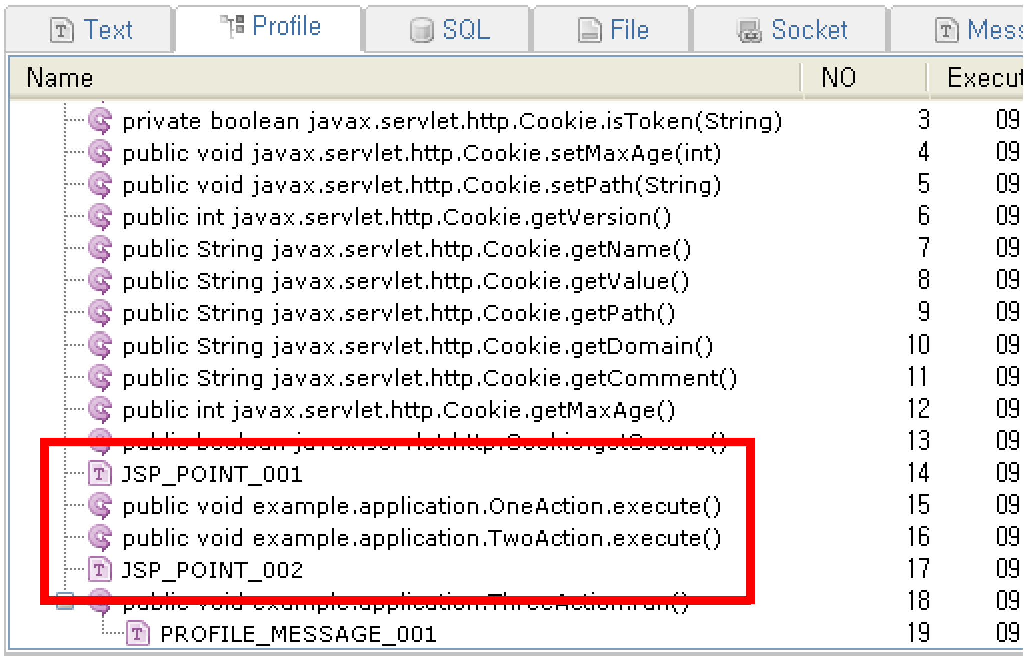
Name	NO	Execu
START	0	08
http-8080-1[native:1760]	0	08
public void org.apache.jsp.index_jsp._jspService(HttpServletRe	1	08
public ServletConfig javax.servlet.GenericServlet.getServle	2	08
private boolean javax.servlet.http.Cookie.isToken(String)	3	08
public void javax.servlet.http.Cookie.setMaxAge(int)	4	08
public void javax.servlet.http.Cookie.setPath(String)	5	08
public int javax.servlet.http.Cookie.getVersion()	6	08
public String javax.servlet.http.Cookie.getName()	7	08
public String javax.servlet.http.Cookie.getValue()	8	08
public String javax.servlet.http.Cookie.getPath()	9	08
public String javax.servlet.http.Cookie.getDomain()	10	08
public String javax.servlet.http.Cookie.getComment()	11	08
public int javax.servlet.http.Cookie.getMaxAge()	12	08
public boolean javax.servlet.http.Cookie.getSecure()	13	08
public void example.application.ThreeAction.run()	14	08
PROFILE_MESSAGE_001	15	08

Using this method, you can easily check the response time of each JSP region.

```
<%
    ...
    ActiveTraceUtil.addProfile("JSP_POINT_001");
    new OneAction().execute();
    new TwoAction().execute();
    ActiveTraceUtil.addProfile("JSP_POINT_002");
    ...
%>
```

If you execute the above JSP, you can analyze each JSP region in the X-View profile data.

Figure 14-4:JSP Region Profile



Name	NO	Execut
private boolean javax.servlet.http.Cookie.isToken(String)	3	09
public void javax.servlet.http.Cookie.setMaxAge(int)	4	09
public void javax.servlet.http.Cookie.setPath(String)	5	09
public int javax.servlet.http.Cookie.getVersion()	6	09
public String javax.servlet.http.Cookie.getName()	7	09
public String javax.servlet.http.Cookie.getValue()	8	09
public String javax.servlet.http.Cookie.getPath()	9	09
public String javax.servlet.http.Cookie.getDomain()	10	09
public String javax.servlet.http.Cookie.getComment()	11	09
public int javax.servlet.http.Cookie.getMaxAge()	12	09
public boolean javax.servlet.http.Cookie.getSecure()	13	09
JSP_POINT_001	14	09
public void example.application.OneAction.execute()	15	09
public void example.application.TwoAction.execute()	16	09
JSP_POINT_002	17	09
public void example.application.ThreeAction.run()	18	09
PROFILE_MESSAGE_001	19	09

For more details on the X-View profile, refer to [X-view and Profiling].

14.1.1.5. Transaction ID Control

Using the getTUID method of the ActiveTraceUtil class, you can obtain the transaction UUID. The transaction UUID is automatically set by the JENNIFER agent.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
long uuid = activeTraceUtil.getTUID();
```

As shown above, you can establish the relationship between the transaction data collected by JENNIFER and the business events or data.

Using the getGUID and setGUID methods of the ActiveTraceUtil class, you can set the GUID for global transaction interconnection tracking.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.setGUID("UNIQUE_GUID_001");
...
String guid = activeTraceUtil.getGUID();
```

The UUID is a long type while the GUID is a java.lang.String type. For more details on global transaction interconnection tracking, refer to [Global Transaction Interconnection Tracking].

14.1.2.ExtraAgent Adaptor

Sometimes, it is necessary to monitor data that is not collected by the JENNIFER agent. Such data can be collected using the ExtraAgent adaptor.

To use the ExtraAgent adaptor, you must first prepare the class that implements the com.javaservice.jennifer.agent.ExtraAgent interface.

```
package com.javaservice.jennifer.agent;

import com.javaservice.jennifer.protocol.RmPacket;

public interface ExtraAgent {

    public RmPacket process(String agent);

}
```

The process method of the ExtraAgent adaptor collects arbitrary data. When the RmPacket object is made from the collected data, the JENNIFER server sends the REMON data to the JENNIFER server or the REMON module. So, the JENNIFER server handles the data sent from the ExtraAgent adaptor in the same way that it handles the REMON data.

The following sample code is used to obtain the number of Java threads in each state. Since the JMX(Java Management Extension) is used, Java 1.5 or higher is required.

```
package example;


import java.lang.management.ManagementFactory;
import java.lang.management.ThreadMXBean;


import com.javaservice.jennifer.agent.ExtraAgent;
import com.javaservice.jennifer.protocol.RmPacket;
import com.javaservice.jennifer.type.INT;


public class ThreadCountExtraAgent implements ExtraAgent {


    public RmPacket process(String agent) {
        ThreadMXBean threads = ManagementFactory.getThreadMXBean();
        RmPacket result = new RmPacket("THREAD_COUNT", "EA1");
        result.addField("TOTAL", new INT(threads.getThreadCount()));
        result.addField("DAEMON", new
INT(threads.getDaemonThreadCount()));
        return result;
    }


}
```

THREAD_COUNT is the script ID of the REMON data, and EA1 is the agent ID of the REMON data. For more details on the script/agent ID, refer to [REMON Script]. For more details on how to use the RmPacket class, refer to [Using the RmPacket Class].

After compiling the source code, pack it into an arbitrary JAR file.

The following configuration is necessary for using the ExtraAgent adaptor. First, set the extra_enable option of the JENNIFER agent to true.

```
extra_enable = true
```

Using the extra_agent_class option of the JENNIFER agent, you can set the class that implements the ExtraAgent interface. If two or more ExtraAgent adaptors are used, semicolons [;] are used as separators.

```
extra_agent_class = example.ThreadCountExtraAgent
```

In addition, you must set the JAR file containing the class that implements the ExtraAgent interface in the extra_agent_classpath option of the JENNIFER agent. If two ore more JAR files are used, semicolons[;] are used as separators.

```
extra_agent_classpath = /jennifer/extension.jar
```

Next, set the execution period of the ExtraAgent adaptor in the extra_data_interval option of the JENNIFER agent. It is expressed in milliseconds.

```
extra_data_interval = 5000
```

If the data collected by the ExtraAgent adaptor is sent to the JENNIFER server, no further configuration is required. However, if you want to send it to the REMON module, set the REMON IP address and a port number in the extra_data_send_target option of the JENNIFER agent.

```
extra_data_send_target=127.0.0.1:7701
```

Warning: Different protocols are used depending on whether you are sending the data to the JENNIFER server or the REMON module. If you are sending it to the JENNIFER server, you must not use this option.

If you are sending the data to the REMON module, you can use a filter to process the data in various ways. For more details, refer to [Data Control].

These options can be modified without stopping the Java application. If you modified the class that implements the ExtraAgent adaptor, follow the steps below.

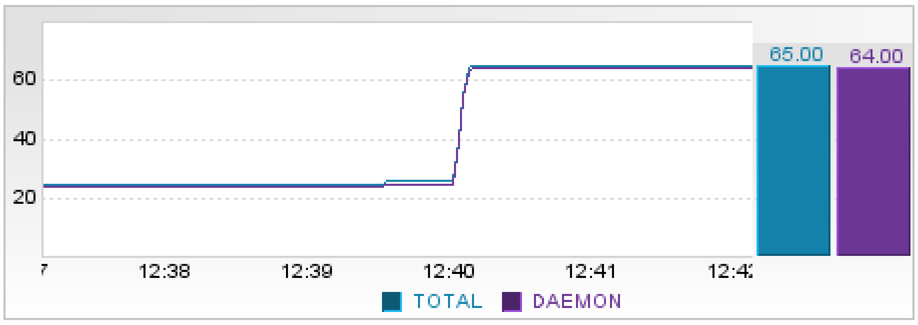
- Set the extra_enable option of the JENNIFER agent to false.
- After packing the modified class into a JAR file, replace the old JAR file.
- Set the extra_enable option of the JENNIFER agent to true.

However, You have to set the extra_agent_hotswap option of the JENNIFER agent to ture. Default is true.

```
extra_agent_hotswap = true
```

The JENNIFER server handles the data sent from the ExtraAgent agent in the same way that it handles the REMON data. Please refer to [User Defined Charts] for more details on the method for monitoring the data collected by the ExtraAgent adaptor. The following line chart represents the number of Java threads.

Figure 14-5:The Number of Java Threads



14.1.3.CustomTrace Adaptor

Using the CustomTrace adaptor, you can perform an arbitrary task before and after the transaction calls the method.

Before using the CustomTrace adaptor, you must first prepare the class that implements the com.javaservice.lwst.CustomTraceAdapter interface. The CustomTraceAdapter interface is as follows:

```
package com.javaservice.lwst;

public interface CustomTraceAdapter {

    public CustomStat start(String className, String methodName,
Object traceParam);

    public void end(CustomStat stat, Throwable e);

}
```

If the transaction calls a certain method of the class, then the start method of the CustomTrace adaptor is called before the method is executed. The first parameter of the start method is the class name, and the second parameter is the method name. The third parameter is the first method parameter of the object that is not an int or long type.

After the method is executed, the end method of the CustomTrace adaptor is called. If the start method returns a null value, the end method is not called. The first parameter of the end method is the CustomStat object returned by the start method, while the second parameter is the exception object that occurs in the method. If no exception has occured, then it returns a null value.

The following com.javaservice.lwst.CustomStat class is returned by the start method. If the start method returns a null value, the end method is not called. Therefore, it is desirable to return a null value when calling the method if the class does not require any processing.

```
package com.javaservice.lwst;

public class CustomStat {

    public String className;

    public String methodName;

    public long startTime;

    public Object traceValue;

}
```

The following example illustrates how the CustomTrace adaptor handles exceptions. In general, if the exception is processed before the transaction has been terminated, then the transaction is deemed successful.

```
public void execute() {
    try {
        new OrderManager().order("0-322-59443-1", 3);
    catch (IllegalStateException ignore) {
    }
}
```

If the inventory is insufficient, then the order method of the OrderManager class generates the java.lang.IllegalStateException type of exception. Although the above code is executed by the transaction, the transaction is deemed successful because the catch text handled the exception. However, you can use the CustomTrace adaptor to process the transaction as a failure, without modifying the existing source code.

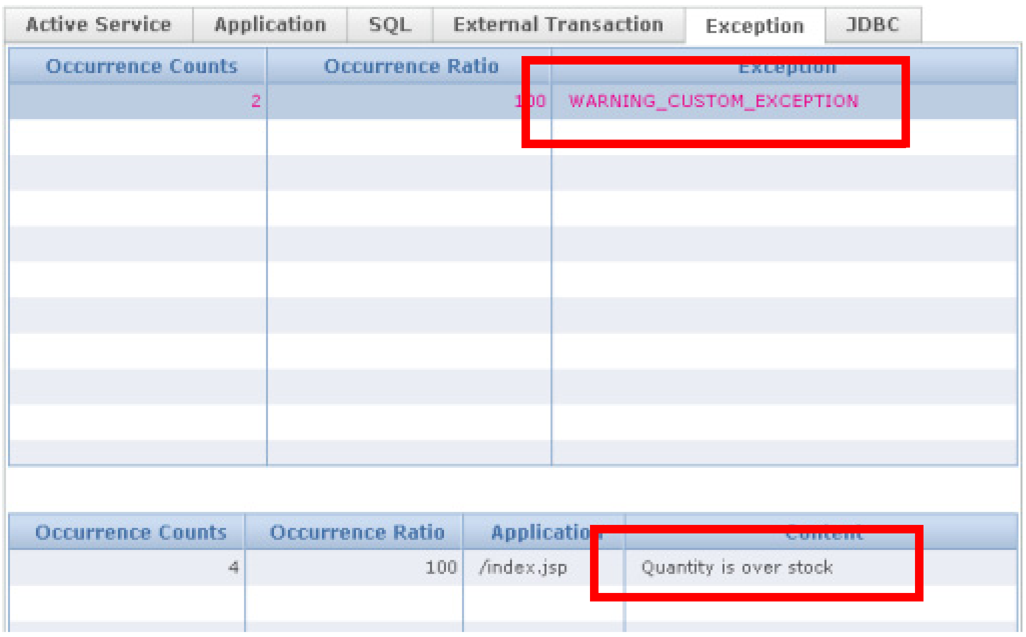
The ExceptionTraceAdapter class is used to implement the CustomTraceAdapter interface. If the end method generates the IllegalStateException type of exception, the set-

AppException of the ActiveTraceUtil class concludes that an exception has occurred in the transaction.

```
package example;

import com.javaservice.jennifer.agent.ActiveTraceUtil;
import com.javaservice.lwst.CustomStat;
import com.javaservice.lwst.CustomTraceAdapter;

public class ExceptionTraceAdapter implements CustomTraceAdapter {

    public CustomStat start(String className, String methodName,
Object traceValue) {
        CustomStat result = new CustomStat();
        result.className = className;
        result.methodName = methodName;
        result.traceValue = traceValue;
        result.startTime = System.currentTimeMillis();
        return result;
    }

    public void end(CustomStat stat, Throwable e) {
        if (e != null && e instanceof IllegalStateException) {
            new ActiveTraceUtil().setAppException(e.getMessage());
        }
    }

}
```

The following figure illustrates how to check exceptions in the exception tab of the **[Real-time Monitoring | Application]** menu.

Figure 14-6:Exception Handling Using the CustomTrace Adaptor



Occurrence Counts	Occurrence Ratio	Exception
2	100	WARNING_CUSTOM_EXCEPTION

Occurrence Counts	Occurrence Ratio	Application	Content
4	100	/index.jsp	Quantity is over stock

The following configuration is necessary for using the CustomTrace adaptor. First, set the class that implements the CustomTraceAdapter interface using the custom_trace_adapter_class_name option of the JENNIFER agent. You can use only one CustomTrace adaptor.

```
custom_trace_adapter_class_name = example.ExceptionTraceAdapter
```

In addition, set the JAR file containing the class that implements the CustomTraceAdapter interface in the custom_trace_adapter_class_path option of the JENNIFER agent.

```
custom_trace_adapter_class_path = /jennifer/extension.jar
```

When you modify the CustomTraceAdapter class, adjust the class as follows:

- After repackaging the JAR file with the modified class, replace the existing JAR file.
- Change the JENNIFER agent configurations in the **[Properties| Properties]** menu. Just click the **[Update]** button rather than changing the context.

But, you should the custom_trace_hotswap option of the JENNIFER agent to true. Default is true.

```
custom_trace_hotswap = true
```

However, the CustomTrace adaptor is not always used to call the methods. Using the custom_trace option of the JENNIFER agent, you can select which method the CustomTrace adaptor is applied to.

Warning: If you use the CustomTrace adaptor for the class that implements the CustomTraceAdapter interface, an infinite loop may be formed. For this reason, you must not use the CustomTrace adaptor for the class that implements the CustomTraceAdapter interface.

For example, if you want to use the CustomTrace adaptor for the business.OrderManager class shown earlier, set the custom_trace_class option of the JENNIFER agent.

```
custom_trace_class = business.OrderManager
```

If you want to use the CustomTrace adaptor for the business.ProductManager class as well, you must set it using the custom_trace_class option. Use a semicolon as a separator. If you want to enter multiple entities in any option related to the use of the CustomTrace adaptor, use a semicolons as a separators.

```
custom_trace_class = business.OrderManager;business.ProductManager
```

The CustomTrace is applicable to specific methods. In the above configuration, the CustomTrace adaptor is used for all methods of the business.OrderManager class.

To use the CustomTrace adaptor for specific methods only, set the custom_trace_target_method option of the JENNIFER agent.

```
custom_trace_target_method = order
```

If some classes to which the CustomTrace is applied share the same methods, and you want to use the CustomTrace adaptor for specific methods only, set it as follows:

```
custom_trace_target_method = business.OrderManager.order
```

On the other hand, if you want to avoid using the CustomTrace adaptor for specific methods, set the custom_trace_ignore_method option of the JENNIFER agent.

```
custom_trace_ignore_method = business.OrderManager.checkStock
```

Using the accessor of the method, you can apply the CustomTrace adaptor. For example, if you want to use the CustomTrace adaptor for all methods that do not have a public accessor or an accessor of any kind, set it as follows.

```
custom_trace_access_method = public;none
```

The following parameters can be set using the custom_trace_access_method option.

- public - public accessor
- protected - protected accessor
- private - private accessor
- none - No accessor

To use the CustomTrace adaptor for all the classes inheriting from a specific class, use the custom_trace_super option of the JENNIFER agent. For instance, if you want to use the CustomTrace adaptor for all the classes that inherit from the business.ejb.BaseSessionBean class, you must set it as follows:

```
custom_trace_super = business.ejb.BaseSessionBean
```

However, in this case, the CustomTrace adaptor is only applied to the classes that inherit from the business.ejb.BaseSessionBean class directly. For example, if class A inherits from the business.ejb.BaseSessionBean class and class B inherits from class A, the CustomTrace adaptor is applied to class A, but not to class B.

To use the CustomTrace adaptor for all classes that implement a specific interface, set the custom_trace_interface option of the JENNIFER agent. For example, to use the CuatomTrace adaptor for all classes that implement the business.IManager interface, set it as follows:

```
custom_trace_interface = business.IManager
```

However, in this case, the CustomTrace adaptor is only applied to the classes that implement the business.IManager interface directly. For example, if class A implements the business.IManager interface and class B inherits from class A, then the CustomTrace adaptor is applied to class A, but not to class B.

Using the class name, you can specify whether or not to apply the CustomTrace adaptor. In this case, use the custom_trace_prefix, custom_trace_postfix and custom_trace_ignore_prefix options of the JENNIFER agent. For example, to use the CustomTrace adaptor for all classes whose names begin with “business”, set it as follows:

```
custom_trace_prefix = business
```

Or, to use the CustomTrace adaptor for all the classes whose names end with “Manager”, set it as follows:

```
custom_trace_postfix = Manager
```

If you want to avoid using the CustomTrace adaptor for the classes that begin with certain names, use the custom_trace_ignore_prefix option of the JENNIFER agent. This option overrides all other options.

```
custom_trace_ignore_prefix =
```

Using the custom_trace option of the JENNIFER agent, you can select various types of configurations. Using the custom_trace_target_method option, apply the CustomTrace adaptor to the necessary methods only.

Let's assume that we have pkg.ClassA and pkg.ClassB and that the CustomTrace adaptor is applied to the run method of pkg.ClassA and the process method of pkg.ClassB. If another run method exists in pkg.ClassB and the CustomTrace adaptor is not applied to it, set it as follows:

```
custom_trace_class = pkg.ClassA;pkg.ClassB
custom_trace_target_method = run;process
custom_trace_ignore_method = pkg.ClassB.run
```

Or, you can set it as follows:

```
custom_trace_class = pkg.ClassA;pkg.ClassB
custom_trace_target_method = pkg.ClassA.run;pkg.ClassB.process
```

14.1.4.Sending Alerts from the JENNIFER Agent to the JENNIFER Server

Using the ExtraAgentUtil class, you can generate an arbitrary alert. Like the ActiveTraceUtil class, this requires modification of the application source code for expansion of the JENNIFER agent. Therefore, if you remove the JENNIFER agent, you must modify all of the source codes that use the ExtraAgentUtil class. The full name of the ExtraAgentUtil class is as follows:

```
com.javaservice.jennifer.agent.ExtraAgentUtil
```

The ExtraAgentUtil class provides the following methods for alerts.

- sendAlertFATAL(String) - USER_DEFINED_FATAL alert
- sendAlertERROR(String) - USER_DEFINED_ERROR alert
- sendAlertWARNING(String) - USER_DEFINED_WARNING alert
- sendAlertMESSAGE(String) - USER_DEFINED_MESSAGE alert
- sendAlert(String) - Same as sendAlertWARNING

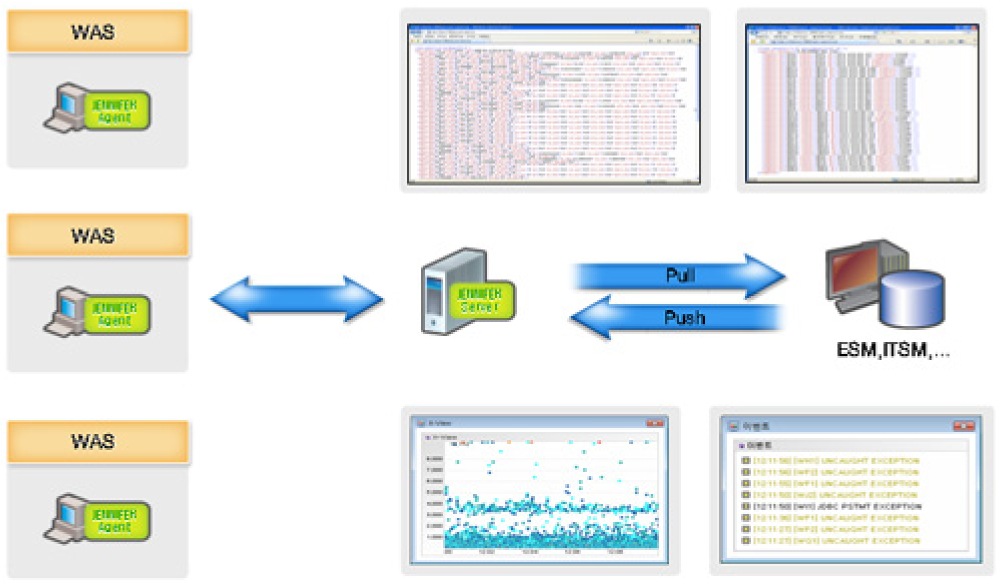
- sendAlert(int, String) - It generates an arbitrary alert designated by the first parameter.The first parameter uses a constant field with the same name as the alert type existing in the com.javaservice.jennifer.util.Def class.

For more details on alerts, refer to [Alerts and Exception Types].

14.2. Performance Data Interconnection

This section describes two methods for sharing the performance data collected by the JENNIFER server with other applications. These methods can be divided into a pull method and a push method.

Figure 14-1:Concept of Interconnecting the Performance Data



14.2.1.Pull Type of Performance Data Interconnection

A pull type of performance data interconnection implies that the external application sends a HTTP request to the JENNIFER server, and the JENNIFER server returns an XML document in response.

14.2.1.1. General Performance Data Inquiry

Using the following request, you can view the general performance data at the time of the request in an XML file.

```
http://jennifer_server_ip:7900/get_perf_agent.jsp
```

The basic structure of the XML file is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<perf version="4.0.1.1" jennifer="SYS1"
      time="2008-11-10 14:16:45.716">

</perf>
```

The top tag is a perf tag, and its attributes are as follows:

- version - JENNIFER server version
- JENNIFER - JENNIFER server domain ID
- time - the creation time of the XML file

And the perf tag is composed of a lower-ranked instance tag that represents the JENNIFER agent.

```
<instance agent="TOT"
   ac0="1" ac1="1" ac2="0" ac3="5"
    act_serv="7"
   act_user="7"
   tps="25.633333"
   res_time="0.16205852"
   con_user="271.74252"
   error_rate="0.2"
   reject_rate="0.0"
   hit_hour="61679" visit_hour="4208"
   proc_cpu="0.0" proc_mem="0.0"
   hit_day="600767" visit_day="28807"
   alert_fatal="7" alert_error="3844" alert_warn="9874" />

<instance agent="W11" ...
```

The following table describes the attributes of the instance tag.

Table 14-21: instance Tag Attributes

Field name	Description
ac0 ~ ac3	Represents the number of active services in each time segment
act_serv	Total number of active services. It is a sum of the numbers of active services in all time segments.
act_user	Active users
tps	Service rate
res_time	Average response time
con_user	Concurrent users
error_rate	Rate of exceptions
reject_rate	Rate of service rejection by the PLC
hit_hour	Hits for the current time
visit_hour	Visitors for the current time
proc_cpu	System CPU utilization
proc_mem	System memory utilization
hit_day	Today’s hits

Table 14-21: instance Tag Attributes

Field name	Description
visit_day	Today's visitors
alert_fatal	Today's fatal alerts
alert_error	Today's error alerts
alert_warn	Today's warning alerts

14.2.1.2. Business Group Performance Data Inquiry

Using the following request, you can view the business group performance data at the time of the request in the form of an XML file.

```
http://jennifer_server_ip:7900/get_perf_biz.jsp
```

The basic structure of the XML file is as follows.

```
<?xml version="1.0" encoding="UTF-8" ?>
<perf version="4.0.1.1" jennifer="SYS1"
      time="2008-11-10 14:16:45.716">

</perf>
```

The top tag is a perf tag, and its attributes are as follows:

- version - JENNIFER server version
- JENNIFER - JENNIFER server domain ID
- time - Creation time of XML file

The perf tag is composed of a lower-ranked biz tag that represents a business group.

```
 <biz id="biz01" sla="10000" name="Biz01"
       service_time="24.59"
       sql_time="14.056"
       etx_time="3.511"
       client_time="0.0"
       service_count="3"
       service_per_min="0.6"
       turnaround_per_min="0.0"
       accu_error="0" accu_hit="3"
       rule="*(cmd=a01)" />
  <biz id="biz02" sla="10000" name="Biz02"
       service_time="15.748"
       sql_time="5.466"
       etx_time="6.637"
       client_time="0.0"
       service_count="1"
       service_per_min="0.2"
       turnaround_per_min="0.0"
       accu_error="0" accu_hit="1" rule="*(cmd=a02)" />
```

The following table describes the attributes of the biz tag.

Table 14-22: Biz Tag Attributes

Field name	Description
id	Unique ID
name	Business group name
sla	SLA threshhold value
service_time	Average response time of the business group
sql_time	Average SQL execution time of the business group
etx_time	Average external transaction time of the business group
client_time	Average client response time of the business group
service_count	Accumulated transaction requests for the business group(5M)
service_per_min	Accumulated transaction requests for the business group per min
turnaround_per_min	Accumulated end user's transaction tracking for the business group per min
accu_error	Accumulated exceptions of the business group

Table 14-22: Biz Tag Attributes

Field name	Description
accu_hit	Accumulated hits for the business group

Using a business group, you can collect various data, including the service rate and the average response time of a specific application group. For example, you can collect performance data in relation to the important tasks, such as ordering and payments.

By selecting the **[Properties | Business Group]** menu, you can set the business group. The method for adding a business group is as follows:

- Click on the **[Add]** button at the bottom of the business group list. A business group input form will appear on the right.
- After filling out the business group input form, click the **[Save]** button at the bottom.

Notice: To reflect the change in the business group, click the **[Apply]** button at the bottom of the business group list.

The following table provides information on the business group.

Table 14-23: Business Group Information

Item	Description
ID	A unique ID. Enter an arbitrary value for it. You can organize a tree-shaped ID system by using a backslash[/] as a separator.
Business division	A suffix for the REMON data. Business groups with the same division have the same REMON data.
Name	A business group name. It is the field name for the REMON data.
Application	The name of an application in the business group. The ENTER key is used as a separator. You can also use a wild card[*].
External transaction	The name of an external transaction in the business group. The ENTER key is used as a separator. You can use a wild card[*]. External transactions are not included in the REMON data. In the **[Statistics |Report |Daily Report]** menu, you can check the status of external transaction processing for the business group.
Domain	Select a JENNIFER server.
Agent	If you want to extract the business group data from specific JENNIFER agents only, enter the JENNIFER agents IDs here. Separate multiple entities with a comma[,].
Status	Whether the business group is used or not

14.2.2.Push Type of Performance Data Interconnection

A push type of performance data interconnection implies that the performance data collected by the JENNIFER server is transmitted to another application using the XVLog or SMS adaptor.

14.2.2.1. X-View Transaction Data Processing using an XVLog

You can perform an arbitrary task on the X-view transaction data collected by the XVLog adaptor in real time.

Notice: To compile the XVLog adaptor, you must first register the JENNIFER_HOME/server/lib/jenniferserver.jar file in the class path.

Before using the XVLog adaptor, you must first prepare the class that implements the com.javaservice.server.XVLog interface. The XVLog interface is as follows:

```
package com.javaservice.jennifer.server;

import com.javaservice.jennifer.util.CircularObjectQueue;

public interface XVLog {

    public void execute(CircularObjectQueue queue);

}
```

The execute method of the class that implements the XVLog interface is called repeatedly. The parameter of the execute method is the CircularObjectQueue object. This object contains the TxPerfPacket object representing the X-View transaction data.

The TxPerfPacket object in the CircularObjectQueue object has the structure shown below. Using the member field that is a public accessor, you can check the necessary data.

```
package com.javaservice.jennifer.protocol;

public class TxPerfPacket {

    public long store_time;
    public long end_time;
    public int elapsed;
    public int cpu_time;
    public int sql_time;
    public int fetch_time;
    public int etx_time;
    public int service_name_hash;
    public long tx_uuid;
    public int client_ip;
    public long wmonid;
    public byte err_type;
    public String agent;

    //using for global transaction trace
    public String guid;
}
```

The member fields that are generally used range from the store_time field to the agent field. The rest of the member fields are only used on special occasions.

- store_time - A long type, it represents the time that the JENNIFER server saves the X-View transaction data. It uses the clock of the Hardware on which the JENNIFER server is installed.
- end_time - A long data type, it represents the end time of the transaction. It uses the clock of the hardware on which the JENNIFER agent is installed.
- elapsed - An int data type, it represents the response time of the transaction. It is expressed in milliseconds.
- cpu_time - An int data type, it is the time of CPU use by the transaction. It is expressed in milliseconds.
- sql_time - An int data type, it is the response time of the SQL executed by the transaction. It is expressed in milliseconds.

- fetch_time - An int data type, it is the fetch time of the SQL executed by the transaction. It is expressed in milliseconds.
- etx_time - An int data type, it represents the response time of the external transaction executed by the transaction. It is expressed in milliseconds.
- service_name_hash - An int data type, it represents the hash value of the application name.
- tx_uuid - A long data type, it represents the transaction UUID.
- client_ip - An int data type, it represents the client IP address.
- wmonid - A long data type, it represents the wmonid cookie value used to calculate the number of visitors and concurrent users.
- err_type - A byte data type, it represents the type of exception that occurred in the transaction.
- agent - A java.lang.String data type, it represents the JENNIFER agent ID.
- guid - A java.lang.String data type, it represents the transaction GUID for global transaction interconnection tracking.
- puid - A java.lang.String data type, itrepresents the previous transaction GUID for global transaction interconnection tracking.

The following sample illustrates how the XVLog adaptor can record the X-View transaction data in the console.

```
package example;

import java.text.DateFormat;
import java.text.SimpleDateFormat;

import com.javaservice.jennifer.protocol.TxPerfPacket;
import com.javaservice.jennifer.server.XVLog;
import com.javaservice.jennifer.server.db.HashedString_DB;
import com.javaservice.jennifer.util.CircularObjectQueue;
import com.javaservice.jennifer.util.IpUtil;

public class ConsoleXVLog implements XVLog {

    private static DateFormat df =
                        new SimpleDateFormat("yyyyMMddHHmmssS");

    public void execute(CircularObjectQueue queue) {
        while (queue.size() > 0) {
            TxPerfPacket packet = (TxPerfPacket) queue.dequeue();
            String time =
                   df.format(new java.util.Date(packet.end_time));
            String appName =
                   HashedString_DB.getAPPL(packet.service_name_hash);
            System.out.println("Time=" + time + ", UUID=" +
packet.tx_uuid + ", Agent=" + packet.agent + ", App="
+ appName + ", Client IP=" + IpUtil.intToAddress(packet.client_ip) + ",
Elapsed Time="
+ packet.elapsed + ", SQL Time=" + packet.sql_time + ", TX Time=" +
packet.etx_time);
        }
    }

}
```

The following configuration is necessary for using the XVLog adaptor. First, set the enable_xvlog option of the JENNIFER server to true.

```
enable_xvlog = true
```

Next, using the xvlog_class option of the JENNIFER server, set the class that implements the XVLog interface.

```
xvlog_class = example.ConsoleXVLog
```

After compiling the XVLog adaptor, pack it into a JAR file and copy it into the JENNIFER_HOME/server/common/lib directory. Now, restart the JENNIFER server.

14.2.2.2. Alert Data Interconnection

Using the SMS adaptor, you can send alert data to other applications.

Notice: Before compiling the SMS adaptor, you must register the JENNIFER_HOME/server/common/lib/jenniferserver.jar file in the class path.

Before using the SMS adaptor, you must prepare the class that implements the com.javaservice.jennifer.server.Sendsms interface. The Sendsms interface is givens as follows:

```
package com.javaservice.jennifer.server;

public interface Sendsms {

    public void sendsms(ErrorObject[] alerts);

}
```

When an alert is issued, the sendsms method of the class that implements the Sendsms interface is called. The parameter of the sendsms method is the ErrorObject object, which represents the type of alert data.

The com.javaservice.jennifer.server.ErrorObject class is defined as follows:

```
package com.javaservice.jennifer.server;

import com.javaservice.jennifer.server.db.HashedString_DB;

public class ErrorObject {

    public long time = 0;

    public String agentname = null;

    public int type = 0;

    public int group_type = 0;

    public int reqkey_hash = 0;

    public String getReqKey() {
        return reqkey_hash == 0 ? null :
               HashedString_DB.getAPPL(reqkey_hash);
    }

    public float value = -1f;

    public String ipaddr = null;

    public String msg = null;

    public long tx_uuid;

}
```

Using the member field that is a public accessor, you can check the content of the alert. The following is a description of each field.

- time - A long data type, it represents the time that the alert is generated.
- agentname - A java.lang.String data type, it represents the JENNIFER agent ID related to the alert. It returns a null value if the alert is irrelevant to the JENNIFER agent.

- group_type - An int data type, it classifies the alert group type. 1, 2, 3 and 4 represents critical, error, warning and message, respectively. 0 represents an unknown type of alert.
- type - An int data type, it represents various types of alerts including ERROR_JVM_DOWN or WARNING_JVM_CPU_HIGH. It compares the constant fields having the same name as the alert type of the com.javaservice.jennifer.util.Def class.
- reqkey_hash - An int data type, it represents the hash name of the application name. The application name is obtained using the getReqKey method.
- float value - A float data type, it represents the value in excess of the threshold when an alert is issued because the threshold is exceeded. A good example is the WARNING_APP_BAD_RESPONSE alert. As such, it is not meaningful to alert types that do not involve the threshold.
- ipaddr - A java.lang.String data type, it represents the IP address of the hardware in which the JENNIFER agent or the WMOND module related to alerts is installed. If the alert type is ERROR_SYSTEM_DOWN, then it represents the IP address of the hardware in which the WMOND module is installed. If the alert is irrelevant to the JENNIFER agent, then it returns a null value.
- msg - A java.lang.String data type, it represents the message related to the alert.
- tx_uuid - A long data type, it represents the transaction UUID.

The following example illustrates how the SMS adaptor is used to only store the ERROR_SERVICE_QUEUING type of alert in the database.

```
package example;

import com.javaservice.jennifer.server.ErrorObject;
import com.javaservice.jennifer.server.Sendsms;
import com.javaservice.jennifer.util.Def;

public class SaveAlertSendsms implements Sendsms {

    public void sendsms(ErrorObject[] alerts) {
        for (int i = 0; i < alerts.length; i++) {
            ErrorObject alert= alerts[i];
            if (alert.type == Def.ERROR_SERVICE_QUEUING) {
                // Saving it in the database
            }
        }
    }

}
```

Now, we will describe the classes and the methods that are useful for interconnecting the alert data. The ErrStr class is used to convert an int type of alert into a java.lang.String type of alert.

```
com.javaservice.jennifer.util.ErrStr
```

- getShortMsg(int id) - This is a static method. If this method is used to call an alert, the summarized alert name is returned.
- getFullMsg(int id) - This is a static method. If this method is used to call an alert, the full alert name is returned.
- getTypeChar(int id) - This is a static method. If this method is used to call an alert, the alert type is returned. C, E, W and I represent critical, error, warning and message alerts, respectively.

The AlertMsgCtr class is used to obtain the message appearing in the alert chart.

```
com.javaservice.jennifer.util.AlertMsgCtr
```

- getMessage(ErrorObject e) - This is a static method. When this method is called, the detailed message for the alert is returned. This message is same as the one shown in the alert chart.

The TrapSender class is used to transmit the SNMP TRAP message.

```
com.javaservice.jennifer.server.snmp.TrapSender
```

- addTrapMsg(String s) - When this method is called, a new SNMP TRAP message is added.

The following is a description of how to use the SMS adaptor. You can set the class that implements the Sendsms interface using the sms_adapter_class_name option of the JENNIFER agent.

```
sms_adapter_class_name = example.SaveAlertSendsms
```

Notice: When registering two or more SMS adaptors, use semicolons [;] as separators.

After compiling the SMS adaptor, pack it into a JAR file and copy it into the JENNIFER_HOME/server/common/lib directory. Then, restart the JENNIFER server.

14.3. Specialized Monitoring

This section describes the method for monitoring the solution for SOA implementation.

Notice: The following content may not be suitable depending on the method in which the solution is used. Therefore, you are recommended to use the following content only as a reference for monitoring the solution.

14.3.1.Oracle WLI Monitoring

To monitor the WLI(WebLogic Integration), the following configuration is required.

First, in many cases, the application service start point is not the javax.servlet.http.HttpServlet class. If this is the case, you must set the tx_server_class and tx_server_target_method option of the JENNIFER agent as follows.

```
tx_server_class =
com.bea.wli.knex.runtime.core.dispatcher.Dispatcher;weblogic.jms.clien
t.JMSSession
tx_server_target_method = remoteDispatch;onMessage
```

To simplify the application name, set the tx_server_ntype option of the JENNIFER agent to SIMPLE.

```
tx_server_ntype = SIMPLE
```

The WLI uses Apace BeeHive for JDBC connections. To perform JDBC monitoring, set the jdbc_connection_justget option of the JENNIFER agent as follows.

```
jdbc_connection_justget =
org.apache.beehive.controls.system.jdbc.JdbcControlImpl.getConnectionF
romDataSource(String,Class)
```

This configuration defines the application service start point and the method of JDBC monitoring. However, for more detailed monitoring, you can use the CustomTrace adaptor to extract the main information from the application.

Notice: You must modify the application source code.

As JENNIFER provides the class to implement the CustomTrace adaptor for the WLI, you do not need to implement the CustomTrace adaptor on your own. Now, set the custom_trace_adapter_class_name and custom_trace_adapter_class_path options of the JENNIFER agent as follows:

```
custom_trace_adapter_class_name = jennifer.custom.AdapterWLI
custom_trace_adapter_class_path = /jennifer/agent/lwst40.custom.jar
```

Notice: The `lwst40.custom.jar` file is located in the `JENNIFER_HOME/agent` directory.

You can set the class to which the CustomTrace adaptor is applied using the custom_trace_class option of the JENNIFER agent.

```
custom_trace_class = jennifer.JenniferGate
```

The JenniferGate class does not provide a JAR file. Prepare the following code for the application that uses the WLI, and register it in the class path.

```
package jennifer;

public class JenniferGate {
    public static void TRAN_NAME(String s){}
    public static void TRAN_STEP(String s){}
    public static void TRAN_ID(String s){}
    public static void WARN(String s){}
    public static void PROFILE(String s){}
}
```

If you set the main information by calling the method of the JenniferGate class in the application, CustomTrace will begin to extract the information. The following describes the method of the JenniferGate class.

- TRAN_NAME - Call this method if you want to change the application service name.
- TRAN_STEP - Call this method if you want to set the status of active service appearing in the active service tab of the **[Real-time Monitoring | Application]** menu.
- TRAN_ID - Call this method if you want to set the GUID for global transaction interconnection tracking.
- WARN - Call this method if you want to process the current transaction as if an exception has occurred within it. The WARNING_CUSTOM_EXCEPTION exception will be generated.
- PROFILE - Call this method if you want to add an arbitrary profile list to the X-View profile data.

14.3.2.Oracle Service Bus(OSB) Monitoring

JENNIFER does not emphasize the monitoring of SOA solutions such as the ESB(Enterprise Service Bus). However, since the ESB-based applications also operate in the WAS, you can use JENNIFER to monitor certain parts of operations. This section describes the method for monitoring the Oracle OSB(Oracle Service Bus or Aqualogic Service Bus), which is one of the popular ESB solutions, using JENNIFER.

The OSB processes single requests with multiple Java threads or transactions. Therefore, it requires global transaction interconnection tracking. To enable this, set the trace_related_transaction option of the JENNIFER agent to true.

```
trace_related_transaction = true
```

If you want to use the transaction UUID granted by the JENNIFER agent as the GUID, set the enable_guid_from_tuid option of the JENNIFER agent to true. .

```
trace_related_transaction = true
```

If you want to use the transaction UUID granted by the JENNIFER agent as the GUID, set the enable_guid_from_tuid option of the JENNIFER agent to true.

```
enable_guid_from_tuid = true
```

The OSB processes a single request with multiple Java threads or transactions. The javax.servlet.http.HttpServlet class is set as the application service start point of the first transaction. However, you will need to set the application service start points for other interconnection transaction on your own. To do this, set the tx_server_class, tx_server_target_method and tx_server_ntype options of the JENNIFER agent as follows:

```
tx_server_class = com.bea.wli.sb.pipeline.MessageProcessor
tx_server_target_method = processResponse
tx_server_ntype = CLASS
```

The main objective of the OSB is to interconnect with external applications. Therefore, you must set the external transaction start points. If you want to interconnect with Oracle Tuxedo or Siebel, you must set the tx_client_class, tx_client_target_method and lwst_txclient_method_using_return options of the JENNIFER agent as follows:

```
tx_client_class =
com.bea.wli.sb.transports.tuxedo.TuxedoTransportProvider; \
                com.bea.alsb.transports.siebel.SiebelTransportProvider
tx_client_target_method = sendMessageAsync;processMessage
lwst_txclient_method_using_return =
weblogic.wtc.gwt.TDMImport.getRemoteName(); \

com.bea.alsb.transports.siebel.impl.SiebelEndpointConfigurationImpl.ge
tBusinessName()
```

If you want to interconnect with other external applications, you will need to add another start points to the above code.

A user can check the phases of the OSB work flow in an X-view profile, and can monitor the global transaction interconnections through a GUID view of an X-view chart.

14.4. JENNIFER Mobile

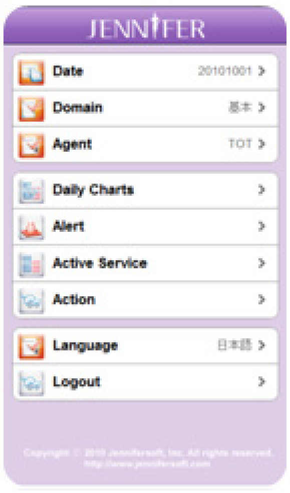
You can use a new JENNIFER client to use in your smart phone. We call this JENNIFER Mobile.

- Be Included in the JENNIFER server. User don't need to install it separately.
- Use only web port.
- Runs on every smart phone using standard browser.
- Use the same ID/Password with the JENNIFER.

14.4.1.Default Parameter

You have to select Date, Domain and Agent in the JENNIFER Mobile screen.

Figure 14-2:Default Parameter

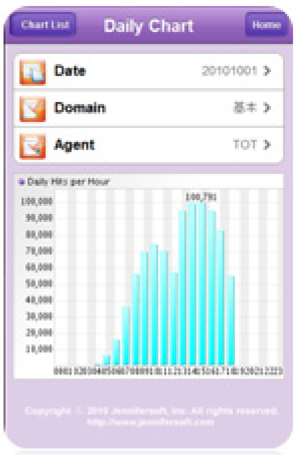


- Date : Select the date
- Domain : Select the domain
- Agent : Select the agent

14.4.2.Daily Charts

Select the chart that you want to search. The daily charts can be searched.

Figure 14-3:Daily Charts



- Hit Per Hour: Hits per hour
- Service Rate : Service rate per second
- Average Response Time: Average response time
- Visitors per Hour : Visitors per hour
- Concurrent Users : Concurrent users
- Active Service : Active services
- System CPU Utilization :System CPU utilization(But except TOT)
- Process CPU Utilization : Process CPU Utilization(But except TOT)
- Heap Memory Rate: Heap memory utilization (But except TOT)
- Process Memory Usage: Process memoryutilization (But except TOT)
- System Memory Usage : System memory utilization(But except TOT)

14.4.3.Alert

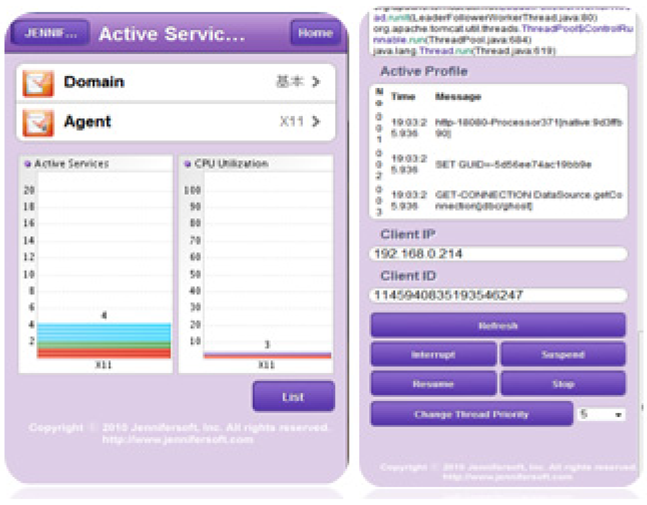
Search the occurred alert list. Input the below data and search. Then, the data is output in a time inverse.

- Type : Select the alert type.
- Time : The time occurred an alert
- Condition : Alert name or detailed string that you want to filter separately.
- Max Count : Maximum prints pages

14.4.4.Active Service

User can search the current active services and detailed profile data. However, if you select TOT, only Active Services are searched. If you select the general agent, you will see the system CPU utilization of the agent.

Figure 14-4:Active Service



ad.run(LeaderFollowerWorkerThread.java:80)
org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:684)
java.lang.Thread.run(Thread.java:619)

No	Time	Message
001	19:03:25.936	http-18080-Processor371[native:9d3fb90]
002	19:03:25.936	SET GUID=-5d56ee74ac19bb9e
003	19:03:25.936	GET-CONNECTION DataSource.getConnection()dbc/ghost]

Client IP: 192.168.0.214

Client ID: 1145940835193546247

14.4.5.Others

- Language : Select the language.
- Logout :Log out.

14.4.6.Action Registration and Use

In the JENNIFER Mobile, user can call the main shell to manage the server on the phone. When you installed the JENNIFER, the example includs server starts and ends and log view of process list.

Notice: e: It can be a back door to control the real system, so you must use the JENNIFER Mobile after guaranteeing the security issue. Install the JENNIFER agent module in the system and set the execution authority and path of shell files in the $JENNIFEr_AGENT/action directory.

If you run the main.sh, RmAgent runs that process the request of the JENNIFER Mobile.

```
JAVA_HOME=/usr/java
JENNIFER_SERVER=192.168.0.101

$JAVA_HOME/bin/java -cp ../rmagent/RmAgent.jar RmAgent.MobileAction \
             -a X11,X12,X13 \
             -t 127.0.0.1:6902 \
             -l 127.0.0.1:7715 \
             -cmd:start "./start.sh"  \
             -cmd:stop ./stop.sh \
             -cmd:log ./log.sh \
             -cmd:ps ./ps.sh \
             -cmd:dummy2 ./dummy2.sh
```

Notice: User have to modify the main.sh interior data directly.

- start.sh : Execute the Java application server
- stop.sh : Stop the Java application server
- log.sh : Search LOG data
- ps.sh : View process

Each Action must be registered in the mobile window and please refer to the following configuration.

```
mobileActions=start:Start;stop:Stop;log:Log View;ps:Process
```

Define the action through ”;”. Write the <action> name in the cmd:<action> registered in main.sh to be displyed in the mobile window.

19 REMON

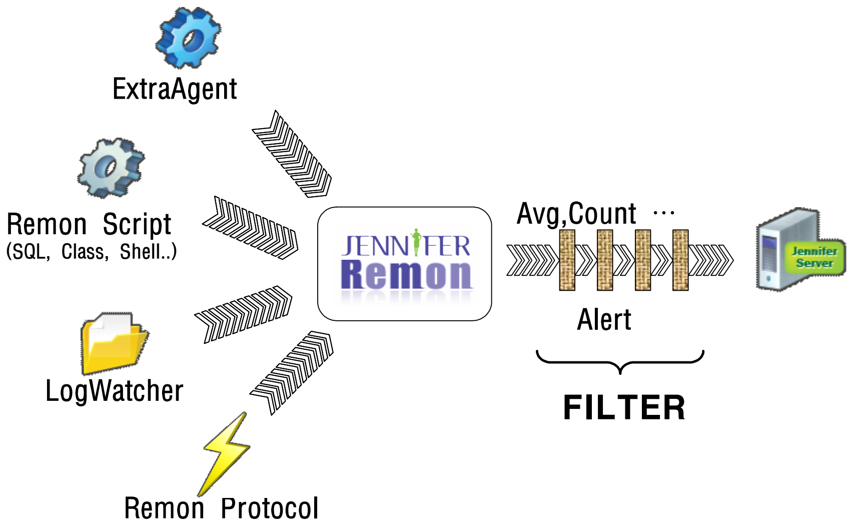
The REMON module is one of the JENNIFER independent agents that collects and processes non-standard data and sends it to the JENNIFER server. In general, a monitoring solution is comprised of various functions such as data collection, processing, transmission, storage and viewing. The REMON module is a generalized framework for data collection, processing and transmission.

Using the REMON module, you can execute an arbitrary script on the OS or an arbitrary SQL on the database, or can execute a Java application that implements a specific SQL. Using these methods, you can collect non-standard data and send it to the JENNIFER server.

The business data and the operating system data that the JENNIFER agent cannot collect by itself can be monitored in this manner.

15.1. REMON Architecture

Figure 15-1:REMON Configuration Chart



The REMON module collects the data by itself, or receives it from an external data source.

First, the module that collects the data is called the REMON script. The REMON does not simply refer to a unix shell script. It includes the SQL and Java class script as well as the unix shell script. The REMON script can be classified as a local script or a remote script, depending on the location of data collection. If data collection is performed at the REMON module itself, it is called a local script, while if it is performed at another operating system, it is called a remote script.

The types of local scripts are as follows:

- OS shell script
- Class script

The types of remote scripts are as follows:

- Telnet script
- SSH2 script
- SQL script

Sometimes the REMON module does not collect the data on its own, but rather receives the data from external sources. The following is a list of external data sources:

- REMONX
- ExtraAgent
- LogWatcher

The data collected by the REMON script and from the external data source is called the REMON data. The REMON data has the same shape, regardless of its source.

The REMON data can be uniquely distinguished by its agent ID and script ID.

- Agent ID - This refers to the source of the REMON data. It must consist of uppercase letters, numbers and [_] and its maximum length is 128 characters. However, if the REMON data is monitored from an X-ViewC chart, the maximum length is 3 characters.
- Script ID - This represents the characteristics of the REMON data. It defines the CPU utilization, the network utilization, and the inventory. It must be comprised of uppercase letters, numbers and [_] and its maximum length is 256 characters.

The REMON data is comprised of one or more fields. Each field must have the same data type. Data types include int, float, long, double and string.

Notice: There is no maximum number of fields. However, if the REMON data is stored at the JENNIFER server, then the maximum number of fields will depend on the maximum number of columns in the table.

Finally, the REMON module sends its data to the JENNIFER server through the UDP. If the REMON data requires processing, then a filter can be used to process the REMON data before transmitting it to the JENNIFER server. You can use multiple filters for a single set of REMOND data.

Notice: You can selectively send the REMON data to two or more JENNIFER servers.

Overall management of the REMON module is performed using a control console.

15.2. Installation and Configuration

The REMON module is located in the remon directory of the JENNIFER installation package. The structure of the REMON directory is as follows:

- conf - The REMON configuration files are located here.
- lib - The Java library files used by the REMON module are located here.
- log - A log file is located here.
- logwatcher - This is a log monitor called LogWatcher. It is based on the REMON module, and is located in the REMON directory.
- remonx - This contains the REMONX module that sends the execution result of an arbitrary script to the REMON module when the REMON module is not installed.

- script - The REMON script files are located here.

If you want to install the REMON module, copy this directory to the OS in which you want to install it.

Notice: If you want to install it on a unix or linux system, assign execution authority to the execution file after copying it.

Since the REMON module is implemented in Java, set Java as a JAVA_HOME environment variable, and set the REMON installation directory as a REMON_HOME environment variable as well.

Notice: To execute REMON, use Java 1.4.2 or higher.

The JAVA_HOME and REMON_HOME environment variables are set in the env.sh(env.bat) files of the REMON_HOME directory. If you are using linux or unix, modify it as follows:

```
REMON_HOME=/usr/jennifer/remon
JAVA_HOME=/usr/java
```

If you are using Windows, modify it as follows.

```
set REMON_HOME=C:/jennifer/remon
set JAVA_HOME=C:/Program Files/Java/jdk1.6.0_07
```

Next, you must set the network information used by the REMON module and the JENNIFER server to receive the REMON data in the REMON_HOME/conf/remon.conf file.

First, set the JENNIFER agent ID as follows. The following configuration is applied to all agent IDs of the REMON data that are not explicitly specified when collecting the data.

```
remon.agent = R01
```

You must set the UDP port that the REMON module uses to receive the external data using the local.udp.port option. The default port number is 7701.

```
local.udp.port = 7701
```

In addition, the REMON module receives a TCP request from the control console and the JENNIFER server. Set the port number using the local.tcp.port option. The default port number is 7701.

```
local.tcp.port = 7701
```

You can use the same port number for the UDP and TCP ports that receive the data or a request. By default, port 7701 is used for the purpose.

Now, set the IP address of the REMON module using the local.ip option. This IP address is used by the JENNIFER server when it initiates a reverse TCP call to the REMON server.

```
local.ip = 127.0.0.1
```

Finally, designate the JENNIFER server that receives the REMON data.

```
jennifer.sys1.ip = 127.0.0.1
jennifer.sys1.port = 6902
jennifer.sys1.agent = *
```

The sys1 parameter has an arbitrary value. In general, it refers to the domain ID of the JENNIFER server. If you need to send the REMON data to multiple JENNIFER servers, then you must add options and use different values than sys1.

```
jennifer.sys1.ip = 192.168.0.1
jennifer.sys1.port = 6902
jennifer.sys1.agent = *

jennifer.sys2.ip = 192.168.0.2
jennifer.sys2.port = 6902
jennifer.sys2.agent = *
```

Set the JENNIFER server IP address using the jennifer.sys1.ip option, and set the port number using the jennifer.sys1.port option. This port is set with the server_udp_listen_port option of the JENNIFER server. The default port number is 6902.

In general, all REMON data is sent to all of the JENNIFER servers. This is because the jennifer.sys1.agent option is set to *. If you want to transmit certain REMON data only, enter their agent IDs separated by [,].

```
jennifer.sys1.agent = R01,R03,R05
```

Notice: Instead of a script ID, you can use an agent ID to select the JENNIFER server that receives the REMON data.

15.3. Execution and Control

15.3.1.Execution

To execute the REMON module, run the remon.sh file in the REMON_HOME directory.

```
./remon.sh
```

At this point, By default, the SeedKey is ‘jennifer’. SeedKey is a key used to encrypt the password used in the REMON script and its configuration file. If you enter an incorrect SeedKey, the REMON module will not operate properly.

If you want to change SeedKey, you must enter the new value when executing the REMON module. After changing SeedKey, you must change all of the passwords in the REMON configuration files and scripts.

15.3.2.Stopping

To stop the REMON module, run the shutdown.sh file in the REMON_HOME directory.

```
./shutdown.sh
```

15.3.3.Control Console

The REMON module is managed using a control console.

Notice: The REMON module in JENNIFER 3.x provides a web-based user interface. However, in JENNIFER 4.0 only a console-based user interface is provided.

To execute the control console, run the console.sh file in the REMON_HOME directory.

```
~/jennifer/remon$ ./console.sh


Jennifer REMON: Release 4.0.1.1(2008-10-31)
Copyright (c) JenniferSoft 1998,2008. All rights reserved.


REMON>
```

To stop the control console, type in the exit command.

```
REMON> exit
```

From the control console, you can use various commands to perform tasks, such as REMON script control and filter control. For more details on these commands, execute the help command.

```
REMON> help
```

The following describes the main commands:

You can use the shutdown command on the control console to terminate the REMON module. When this command is invoked, the control console will also be terminated.

```
REMON> shutdown
```

To check the Java environment variables that execute the REMON module, type in the env command.

```
REMON> env
```

To check only the environment variables that begin with a certain text string, enter the following command.

```
REMON> env [Arbitrary text string]

ex) env java
```

To check the environment configuration of REMON, enter the conf command.

```
REMON> conf
```

To only check the REMON environment configuration that begins with a certain text string, enter the following command.

```
REMON> conf [Arbitrary text string]

ex) conf local
```

To check the Java heap memory utilization and the status of the data queue using the REMON module, enter the perf command.

```
REMON> perf
```

To check the status of errors that occur, while operating the REMON module, enter the err command.

```
REMON> err
```

To clear the information regarding errors that occur while operating the REMON module, enter the errc command.

```
REMON> errc
```

To execute the REMON script, you need to set a password for the remote server or database in the REMON script file. For security reasons, you must set the password based on SeedKey. In this case, use the enc command.

```
REMON> enc <text string to be encrypted>
```

Other commands will be described in subsequent sections.

15.4. Data Collection

The REMON module may collect the data on its own using a REMON script, or may receive it from various external sources, such as REMONX, ExtraAgent or LogWatcher.

15.4.1.REMON Script

Every REMON script can be classified according to its extension in the text file. The REMON script should be located in the REMON_HOME/script directory. Child directories and other directories cannot detect it.

Notice: Sample REMON scripts are found in the REMON_HOME/script/sample directory.

The following is a list of REMON script types:

- OS shell script

- Telnet script
- SSH2 script
- SQL script
- Class script

All REMON scripts have attributes used to control executions. The following is a description of commonly used attributes, regardless of the type of REMON script.

Table 15-1: REMON Scripts Common Attritbutes

Attribute	Description	Remarks
script	Sets the script ID that characterizes the data. Sets the ID for every script. A script ID and an agent ID are used altogether to refer to the REMON data.	Uppercase letters, numbers, _
agent	Sets the agent ID that defines the source of the data. If not specified, the default value in the remon agent option of the REMON configuration file will be used. An agent ID and a script ID are used together to refer to the REMON data. If you want to monitor it on an X-ViewC chart, an agent ID should be three characters in length.	Uppercase alphabets, numbers,_
interval	Sets the execution period for the REMON script. It is expressed in seconds. The default value is one second.	Positive integer
fieldname	The data collected by the REMON script is composed of one or more fields. This is used to assign a name to the field. You must use a comma [,] as a separator. If field names are not specified, the data will be assigned names such as f0, f1, and f2 in order.	Lowercase alphabets, numbers
fieldtype	Sets the type of the field that constitutes the REMON data. Every field must be of the same type. For this reason, unlike the fieldname attribute, only one value is assigned to it. The default value is a string.	[int | float | long | double | string]
request	This means that the REMON script is executed only when the JENNIFER server requests it. The default configuration is false.	[true | false]
autostart	Specifies whether or not to automatically execute the REMON script when the REMON module is executed. If set to false, then the REMON script is not automatically executed. If set to true, then the REMON script is automatically executed when the REMON module is executed. The default configuration is true.	[true | false]

The REMON script attribute is described in a single line that begins with #$.

```
#$ agent = R01
```

However, for a Windows bat file, REM must be added at the beginning.

```
REM #$ agent= R01
```

15.4.1.1. OS Shell Script (.sh / .bat)

You can use a shell script of the OS in which the REMON module is installed as a REMON script. This is called an OS shell script. An OS shell script is a local script, and its extension must be sh or bat. The following example illustrates connections of the network sockets for the Unix or Linux OS in which the REMON module is installed. The following content is saved as the net.sh file in the REMON_HOME/script directory. .

```
#!/bin/sh

##############################################
#  REMON Script Attribute
#$ script = NET
#$ agent = R01
#$ interval = 5
#$ fieldtype = int
#$ fieldname = est,tim,fin
##############################################

net=`netstat -an`
est=`echo $net | grep EST | wc -l`
tim=`echo $net | grep TIME | wc -l`
fin=`echo $net | grep FIN | wc -l`
echo $est,$tim,$fin
```

The est, tim and fin fields are used to collect the int type of data every five seconds, which consist of the number of established TCP connections, TIME_WAIT TCP connections and FIN_WAIT TCP connections. The REMON data uses NET as a script ID, and R01 as an agent ID.

Warning: As the REMON module executes the OS shell by itself, it must be given execution authority.

15.4.1.2. Telnet Script(.telnet)

Using telnet, you can remotely access other OS and execute a shell script to collect the data. This is called a telnet script. A telnet script is a remote script, and its extension must be telnet.

The following is a description of the attributes used in a telnet script.

Table 15-2: Attributes used in a Telnet Script

Attributes	Description	Remarks
telnet.ip	Connection server IP address	A host name or IP address
telnet.port	Connection server port number	Integer
telnet.user	Connection server user ID	Text string
telnet.password	Connection server user password	The enc command for the control console should be used to encrypt the password to be registered. After the SeedKey is changed, it must be reset.
prompt.user	Text string for a login prompt	
prompt.password	Text string for a password prompt	
prompt.char	Text string for a general prompt	
prompt.beforelogin	Should be registered if the user is required to enter certain input before a login prompt appears.	
prompt.preprompt	Should be registered if the user is required to enter certain input before the prompt.char is displayed on the screen after entering the password.	

A local script uses the interval attribute to adjust the interval of executions, but a remote script should be made in an infinite loop to periodically collect the data.

```
while true
do
  netstat -an | wc -l
  sleep 5
done
```

15.4.1.3. SSH2 Script(.ssh2)

Using SSH2, you can remotely access other OS and execute a shell script to collect the data. This is called an SSH2 script. An SSH2 script is a remote script, and its extension must be ssh2.

The following is a description of the attributes used in an SSH2 script:

Table 15-3: Attributes used in an SSH2 Script

Attributes	Descriptions	Remarks
ssh2.ip	Connection server IP	A host name or IP address
ssh2.port	Connection server port number	Integer
ssh2.user	Connection server user ID	Text string
ssh2.password	Connection server user password	The enc command for the control console should be used to encrypt the password to be registered. After the SeedKey is changed, it must be reset.

A local script uses the interval attribute to adjust the interval of executions, but a remote script must be made in an infinite loop to periodically collect the data.

```
while true
do
  netstat -an | wc -l
  sleep 5
done
```

15.4.1.4. SQL Script(.sql)

You can execute the SQL on the external database to collect the data. This is called an SQL script. An SQL script is a remote script, and its extension must be sql.

The following is a description of the attributes used in an SQL script:

Table 15-4: Attributes used in an SQL Script

Attritbutes	Descriptions	Remarks
jdbc.driver	JDBC driver class name	
jdbc.url	Database connection URL	
jdbc.user	Database user ID	
jdbc.password	Database user password	The enc command for the control console should be used to encrypt the password to be registered. After the SeedKey is changed, it must be reset.

The name of the column searched by the SQL becomes the data field name. For this reason, you do not need to set a fieldname attribute.

Warning: After copying the JDBC driver file into the REMON_HOME/lib/script directory, you must restart the REMON module.

An SQL script is a remote script, but it can set the execution period using the interval attribute.

The following is a sample SQL script.

```
########################################################
#  REMON Script Attribute
#$ script = PERF_CNT
#$ agent = R01
#$ interval = 2
#$ fieldtype = int

#$ jdbc.driver = org.apache.derby.jdbc.ClientDriver
#$ jdbc.url =  jdbc:derby://localhost:1527/jennifer
#$ jdbc.user = jennifer
#$ jdbc.password = rf7CtZ/Oy6YGEtFFY/fo2A==

SELECT COUNT(*) CNT FROM PERF_X_29
```

15.4.1.5. Class Script(.cls)

By executing the class that implements the remon.IClassScript interface, you can collect the data. This is called a class script. A class script is a local script and its extension must be cls.

The following table describes the attributes used in a class script:

Table 15-5: Attributes used in a Class Scrip

Attributes	Descriptions	Remarks
exec	The name of the class that implements the remon.IClassScript interface	

Since the class that implements the remon.IClassScript interfaces does most of the work involved in data collection, its structure is simpler than that of other REMON scripts. The following is a sample code used to collect the Java heap memory utilization data.

```
##############################################
#  REMON Script Attribute
#$ script = REMON_HEAP
#$ agent = R01
#$ interval = 1

#$ exec = example.HeapMemoryClassScript
```

The class set with the exec attribute must implement the remon.IClassScript interface. However, it is generally recommended to inherit from the remon.AbstractClassScript class that implements the remon.IClassScript interface.

Notice: To compile it, register the REMON_HOME/lib/sys/remon.jar file in the class path. Then pack the compiled class into a JAR file and copy it into the REMON_HOME/lib/script directory. After that, restart the REMON module. When the class has been modified, repeat the above steps and restart the REMON module.

The remon.AbstractClassScript class is as follows:

```
package remon;

import java.util.Properties;
import com.javaservice.jennifer.protocol.IRmPacket;

abstract public class AbstractClassScript implements IClassScript {

    final public Properties properties = new Properties();
    abstract public IRmPacket process(String[] param);

}
```

Therefore, after inheriting from the remon.AbstractClassScript, you must implement the process method.

```
package example;

import remon.AbstractClassScript;

import com.javaservice.jennifer.protocol.IRmPacket;
import com.javaservice.jennifer.protocol.RmPacket;
import com.javaservice.jennifer.type.INT;

public class HeapMemoryClassScript extends AbstractClassScript {

    public IRmPacket process(String[] param) {
        return null;
    }

}
```

The process method returns the data created and transmitted by the com.javaservice.jennifer.protocol.RmPacket object that implements the com.javaservice.jennifer.protocol.IRmPacket interface.

```
private static final int MB = 1024 * 1024;

public IRmPacket process(String[] param) {
    long now = System.currentTimeMillis();
    RmPacket packet = new RmPacket(now, "SCR1", "A01");

    Runtime runtime = Runtime.getRuntime();
    int total = (int) (runtime.totalMemory() / MB);
    int used = (int) (runtime.totalMemory() / MB - runtime.freeMemory()
/ MB);

    packet.addField("total", new INT(total));
    packet.addField("used", new INT(used));

    return packet;
}
```

First, the created RmPacket object contains the parameters indicating data collection time, script ID, and agent ID. However, if the script ID or agent ID has already been set in the class script, then it will be used as previously set. Therefor, in this sample, the REMON data's script ID is REMON_HEAP, not HEAP.

By using the addField method of the RmPacket class, you can add the data. Assign a field name to the first parameter of the addField method, and data to the second parameter. The type of the second parameter can be specified with the child class of the com.javaservice.jennifer.type.TYPE class.

Notice: Unlike other REMON scripts, you can use various types of classes from the remon.IClassScript implementation class.

For more details on how to use the RmPacket class, refer to [Using the RmPacket Class].

All of the attributes set in the class script are transferred to the properties member field of the remon.AbstractClassScript class. This means that you can easily register the necessary attributes in the class script.

```
##############################################
#  REMON Script Attribute
...
#$ max = 1000
...
```

```
public IRmPacket process(String param[]) {
    ...
    String max = properties.get("max");
    ....
}
```

15.4.2.Registering and Controlling a REMON Script

All of the REMON scripts(text file) should exist in the REMON_HOME/script directory. However, simply adding the REMON script to this directory does not mean that it will operate. You need to register it in the control console.

Notice: If you restart the REMON module, then all REMON scripts in the REMON_HOME/script directory are automatically registered.

To check the list of the REMON scripts being executed, enter the ls command in the control console.

```
REMON> ls
```

To check the list of the REMON scripts that exist in the REMON_HOME/script directory but are not yet registered in the REMON module, enter the lsn command in the control console.

```
REMON> lsn
perf.sql
net.sh
```

If you want to register certain REMON scripts, use the load command and enter the REMON script file name as a parameter.

```
REMON> load [REMON Script file name Prefix]

ex) load net.sh
```

If you want to register all REMON scripts, enter the load command without any parameters.

```
REMON> load
```

The REMON script added by the load command is currently stopped. If you want to start it, use the start command, and enter the script and agent ID as a parameter.

```
REMON> start [Script and agent ID Prefix]

start NET
    - Starts all REMON scripts whose script ID starts with NET.
start NET.R
    - Starts all REMON scripts whose script and agent ID starts with
NET and R.
start NET.R01
    - Starts all REMON scripts whose script and agent ID starts with
NET and R01.
```

If you want to start all of the REMON scripts, use the start command without any parameters.

```
REMON> start
```

To check the final data collected by the REMON script in the control console, use the data command and enter the script and agent ID.

```
REMON> data [Script ID, agent ID Prefix]


예) data NET
2008-07-29 19:50:48.811:NET:R01(est=4,tim=3,fin=1)
```

To check all data, use the data command without any parameteras.

```
REMON> data
```

To delete the REMON data, use the dataclear command. This does not affect data collection or transmission.

```
REMON> dataclear
```

To update the REMON script after changing it, use the reload command and enter the script and agent Id as a parameter.

```
REMON> reload [Script Id.Agent Id Prefix]
```

To restart all the REMON scripts, use the reload command without any parameters.

```
REMON> reload
```

To remove the REMON script, use the unload command and enter the script and agent ID as a parameter. You can only remove a REMON script that is stopped..

```
REMON> unload [Script Id.Agent Id Prefix]
```

If you want to remove all REMON scripts, use the unload command without any parameters.

```
REMON> unload
```

To check the content of the REMON script file, you can use the cat command. In this case, enter an accurate script and agent ID.

```
REMON> cat NET.R01
```

15.5. Data Control

The REMON module can process the data collected by the REMON script or from the external data source. Data processing is done by the class that implements the remon.IFilter interface. It is called a filter because overlapping is allowed.

It provides frequently used filters, and a user can add an arbitrary filter for use.

Notice: Set a filter for each REMON script.

15.5.1.Basic Filters

To check the list of filters, enter the filter command in your control console.

```
REMON> filter
```

To check a filter with a specific name, use the filter command and enter the name as a parameter.

```
REMON> filter [filter name Prefix]


ex) filter ALERT
```

A filter is applied to the script ID of the REMOND script. It is not relevant to the agent ID. If you want to check the status of filter use for each REMON script, enter the lsf command.

```
REMON> lsf
```

To check the use of a filter for a certain script, use the lsf command and enter the script ID as a parameter.

```
REMON> lsf [Script ID Prefix]


ex) lsf NET
```

If you want to add a specific filter to the REMON script, use the addf command.

```
REMON> addf <Script ID> <Filter name>
```

To update the filter configuration, use the appf command and enter the script ID as a parameter. The filter configuration is then updated.

```
REMON> appf NET
```

To delete the filter that has been applied to the REMON script, use the delf command. You can check the filter index by using the lsf command.

```
REMON> delf <Script ID> <Filter index>

예) delf NET 1
```

To remove all filters applied to the REMON script, use the rmf command and enter the script ID as a parameter.

```
REMON> rmf <Script ID Prefix>

ex) rmf NET
```

If you want to replace a specific filter applied to the REMON script with another filter, then use the setf command and enter the script ID, the filter index and the filter name as parameters.

```
REMON> setf <Script ID> <Filter index> <New filter name>

ex) setf NET 1 ALERT
```

To add a new filter to the REMON script, use the insf command and enter the script ID, the filter index and the filter name as parameters.

```
REMON> insf <Script ID> <Filter index> <New filter name>

ex) insf NET 1 ALERT
```

The following is a description of the main filters. For descriptions of other filters, use the filter command in the control console.

15.5.1.1. DB_SAVE

By default, the REMON data sent from the REMON module to the JENNIFER server performance database is not stored in the JENNIFER server. If you want to store certain REMON data in the JENNIFER server performance database, use the SERVER_DB_SAVE filter. For example, if you want to store the REMON data sent from the REMON script with a script ID equal to NET in the JENNIFER server performance database, use the following command.

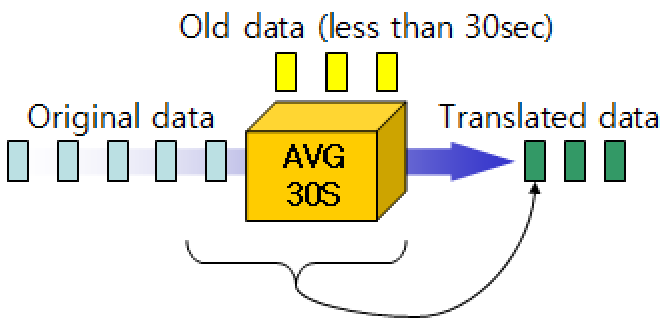
```
REMON> addf NET DB_SAVE 30
...
REMON> appf NET
...
```

The last parameter in the addf command specifies the storage period, and is expressed in seconds. Since it is set to 30 in the example, the JENNIFER server stores the REMON data in the database every thirty seconds.

15.5.1.2. AVERAGE

By default, the REMON data that has been collected the most recently is sent to the JENNIFER server. However, it is sometimes necessary to obtain an average from the data collected in a certain period of time, in order to eliminate extreme values. To achieve this, the AVERAGE filter is used. There are various types of AVERAGE filters, including AVG_30S, AVG_5M, and AVG_60S.

Figure 15-2:Execution of AVG_30S



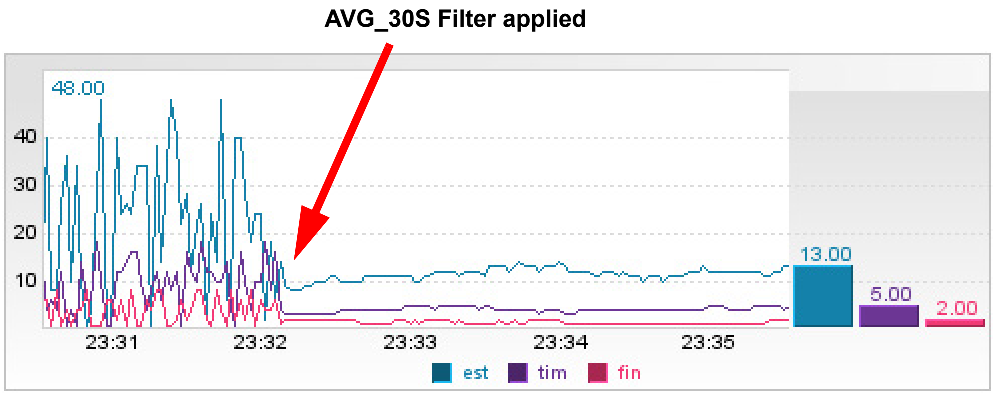
Notice: AVG_30S is a thirty-second average. The incoming data to the filter is stored for thirty seconds, and is then converted to average data when the new data arrives.

To apply the AVG_30S filter to the REMON script with a script ID equal to NET, execute the following command.

```
REMON> addf NET AVG_30S
...
REMON> appf NET
...
```

The following chart represents the change that takes place after using the AVG_30S filter.

Figure 15-3:Application of the AVG_30S Filter



The AVG_5M filter processes the data into a five-minute average, while the AVG_60S filter processes it into one-minute average.

15.5.1.3. SEND

The SEND filter is used to modify the data transmission period.

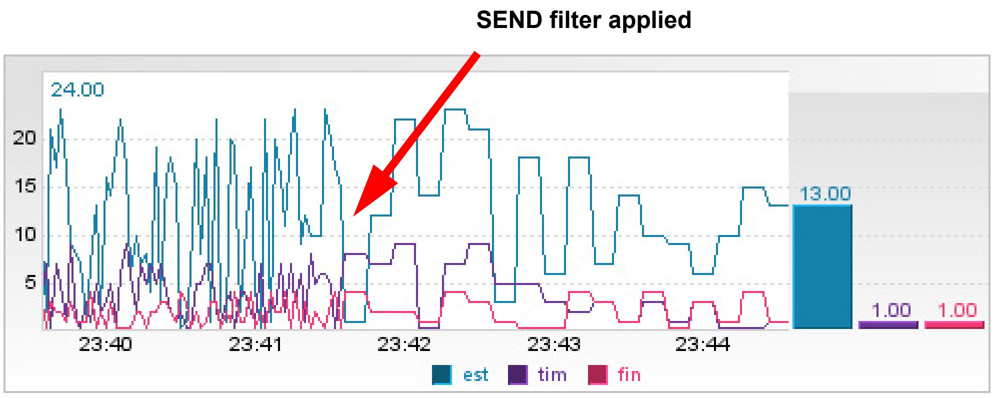
If you want to set the data transmission period for the REMON script with a script ID equal to NET to 10 seconds, use the SEND filter as follows.

```
REMON> addf NET SEND 10
...
REMON> appf NET
...
```

The last parameter of addf sets the transmission period in seconds.

The following chart represents the change that takes place after using the SEND filter.

Figure 15-4:Application of the SEND Filter



Although the REMON script collects the data once per second, the SEND filter only sends the data 10 seconds after the transmission of the final data. Therefore, if the SEND filter is used, the data will be observed as shown in the above figure.

15.5.1.4. STOP

The STOP filter is used when the data that has been collected by the REMON script should not be sent to the JENNIFER server. If this filter is used, the REMON script will collect the data, but does not send it to the JENNIFER server.

```
REMON> addf NET STOP
...
REMON> appf NET
...
```

15.5.1.5. ALERT

Using the ALERT filter, you can issue an alert for the data collected by the REMON script. When you add the ALERT filter, you must set the alert condition and messages.

For instance, the REMON script with a script ID equal to NET collects the int type of data using the est, tim and fin fields.

```
REMON> data NET
2008-07-31 21:20:11.809:NET:R01(est=5,tim=2,fin=1)
```

If the est field is greater than 20, you must use the following ALERT filter to issue the USER_DEFINED_FATAL alert:

```
REMON> addf NET ALERT "est > 20"  "FATAL: The est is over 20(${est})"
...
REMON> appf NET
...
```

When the parameter has some space within it, surround it with a ["].

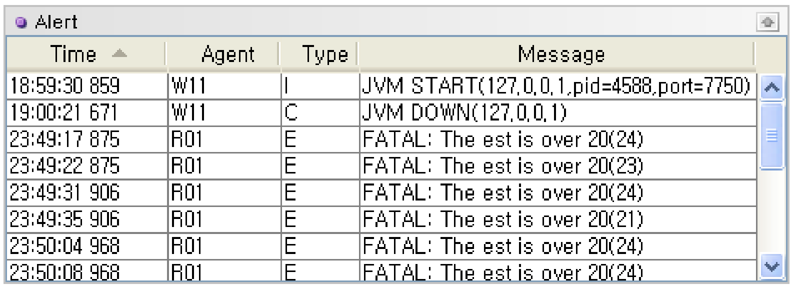
The first parameter can use a variable and an operator. A data field name can be used for a variable, while the available operators include arithmetic (+, -, *, /), boolean (&&, ||), parenthesis, and ternary operators (? :).

The second parameter is an alert message and it should start with either FATAL, ERROR or WARN. The following alert types are available.

- FATAL - The JENNIFER server issues a USER_DEFINED_FATAL alert.
- ERROR - The JENNIFER server issues a USER_DEFINED_ERROR alert.
- WARN - The JENNIFER server issues a USER_DEFINED_WARNING alert.

A message can include a certain value for the data field. If written as ${field name}, the message will be replaced with a value, and then sent to the server.

Figure 15-5:Alert Chart



Alert

Time	Agent	Type	Message
18:59:30 859	W11	I	JVM START(127.0.0.1,pid=4588,port=7750)
19:00:21 671	W11	C	JVM DOWN(127.0.0.1)
23:49:17 875	R01	E	FATAL: The est is over 20(24)
23:49:22 875	R01	E	FATAL: The est is over 20(23)
23:49:31 906	R01	E	FATAL: The est is over 20(24)
23:49:35 906	R01	E	FATAL: The est is over 20(21)
23:50:04 968	R01	E	FATAL: The est is over 20(24)
23:50:08 968	R01	E	FATAL: The est is over 20(24)

Like other alerts, the REMON alerts are stored in the ALERT_01~31 tables of the JENNIFER server. When required, you can browse through the data using the report template and the query tools.

If the same alerts are issued consecutively from the same REMON data, only the first alert will be sent to the JENNIFER server, while the remaining alerts will be ignored. If you want to send all of the alerts to the JENNIFER server, use the ALERT_ALWAYS filter.

```
REMON> addf NET ALERT_ALWAYS
...
REMON> appf NET
...
```

15.5.2.Adding a New Filter

A user can make a new filter for use.

15.5.2.1. Implementing the Filter Class

First, you must prepare the class that implements the remon.IFilter interface. In general, you must inherit from the remon.AbstractFilter class that implements the remon.IFilter interface in order to make a new filter.

Notice: Before compiling it, you must register the REMON_HOME/lib/sys/remon.jar file in the class path. Next, pack the compiled class into a JAR file and copy it into the REMON_HOME/lib/script directory. Then restart the REMON module. If the class has been modified, repeat the above steps and restart the REMON module.

The remon.IFilter interfaces is as follows:

```
package remon;

public interface IFilter {
    public void open(String script, String[] args);
    public RmPacket process(RmPacket pack);
    public void close();
}
```

The remon.AbstractFilter class is as follows:

```
package remon;

import com.javaservice.jennifer.protocol.RmPacket;

abstract public class AbstractFilter implements IFilter {

    public void open(String script, String[] args) {
    }

    public void close() {
    }

    public RmPacket process(RmPacket rmdata) {
        return null;
    }

}
```

The open, close and process methods can be implemented. The characteristics of each method are as follows:

- open - Called when the filter is loaded
- close - Called when the filter is removed
- process - Called each time the REMON data is collected

A filter is applied to each REMON script. Therefore, when a filter is first applied to the REMON script, the open method is called. This method initializes the filter. The first parameter of the open method is a java.lang.String type of script ID. The second parameter is a java.lang.String[] type that represents the list of parameters to be set when a filter is added by using the addf command in the control console. For example, let us assume that the following filter is added:

```
REMON> addf NET DOUBLE 10 true
```

```
// script는 NET
public void open(String script, String[] args) {
    String second = args[0]; // 10
    String isOrder = args[1]; // true
}
```

The close method is called when the filter is removed. Therefore, the close method implements the logic to release the resource assigned by the open or process method.

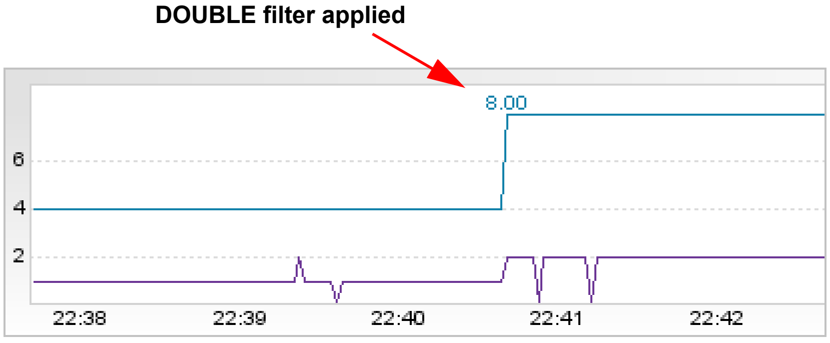
The process method performs the actual processing of the data. This method is called each time the REMON data is collected. If this method returns a null value, then no data is sent to the following filters or to the JENNIFER server.

The original REMON data is transferred to the RmPacket type of parameters of the process method. For example, you can implement a filter that doubles the data value, as follows:

```
package example;

import remon.AbstractFilter;

import com.javaservice.jennifer.protocol.RmPacket;
import com.javaservice.jennifer.type.FIELD;
import com.javaservice.jennifer.type.INT;
import com.javaservice.jennifer.type.TYPE;

public class DoubleFilter extends AbstractFilter {

    public RmPacket process(RmPacket pack) {
        int count = pack.count();
        for (int i = 0; i < count; i++) {
            FIELD field = pack.getField(i);
            TYPE type = field.getValue();
            if (type.getType() == TYPE.INT) {
                INT t = (INT) type;
                t.value *= 2;
            }
        }
        return pack;
    }

}
```

The following chart illustrates the manner in which the data is changed after using the DOUBLE filter.

Figure 15-6:Application of the DOUBLE Filter



For more details on the use of the RmPacket class, please refer to [Using the RmPacket Class].

15.5.2.2. Registering a Filter

You can register a new filter as follows:

1. First, pack the class that implements the remon.IFilter interface into a JAR file, and copy it into the REMON_HOME/lib/script directory.
2. Set the filter information in the REMON_HOME/conf/filter.conf file.

```
[DOUBLE]
    class = example.DoubleFilter
    option =
    desc = Make the value into a double.
```

[] The filter name is DOUBLE.

Using the class attribute, you must set the class that implements the remon.IFilter interface. In the option attribute, you must describe the parameter used by the filter, and in the desc attributes, you must describe the filter itself.

3. Restart REMON.

4. Verify that the filter has been registered correctly by using the filter command in the control console.

```
REMON> filter DOUBLE
Available Filters
------------------
[DOUBLE]
    class = example.DoubleFilter
    option =
    desc = Make the value into a double.
```

15.6. Managing and Viewing the Data

The REMON module sends its data to the JENNIFER server. Now, we will describe the mannet in which the JENNIFER server can manage and check the REMON data.

15.6.1.Debugging the REMON data at the JENNIFER Server

To check the REMON data in the start console of the JENNIFER server, set the remon_debug option of the JENNIFER server to true. The default configuration is false.

```
remon_debug = true
```

15.6.2.Saving the Data

REMON data is stored in file and DB in JENNIFER SERVER.

15.6.2.1. Saving in DB

By default, the JENNIFER serve store the REMON data as a ISAM file type in the database. However, data sent by the REMON script to which the SERVER_SAVE filter has been applied is stored in the both file and database at the same time. But, if you set the enable_remon_data_save option of the JENNIFER server into 'false', the JENNIFER server does not store the data in the file. Default is true.

```
RM_<Script_ID>_01~31
```

For example, the data sent by the REMON script whose script ID is CPU is stored in the RM_CPU_01~31 table.

All tables containing REMON data have the following columns:

Table 15-6: REMON Table

Column	Type	Restraints	Descriptions
LOG_TIME	BIGINT		Record time
SCRIPT	VARCHAR(256)		Script ID
AGENT	VARCHAR(128)		Agent ID
LOG_DT	CHAR(8)		Date(YYYYMMDD)
LOG_HH	CHAR(2)		Hour(HH)
LOG_MM	CHAR(2)		Minute(MM)
LOG_SS	CHAR(2)		Second(SS)
MESSAGE	VARCHAR(32672)		REMON message

Another column corresponds to the field name of the REMON data. The type of column depends on the field type.

Warning: If the field type is modified, then the REMON data cannot be saved properly. In this case, modifying the REMON script ID is recommended.

15.6.2.2. Saving in File

If you do not want to store the REMON data at the JENNIFER server regardless of whether or not the SERVER_SAVE filter has been applied, set the enable_remon_file_save of the JENNIFER server to false. The default configuration is true.

Notice: When you search the saving data, reser to [TOOLS | REMON Data Serarch].

15.6.3.Deleting the Data

The JENNIFER server automatically deletes the REMON data from the database using CleanerActor scheduler. By default, the following configuration is used for the JENNIFER server.

```
time_actor_11 = com.javaservice.jennifer.server.timeactor.CleanerActor
02 REMON DAY 7
```

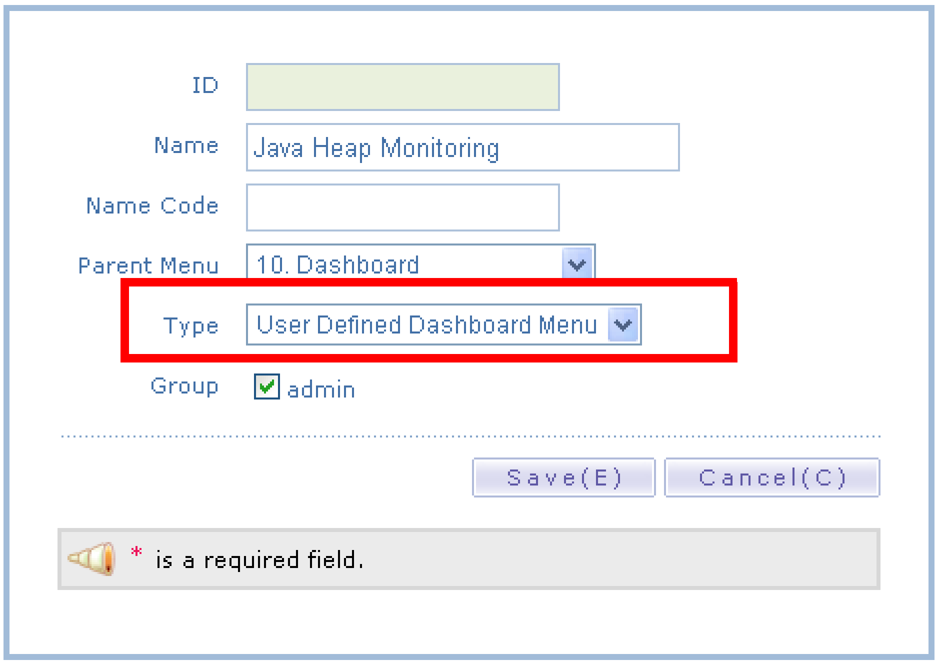
As shown above, REMON data older than seven days will be deleted every day at 2 AM.

15.6.4.REMON Data Monitoring Using the User Defined Dashboard

Using the user defined dashboard, you can monitor the REMON data. For more details on the user defined dashboard, refer to **[User Defined Dashboard].**

By selecting the **[Properties | Menu Setting]** menu, you can add a user defined dashboard menu.

Figure 15-7:Adding a User Defined Dashboard Menu

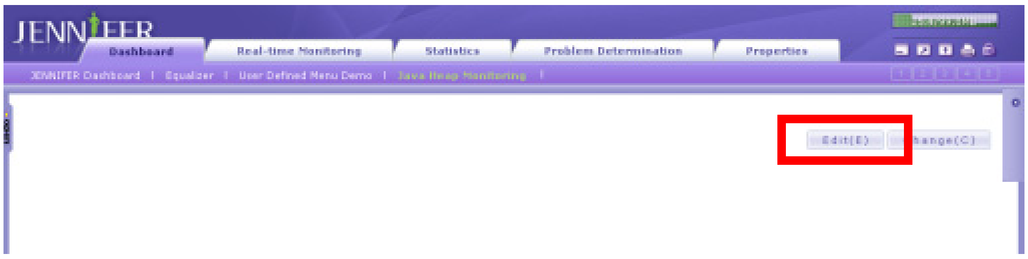


ID	
Name	Java Heap Monitoring
Name Code	
Parent Menu	10. Dashboard
Type	User Defined Dashboard Menu
Group	✓ admin

* is a required field.

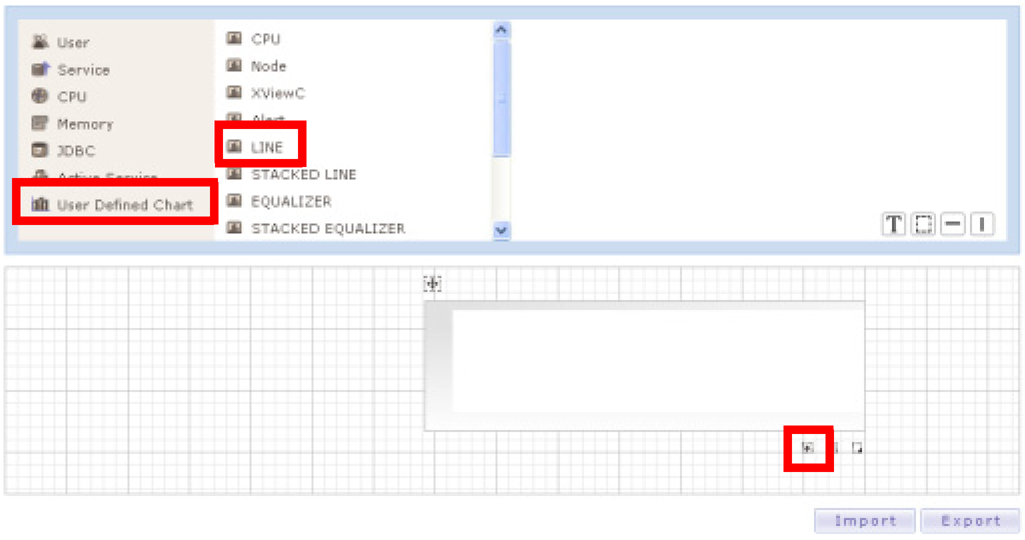
After adding a menu, select it. By clicking the **[Edit]** button on the right, you can move to the user defined dashboard editing screen.

Figure 15-8:User Defined Dashboard Menu



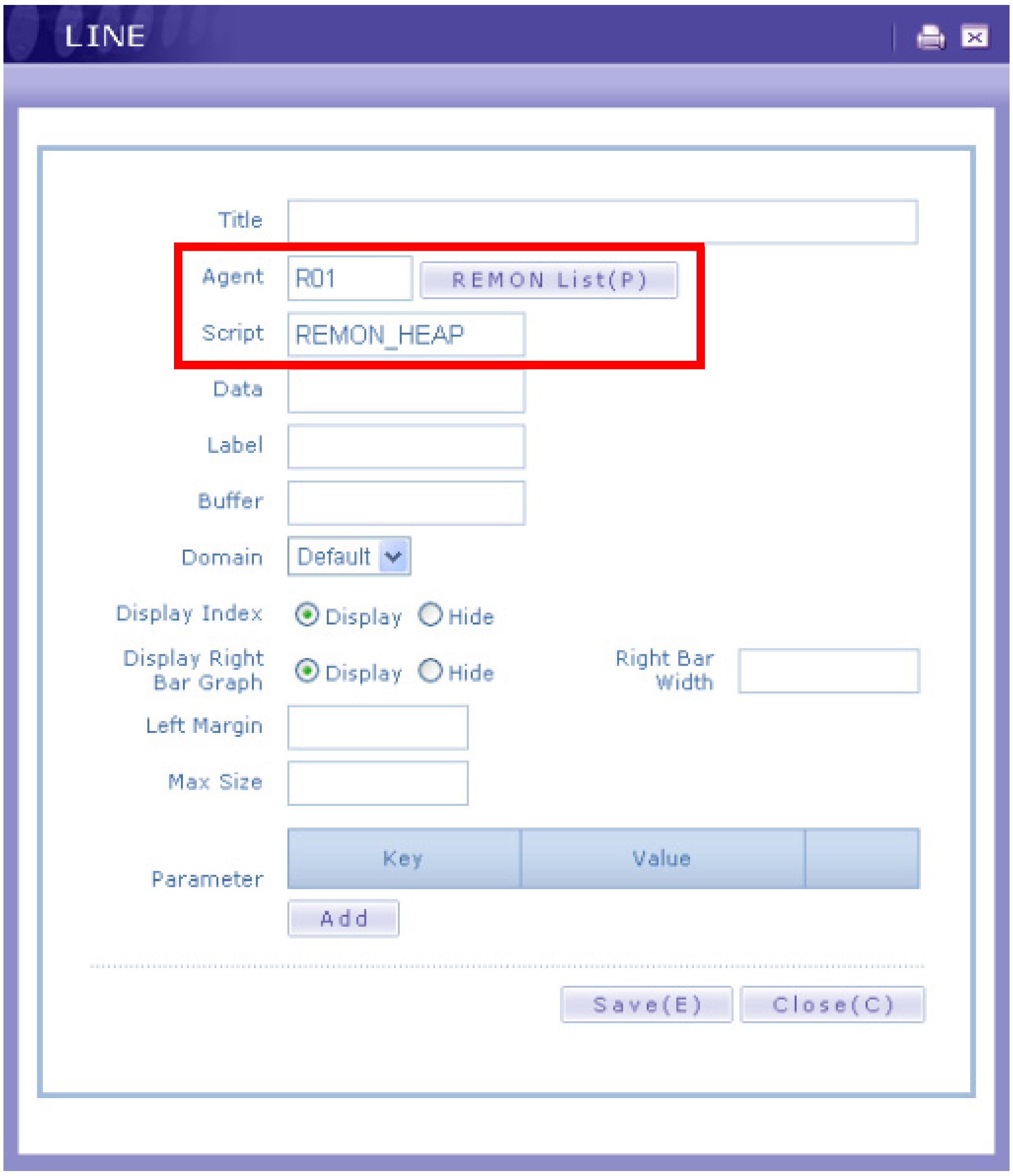
Select a line chart from the [User Defined Chart] group of the chart selection area. A chart will appear on the screen. Using the drag and drop method, move the chart into the chart configuration area.

Figure 15-9:LINE Chart Selection



If you click the **[Option Setting]** icon in the lower right corner of the chart, an option setting pop-up window will appear.

Figure 15-10:Option Setting Pop-up Window

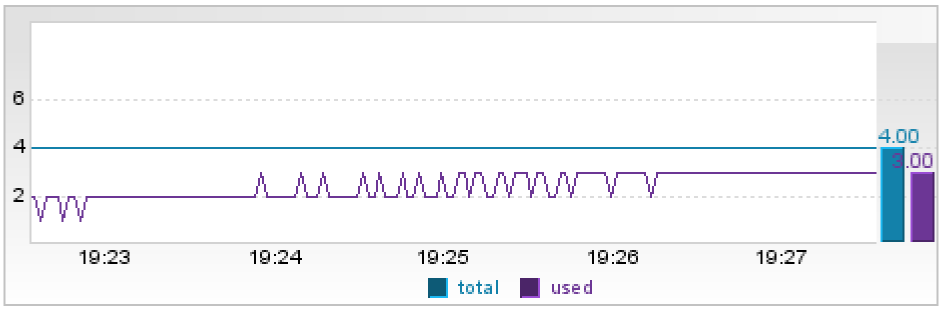


Title	
Agent	R01
Script	REMON_HEAP
Data	
Label	
Buffer	
Domain	Default
Display Index	[✓ Display] Hide
Display Right Bar Graph	[✓ Display] Hide
Right Bar Width	
Left Margin	
Max Size	
Parameter	

In the option setting pop-up window, click the **[REMON list]** button. After selecting the REMON data to monitor, click the **[Select]** button at the bottom. Then agent and script ID of the REMON data will then be entered into the agent and script fields.

Click the **[Save]** button at the bottom of the option setting pop-up window. The line chart will now display the REMON data.

Figure 15-11:REMON Data Monitoring with a LINE Chart



15.7. LogWatcher

LogWatcher acts as a log monitor. It can detect patterns in a text log file recorded by various applications or OS, and generate events to send the data to the REMON module.

15.7.1.Installation and Configuration

LogWatcher is located in the REMON_HOME/logwatcher directory, and it is comprised of log.sh, logwatcher.jar and logw.conf files. To install it, you must first copy the logwatcher directory to the OS that you want to monitor.

Next, you will need to set the various options in the logw.conf file. You must set the common options at the beginning of the logw.conf file. These options should be located ahead of the log file configuration that starts with [XXX].

The main options are given as follows.

```
agent = L01
control_port = 6001
target_host = 127.0.0.1
target_port = 7701
```

The following is a description of the common options.

- agent - This sets the agent ID of the REMON data collected by LogWatcher. It has the same constraints as the REMON agent ID.
- control_port - LogWatcher is managed using the control console. With this option, you can set the TCP port number that receives requests from the control console.

- target_host - This sets the REMON IP address to which the REMOND data collected by LogWatcher is transmitted.
- target_port - This sets the REMON port number to which the REMOND data collected by LogWatcher is transmitted.

Next, you will need to set the monitoring log file using LogWatcher in the logw.conf file. Each log option starts with [XXX] where XXX is a combination of uppercase letters and numbers. This is the script ID of the REMON data collected by LogWatcher. The following sample illustrates two sets of log file monitoring, named ALOG and BLOG.

```
[ALOG]
...

[BLOG]
...
```

The following is a detailed description of ALOG.

```
[ALOG]
file = /tmp/${today}.log
rule_01 = [*]ABC, ok
rule_02 = [4]###, fatal
```

- file - This sets the path for the monitored file. If its name depends on the data, use the ${today} variable.
- rule_nn -The rule_01 and rule_02 options are used to set the rules. Each rule is separated by [,]. The first value is the text testing condition, while the second value is the name of the rule, and is also used as the field name of the REMON data. Therefore, the REMON data in the above example have two fields (the ok field for rule_01, and the fatal field for rule_02).

If you do not want to apply the rule, but wish to transmit the original log entry to the REMON module, set the send_raw_data option to true.

```
[BLOG]
file = /tmp/access.log
send_raw_data = true
```

When the logw.conf file has been changed, restart the LogWatcher module.

LogWatcher is implemented in Java. For this reason, you must set Java as a JAVA_HOME environment variable.

Notice: To run LogWatcher, use Java 1.4.2 or higher.

The JAVA_HOME environment variable can be set in the env.sh(env.bat) files of the REMON_HOME/logwatcher directory. If you are using linux or unix, modify it as follows:

```
JAVA_HOME=/usr/java
```

If you are using Windows, modify it as follows:

```
set JAVA_HOME=C:/Program Files/Java/jdk1.6.0_07
```

To run LogWatcher, run the logw.sh file in the REMON_HOME/logwatcher directory.

```
./logw.sh
```

To stop LogWatcher, run the shutdown.sh file in the REMON_HOME/logwatcher directory.

```
./shutdown.sh
```

LogWatcher can be managed using the control console. To start the control console, run the console.sh file in the REMON_HOME/logwatcher directory.

```
./console.sh
```

To check the monitoring status, enter the ls command in the control console.

```
logw> ls
```

15.7.2.LogWatcher and REMON Data

LogWatcher reads each line from a log file to see if it matches any rule. It counts the number of matches with each rule when it organizes the REMON data. Thus, each rule is a field, and the number of matches with the rule is the value of the field.

The REMON data is sent to the REMON module once every five seconds.

Notice: If the send_raw_data option is set to true, the data is sent once per second.

Let us assume that there are 100 lines of ALOG logs accumulated in the course of five seconds. If ten lines match rule_01 while five lines match rule_02, then the following data will be sent to the REMON module.

```
Agent ID - L01
Script ID - ALOG
ok field value - 10
fatal field value - 5
```

15.7.3.Setting the Rule

A rule is comprised of a [Test Condition] and a [Field Name]. A test condition consists of a keyword that sets the [Word Location] and the word to be tested. To combine various conditions, you can use && or || operators. However, you may not use both && and || in a single rule.

```
[*] - any position
[n] - n's position (n is any positive number like 0, 1... 999)
[$] - end of the line
```

The following is a list of example test conditions that can be set. A rule is applied to each row.

- [*]ABC - If the text string ‘ABC’ is in the row, the rule is satisfied.
- [4]BBB - If the text string ‘BBB’ is found starting from the fifth letter, the rule is satisfied.
- [$]XXX - If the row ends with the text string ‘XXX’, the rule is satisfied.
- [*]AAA || [$]XXX - If the text string ‘AAA’ is in the row, or if the row ends with the text string ‘XXX’, the rule is satisfied.

Notice: The text string such as [*]AAA || [$]XXX && [*]BB is not used, as && and ||.cannot be combined.

15.7.4.Monitoring the Apache Web Server Access Log using an X-ViewC Chart

Now, a method for monitoring the Apache web server access log using LogWatcher, REMON and an X-ViewC chart will be described.

Notice: An X-ViewC chart displays the distribution of a log file in a certain format. Each row in a log should include the time, the value(response time or utilization), and the name (URL). You can monitor the Java GC log and the Apache web server access log by using an X-ViewC chart.

First, using LogWatcher, read in an Apache web server access log file. You must install LogWatcher on the server in which the Apache web server is installed, and set the logw.conf file as follows:

```
agent = L01
control_port = 6001
target_host = 127.0.0.1
target_port = 7701

[ACCESS]
send_raw_data = true
file = /usr/local/apache2/logs/www_access_log
```

If you want to monitor it using an X-ViewC chart, the agent ID must be three characters in length.

Run LogWatcher and REMON. Then apply the XVIEW_APACHE filter to the REMON data sent from LogWatcher in the REMON control console.

```
REMON> addf ACCESS XVIEW_APACHE
...
REMON> appf ACCESS
...
```

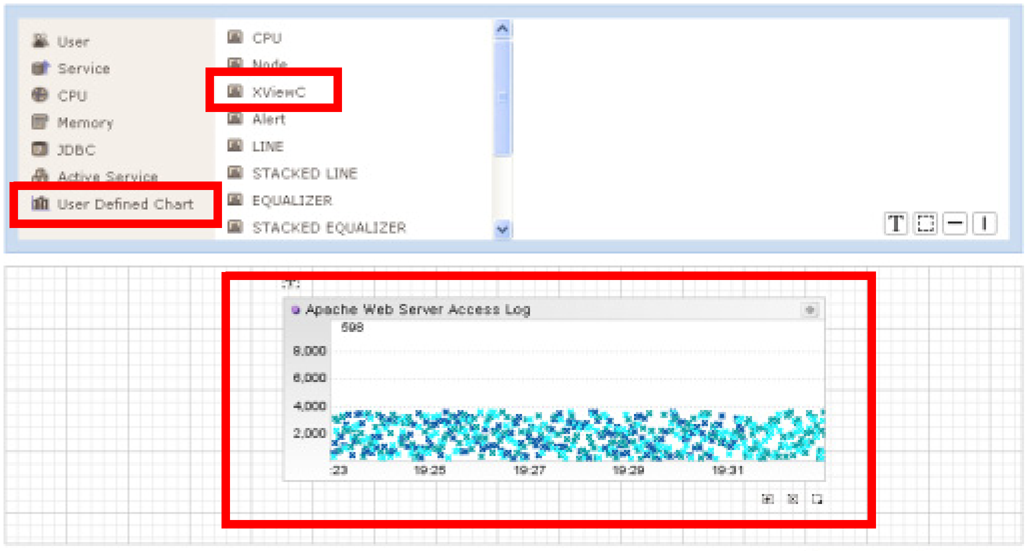
By default, the Y-axis of an X-ViewC chart represents access bytes. If you want to set the Y-axis to represent response time, use the following filter.

```
REMON> addf ACCESS XVIEW_APACHE 2
...
REMON> appf ACCESS
...
```

Notice: The Apache web server access log file should record the response time as well. For more on the configuration methods, refer to an Apache web server manual.

Finally, select the user defined dashboard editing screen of the JENNIFER server. In the [User Defined Chart] group of the chart selection area, select the X-ViewC chart. A chart will appear on the screen. Using the drag and drop method, place the chart in the chart configuration area.

Figure 15-12:X-ViewC Chart



15.8. Using the RmPacket Class

To implement a class script, a filter or ExtraAgent, you must first understand the RmPacket class. For this purpose, a method for using the RmPacket class will be also described.

The RmPacket class name, including the package name, is as follows:

```
com.javaservice.jennifer.protocol.RmPacket
```

RmPacket is the class that implements the com.javaservice.jennifer.protocol.IRmPacket interface, and it represents the REMON data. Through the method provided in the IRm-Packet interface, you can check the basic information of the REMON data.

```
package com.javaservice.jennifer.protocol;


public interface IRmPacket {
    public String getAgent();
    public String getScript();
    public long getTime();
    public byte[] toBytes() throws Exception;
}
```

The getAgent method returns the agent ID of the REMON data, while the getScript method returns the script ID of the REMON data. The getTime method returns the created time of the REMON data. Finally, the toBytes method returns the byte format of the REMON data.

Notice: When implementing a class script, a filter or ExtraAgent, you do not need to use the toBytes method.

You can organize and read the REMON data using the method provided by the RmPacket class. First, let us describe how the REMON data is collected bycreating a new RmPacket object using a class script and ExtraAgent.

The first method for creating the RmPacket object is as follows. Using the script and agent ID as parameters, execute a constructor.

```
RmPacket packet = new RmPacket("HEAP", "R01");
```

In this case, the time of REMON data collection is the current time. You can set the time of REMON data collection as follows:

```
long time = ...
RmPacket packet = new RmPacket(time, "HEAP", "R01");
```

The REMON data consists of fields. Each field is added using the addField method of the RmPacket class. The first parameter of the addField method is a java.lang.String type that stands for a field name, and the second parameter is a com.javaservice.jennifer.type.TYPE type that uses the child class that matches the field data type.

```
RmPacket packet = ...
packet.addField("max", new INT(4));
```

The INT class is used to add an int type of data. The following table describes the TYPE class and its child classes.

Table 15-7: TYPE Class

Class name	Description	Type of database columns
BTYE	1 byte data	This type of field is not stored in the database.
BYTES	32,767 byte data (Short, maximum)	This type of field is not stored in the database.
TINY_BYTES	256 byte data	This type of field is not stored in the database.
LONG_BYTES	2,147,483,647 byte data (Integer, maximum)	This type of field is not stored in the database.
SHORT	short type of data	INT
INT	int type of data	INT
LONG	long type of data	INT
FLOAT	float type of data	REAL
DOUBLE	double type of data	REAL
STRING	java.lang.String type of data. The max number of letters is 32,672.	VARCHAR(32672)
TINY_STRING	java.lang.String type of data. The maximum number of characters is 256.	VARCHAR(257)
LONG_STRING	java.lang.String type of data. The maximum number of characters is 2,147,483,647.	This type of field is not stored in the database.

You can assign a value to each TYPE class using the constructor.

```
new INT(4);
new FLOAT(3.2f);
new STRING(“WARNING”);
```

You can implement a filter or check and modify the REMON data using the RmPacket object. You can use the count method of the RmPacket class to check the number of fields.

```
int count = packet.count();
```

Next, obtain the FIELD object representing the field using the getField method that uses the index as a parameter.

```
int count = packet.count();
for (int i = 0; i < count; i++) {
    FIELD field = packet.getField(i);
}
```

Using the getName method of the FIELD object, you can check the field name, and using the getValue method, you can obtain the TYPE object that represents a value.

```
for (int i = 0; i < count; i++) {
    FIELD field = packet.getField(i);
    String name = field.getName();
    TYPE type = field.getValue():
}
```

The actual type of the TYPE object can be obtained by comparing the value returned by the getType method and the constant of the TYPE class. The constant's name is the same as the name of the child class of the TYPE class.

```
TYPE type = field.getValue();
switch (type.getType()) {
case TYPE.INT:
    INT t = (INT) type;
    ...
    break.
case TYPE.STRING:
    STRING t = (STRING) type;
    ...
    break;
}
```

To check the value of each TYPE object, use the value member field. The type of this field varies depending on the child class of the TYPE class.

```
INT t = (INT) type;
int value = t.value;
```

When it is necessary to change the value, you can modify the value member field of the TYPE class on your own.

```
TYPE type = field.getValue();
if (type.getType() == TYPE.INT) {
    INT t = (INT) type;
    t.value *= 2;
}
```

Newly Upgraded Jennifer

Application Performance Management (APM)

Application Performance Management (APM) is a process of developing a progressive system that effectively monitors the availability of enterprise applications, identifies and resolves the application performance problems, and predicts/prevents future application performance problems, considering the limit and potential of the resources available to an organization's IT infrastructure and its supporting systems. Unlike the traditional system management (SMS and NMS), APM enhances the organization's ability to manage the enterprise application services and resolve application performance problems, resulting in reduced Total Cost of Ownership (TCO) and enhanced customer service.

JENNIFER

JENNIFER is the Application Performance Management (APM) solution developed by JenniferSoft Inc. JENNIFER provides total application performance management and operational support services for enterprise web system, performing tasks such as real-time resource and service monitoring, immediate performance problem diagnosis, and effective performance problem resolution. JENNIFER is the premier APM solution in Korea.

Dynamic Profiling

JENNIFER can register additional package, class, method and/or activate/deactivate transaction-profiling without restarting the web application server.

Dynamic StackTrace

The traditional method for extracting Java Full StackTrace is intentionally causing an exception/error for an application resource and outputting it onto the stacktrace; JENNIFER can register a class/method during operation, allowing dynamic full stacktrace of additional class/method without changing the application source code.

Monitoring per Domain

In a large scale of enterprise environment, many different business systems may exist, triggering a need for a solution that individually monitors each business system under integrated one view. JENNIFER provides performance management capability per domain that allows the user to allocate multiple business systems into different domain and manage each system under one umbrella.

Extended Monitoring Adaptor

JENNIFER extracts the performance data from web application server and communication between WAS and other system devices. JENNIFER features Extended Monitoring Adaptors Functionality (EMAF) that allows performance data from other system devices to be extracted and inputted into JENNIFER for analysis and reporting.

JENNIFER®
Enterprise Web Business Monitoring

www.jennifersoft.com
Starvalley 1104, 60-11 Gasan-dong, Geumcheon-gu, Seoul, Korea.
TEL. +82.2.2027.0391~9 **FAX.** +82.2.2027.0390 **E-mail.** sales@jennifersoft.com