



CheckIT
Management Platform
Version 0.5

User & Administration Manual



Jilroy Technologies LTD 2010

Jilroy Software makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Jilroy Software shall not be held liable for errors contained herein or direct, indirect, special, incidental, or consequential damages in connection with the furnishing, performance, or use of this material.

All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Jilroy Software. The information contained in this document is subject to change without notice.

Copyright © 2000-2010 Jilroy Software. All rights reserved.

Reproduction, adaptation, or translation of this document without prior written permission is prohibited, except as allowed under the copyright laws.

Microsoft® is a US registered trademark of Microsoft Corporation.

Windows® and MS-Windows® are US registered trademarks of Microsoft Corporation.

Pentium® is a US registered trademark of Intel Corporation.

UNIX® is a registered trademark of The Open Group.

All other company and product names may be trademarks or registered trademarks of their respective owners.

Table of Contents

Preface.....	8
Introduction.....	10
What is CheckIT.....	11
The CheckIT Management Platform Server components.....	12
Current collectors.....	13
ICMP Monitor.....	13
The SNMP collector.....	14
The SQL collector.....	14
The CSV collector.....	15
The HTTP Status monitoring.....	15
The WMI collector.....	15
The Open TCP Port collector.....	16
The Discovery Fields data collector.....	16
The SNMP Traps collector.....	16
Data collection.....	17
Scheduling mechanism.....	18
RRD Daemon.....	18
Extendability in the GUI.....	18
The Network Discovery component.....	18
Not yet another monitoring platform!.....	19
Additional Advantages.....	20
Getting Started on Unix Platforms.....	21
Getting the software.....	21
Installing the CheckIT Management Platform	21
Execution Permission.....	23
Post Installation scripts on UNIX.....	23
Post-Installation scripts.....	23
Database configuration.....	24
Automatic Startup.....	24
removing all presence of the product.....	25
remove the crontab additions.....	25
GUI only Installation on Unix Platforms.....	26
Preparing the system for install.....	26
TCP buffer settings.....	26

Unix System Requirements.....	27
Installation on UNIX.....	27
Getting the software.....	27
Installing the GUI.....	27
removing all presence of the product.....	29
Getting Started on Windows Platforms.....	30
Windows System Requirements.....	30
Installation on Windows.....	30
Getting the software.....	30
Installing the CheckIT Management Platform v.3.0.....	30
License.....	36
Database configuration.....	36
Automatic Startup.....	37
Un-installation on Windows.....	37
GUI only Installation on Windows Platforms.....	40
Windows System Requirements.....	40
Installation on Windows.....	40
Getting the software.....	40
Installing the CheckIT Management Platform v.3.0 GUI.....	40
License.....	44
Un-installation on Windows.....	44
Configuration.....	47
Config.prop.....	47
General.....	51
Database.....	51
Default_monitor_parameters.....	52
ICMP.....	54
ICMP_Backbone_failure.....	54
SNMP.....	55
SNMP_Backbone_failure.....	56
WatchDog.....	56
configGUI.prop.....	57
The SQL Filtering Files.....	64
ICMPSQLFilteringFile.txt.....	65
SNMPSQLFilteringFile.txt.....	67
SQLCommandsSQLFilteringFile.txt.....	70
CSVCommandsSQLFilteringFile.txt.....	73
HTTPCommandsSQLFilteringFile.txt.....	76

SNMPTrapsSQLFilteringFile.txt.....	79
The Agents Technology.....	82
AgentsList.txt.....	83
The ICMP Data Collector agent logic.....	85
The collector functionality.....	85
Status monitoring.....	85
Sending Response time information.....	85
Flow control parameters.....	86
Jitter measurement.....	86
The SNMP Data collector logic.....	88
The collector functionality.....	88
Status monitoring.....	88
Collecting SNMP data.....	88
Flow control parameters.....	89
The SNMP Trap Data collector logic.....	90
The collector functionality.....	90
The SQL Data collector logic.....	91
The collector functionality.....	91
Status monitoring.....	91
Collecting SQL data.....	92
Flow control parameters.....	92
The CSV Data collector logic.....	93
The collector functionality.....	93
Status monitoring.....	93
Collecting CSV data.....	94
Flow control parameters.....	94
The Discovery Field Data collector logic.....	95
The collector functionality.....	95
Status monitoring.....	95
Collecting CSV data.....	96
Flow control parameters.....	96
The HTTP Data collector logic.....	97
The collector functionality.....	97
Status monitoring.....	97
Collecting HTTP data.....	98
Flow control parameters.....	98
The WMI Data collector logic.....	99
The collector functionality.....	99

Status monitoring.....	99
Collecting WMI data.....	100
Flow control parameters.....	100
The CheckIT database scheme	101
Utility commands.....	103
The CheckIT GUI.....	104
Edit Product Config File.....	107
Edit Provisioning File.....	108
Interfaces status.....	118
Service Level Report.....	120
Backup and Restore for the Database.....	125
exportable2csv.....	125
Additional SQL related commands.....	127
Query analyzer for the database.....	127
createdb.....	127
createtables.....	127
dbconf.....	127
loadtableinfo.....	128
sqlexec.....	128
startdb.....	128
stopdb.....	128
truncatetables.....	129
The product Scheduler.....	130
Categories file.....	132
Internal commands.....	135
The product Security mechanism.....	139
The Users and Permissions table.....	139
tblUsers.....	139
tblPermissions.....	140
The connection.prop file.....	140
Client Side	141
General Commands permitted.....	145
The Watch-Dog.....	146
How to define the watched services.....	146
WatchDogServicesList.txt.....	146
WatchDogAgentList.txt.....	147
Watch-Dog Database table scheme.....	147
Backbone failure processing.....	150

What is a backbone failure.....	150
Backbone Failure configuration parameters.....	150
Backbone Failure feature logic.....	151
Logging Daemon.....	152
Configuring the Logging Daemon.....	153
RRD Daemon.....	154
The RRD databases configuration files.....	154
A sample of an RRD configuration file.....	154
RRDDefinitionEntries.....	157
RRDDefinitionEntry.....	158
Parameters.....	158
Archives.....	160
Archive.....	160
Performance related tricks & Tips.....	162
Initial status of an object.....	162
Dynamic load of monitored elements in run time.....	162
Major changes Delta load.....	163
Minor changes in Delta load.....	163
Final Page.....	165

Preface

Welcome to the CheckIT Management Platform . This chapter provides an introduction to the structure and assumptions of this guide.

The Purpose of This Guide

This guide contains information needed to use the CheckIT Management Platform from Jilroy Software efficiently and effectively.

Who Should Use This Guide

This guide is intended for network managers & system administrators familiar with network & systems management and their fundamental concepts.

Prerequisites

It is recommended that you read with this book the additional manuals supplied with this product, dealing with customizing the product.

Organization of This Guide

This guide is structured to reflect the following conceptual divisions:

- Preface – A description of the guide's purpose, intended audience, organization, and conventions.
- Introduction section – A general description of the CheckIT Management Platform and its goals, its way of operation and general guidelines
- Installation section – Information about installing, un-installing, and configuring the CheckIT Management Platform .
- The product's User Interface section – Contains information about the specific User Interface of the product.
- Additional General capabilities and services section – Contains information about general features like the Scheduling mechanism, and the security implementation.
- Final Page – Information about contacting Jilroy Software.

Conventions

The manual uses the following conventions:

- Names of dialog boxes, windows, and unnamed screen areas are displayed in *italics*.
- Names of buttons, tabs, check-boxes, and other screen elements are displayed in **bold**. For example, click **OK** or type the **Start date**.
- **This font** is used for text that you enter.
- `This font` is used for code, directory names, file names, and system activity.
- UPPERCASE is used for keys and acronyms.
- Steps that involve two or more selections from a menu may be presented as a combination of selections separated by an > angled bracket.

For example, when you see **File > New**, click the **File** menu on the menu bar. This will open a drop-down menu. Then select the **New** command.

- Cross-references are underlined. For example, see Chapter 2.
- Hyperlinks are underlined and [blue](#).
- The ⓘ symbol signifies notes, which are used to provide extra or special information regarding the preceding topic.
- The *Italic* font style is used to *emphasize* words and phrases in special cases.

Whats new

Version 0.5

- Event log console support
- Event log History support
- Event actions support

Introduction

Organizations today are moving from reactive management to more proactive management. Proactive management is some times understood as trends analysis, however this is the lowest level of proactive management, which is called off-line proactive management.

There is a need for an On-line proactive management methodology. For example we would like to discover problems and anomalies in the enterprise just before they occur or at the time they occur. We must know about them before our customers discover them. In order to do that we need to monitor all the relevant components of our enterprise in a high frequency, and make sure that all is well.

Examples of such monitoring in the network management area can be:

- Availability monitoring of all important end-points
- Line quality of all clients
- Bandwidth utilization exceptions
- Memory utilization exceptions
- CPU utilization exceptions

and more.

There can be many areas where this type of proactive high frequency monitoring can be relevant:

- The enterprise network
- The enterprise server farm hardware & software
- The end-users desktops, telephone and other related devices
- Business applications and components

and many others.

What is CheckIT

CheckIT is a multi-user, multi-platform flexible & customizable management platform. It supplies among others the following features:

- A network & servers customizable discovery based inventory engine. This engine supports a large number of discovery protocols
- A customizable Multi protocol monitoring platform
- A Multi platform, easy to customize user interface. The UI, supports user authentication and role based rules.
- CheckIT is a Java based multi-platform product, both in the engine part and in the GUI part.

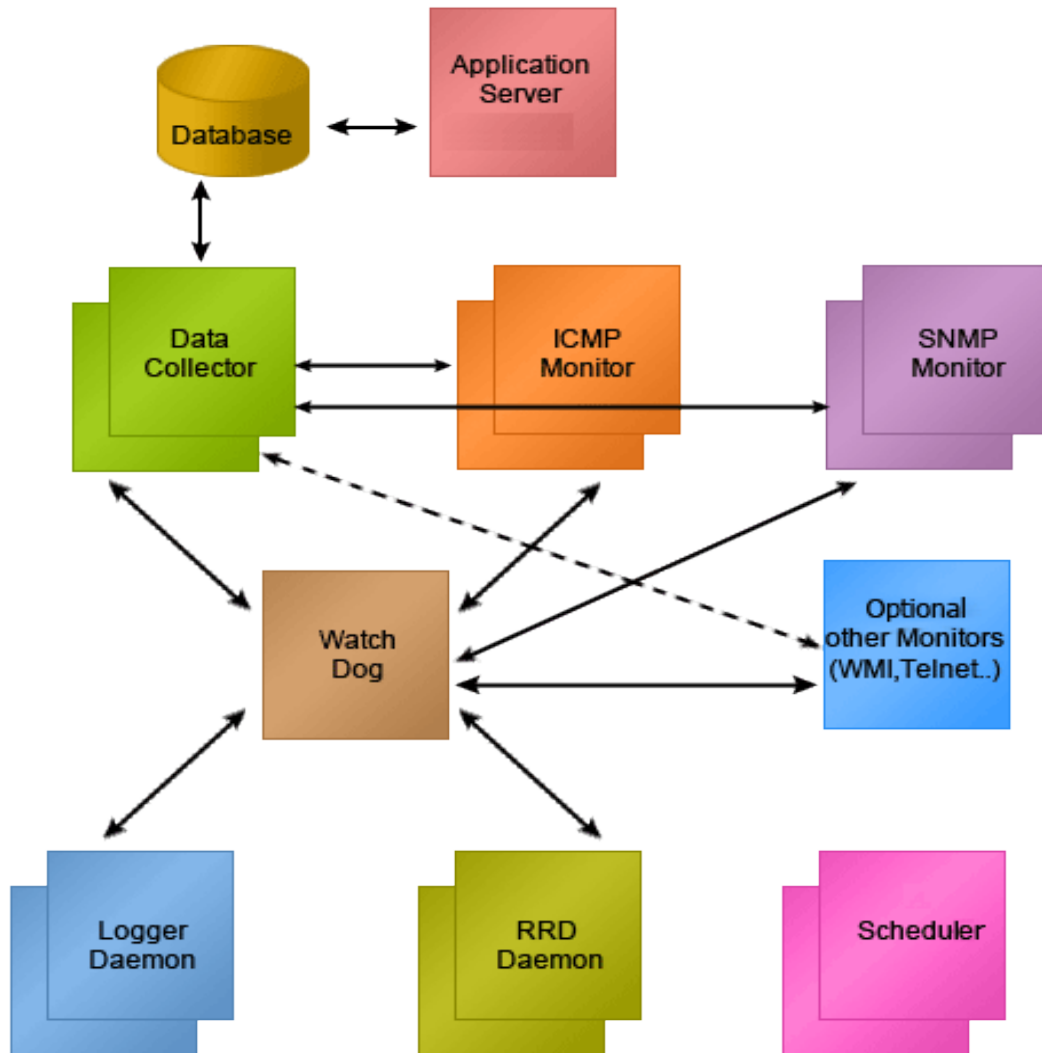
CheckIT is built using components from the Jilroy Monitoring Platform Genie and the Inventory Platform Genie.

Opposed to the Inventory platform and the Monitoring platform that require customization CheckIT was built as an off the shelf product, that can give value immediately after installation.

CheckIT is a limited version of those powerful tools, however it can be an excellent solutions for small and mid size organizations.

The CheckIT Management Platform Server components

The CheckIT Management Platform product is a compound product that contains several services.



The following is a short explanation on some of the product's components.

Current collectors

Following are some of the CheckIT Management Platform supported Monitors:

- ICMP monitor
- SNMP monitor
- SQL monitor
- CSV monitor
- HTTP monitor
- WMI monitor
- TCP port monitoring
- SNMP Traps forwarder

The CheckIT Management Platform was built in an extend-able way, which makes it relatively easy to add new monitoring methods, while re-using all of the infra-structure.

ICMP Monitor

The ICMP collector is an agent of the monitoring platform that sends ICMP echo requests to clients and measures the fact if there is a reply and its time.

It was built to perform this operation very fast using complex algorithms, that enables it to monitor tens of thousands of IP addresses in cycles of around a minute.

It contains many flow control parameters that enables it to work in all kinds of networks.

Status measurement

Its basic function is to monitor the status of IP addresses. It checks if a remote IP address is answering to a Ping request, and when a status change is discovered, it launches a user defined command.

Jitter measurement

The ICMP monitor has a capability of measuring the delay of the answer to the ping. It has the capability to send multiple ICMP requests to a destination in the same cycle of measurement, and

calculate the response time to these requests. It launches a user defined command when a user defined limits are reached, for example:

- The average (or mean) response time is above a given limit
- The standard deviation is above a given value,

The SNMP collector

The SNMP collector is an agent of the monitoring platform that collects SNMP OID values defined by the user. It was built to perform this operation very fast using complex algorithms. This enables it to collect tens of thousands of values in short cycles of around a minute.

It contains many flow control parameters that enables it to work in all kinds of networks, and with all kinds of equipments.

The SNMP Status monitoring

The SNMP collector comes with a basic feature of monitoring the Interface status of network nodes. It calculates the status from the Interface operational status and the Interface administration status of the selected interfaces.

When an interface is changing its status a user based command is launched.

The SQL collector

The SQL collector is an agent of the monitoring platform that collects results of SQL queries defined by the user. It was built to perform this operation very fast using complex algorithms. This enables it to collect 1000s of results from different distant repositories in short cycles of around a minute.

It contains many flow control parameters that enables it to work in all kinds of networks and databases.

The SQL Status monitoring

The SQL collector comes with a basic feature of monitoring the returned values of SQL queries. It calculates if there was a change between 2 consecutive results of a given query. If there was such a change a user based command is launched.

The CSV collector

The CSV collector is an agent of the monitoring platform that collects results of SQL queries

performed on CSV files. The CSV files can be located remotely when there is a way to retrieve them using http, or locally when they can be accessed using the [file:///](#) directive. It was built to perform this operation very fast using complex algorithms. This enables it to collect 1000s of results from different distant files in short cycles of around a minute.

It contains many flow control parameters that enables it to work in all kinds of networks and databases.

The CSV Status monitoring

The CSV collector comes with a basic feature of monitoring the returned values of SQL queries on CSV files. It calculates if there was a change between 2 consecutive results of a given query. If there was such a change a user based command is launched.

The HTTP collector

The HTTP collector is an agent of the monitoring platform that collects web pages from web sites and returns the result of a regex operation on the returned data.

The agent knows how to perform both GET and POST requests. It was built to perform this operation very fast using complex algorithms. This enables it to collect 1000s of results from different distant sites in short cycles of around a minute.

It contains many flow control parameters that enables it to work in all kinds of networks and databases.

The HTTP Status monitoring

The HTTP collector comes with a basic feature of monitoring the returned values extracted from the web pages. It calculates if there was a change between 2 consecutive results of a given query. If there was such a change a user based command is launched.

The WMI collector

The WMI collector is an agent of the monitoring platform that collects WMI data from WMI supporting stations and returns the result on the returned data. It was built to perform this operation very fast using complex algorithms. This enables it to collect 1000s of results from different distant repositories in short cycles of around a minute. This feature requires an installation of the product, or an agent for the product on a windows platform.

The Open TCP Port collector

The Open TCP port status monitoring checks for the availability of ports in target machines. It reports about status change in the availability of the services. It was built to perform this operation very fast using complex algorithms. This enables it to collect 10000s of results from different distant repositories in short cycles of around a minute.

The Discovery Fields data collector

This collector uses the full power of the discovery tool to perform periodic data extraction with the full strength of the discovery agent.

This is a very powerful tool that can do both discovery and automation on distant machines.

It now enables among other things to use the telnet & ssh protocols as monitoring protocols.

The SNMP Traps collector

The SNMP Traps collectors filters arriving traps by multiple parameters and forwards them to required locations to be processes. This feature allows collecting traps from DMZ's without opening a UDP port in the DMZ's firewall. It can reduce the number of processed traps by defining simple filtering rules.

The CheckIT Management Platform – scales to very large networks

The CheckIT Management Platform can easily be extended to the capabilities of the Jilroy's Monitoring Platform Genie which was built for very large networks. This product was built for scale. Its agents were built in a way that enables them to be distributed, near the sites they monitor, in order to reduce traffic on the network, and to improve performance.

Most of its server components can have multiple instances too. In this way the product can handle any size of network, without losing its performance.

The CheckIT Management Platform – built for high frequency monitoring

The CheckIT Management Platform can easily be extended to the capabilities of the Jilroy's Monitoring Platform Genie which was built for high frequency monitoring. By high frequency we mean that the product can perform a monitoring cycle collecting tens of thousands of measurements in

about a minute or so.

This feature is possible due to complex algorithms that enable asynchronous data collection in parallel.

Opposed to many other products, the algorithms are not disturbed by non answering devices.

Data collection

The product has the capability of collecting the monitoring data in log files, for later processing. The product comes with logger daemons that write the collected data to CSV files.

The collection can be done remotely to the measurement processes in a way that will not disturb them.

Event generation

The user can define limits and conditions on the data collected, and the product will automatically generate alerts when a condition is met.

When the condition is met the product will launch a user defined command based on the condition which was met.

Backbone failure event storm handling

The CheckIT Management Platform was built to handle backbone failure event storms for its ICMP and SNMP status monitoring. The user can define limits for the number of events that define a backbone failure, and when those limits are reached then the events are not generated.

The product has mechanisms to automatically detect when the backbone failure was resolved, and in this case the monitoring continues normally, without losing status changes that occur during the backbone failure period.

Scheduling mechanism

The product comes with a built in scheduler that launches its local services, and its event handlers. The scheduler comes with a capability to limit the number of concurrent event handlers from a given type in order to prevent the system flooding in case of event storms.

RRD Daemon

The product has a daemon that reads from the logger's file, relevant information and creates RRD databases with the collected information. These RRD databases will contain the history information displayed later in the product's GUI as graphs.

The user will be able to configure, what will be collected as RRD data and in what format.

Extendability in the measuring

The product was designed for extendability.

The user of the product can define in simple configuration files:

- What data will be collected and from which node.
- What will be the collection frequency
- What thresholds will be defined.
- Which data will be accumulated in RRD databases
- Which reports will exist for the users

and much more.

Extendability in the GUI

The product GUI is a Java based GUI, which can run on any platform.

The GUI was built in an extend-able way that can be easily customized by the user to add more reports and graphs.

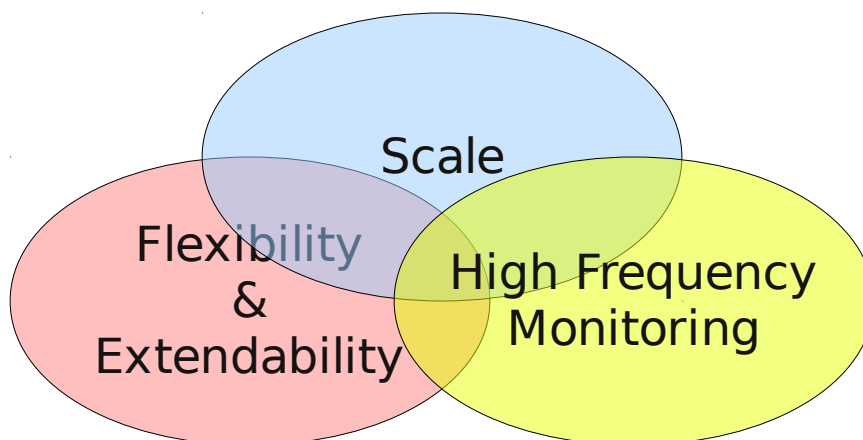
The Network Discovery component

The CheckIT Management Platform can easily be extended to the full capabilities of the Jilroy's Discovery based Inventory. Please refer to the Network Discovery attached documentation for highlights on this product.

The Network Discovery product, builds an inventory of the monitored elements. The data collected by it is accessible using the product's GUI.

Not yet another monitoring platform!

The product's differentiation points from other monitoring products are:



Flexibility & extendability

The product was designed for flexibility and extendability by its users. The users can define what will be monitored, when and when, and what will be the output for the product's users.

The users can also define thresholds that will be translated to events, that can generate traps to

Scale

The product can be extended without an effort to the Jilroy's Monitoring Platform Genie, and to scale to any size of organization, and enable monitoring large numbers of parameters from a single server.

The product can also be scaled by distributing its components over machines and locations to grow as large as needed.

Large customers, such as Telco, Service Providers and large organizations will make benefit from this capability.

High frequency monitoring

The product uses advanced and complex algorithms using asynchronous & parallel capabilities to perform high frequency monitoring, i.e monitoring the same parameters in every short time period.

management platforms, or generate actions defined by the user

A full monitoring product

The product can act as a full stand-alone monitoring platform with reports and graphs.

An extension to existing management platforms

The product can act as an extension to existing management platforms sending them traps on events discovered by it, and extracting configuration data from them.

The product can easily integrate with the common management platform (IBM Tivoli, HP OpenView, CA unicenter, BMC patrol, and others).

Additional Advantages

- The product enables its users to get near on-line monitoring information, on every parameter they would like.
- The core of the product is a full Java based solution and as such can run on any java supporting platform from Unix platforms (Linux/sun/hpux) to Windows and others.
- Its output is redirected into a database which can be accessed using jdbc
- The product comes with a network discovery component that builds its database of the monitored elements. Please refer to the Network Discovery user manual for information about this built in capability.

Jilroy Software encourages you to use the product and extend it.

If you have any questions and recommendations about how to use or how to improve the product you are most welcome to send us your thoughts to info@jilroy.com

If you need us to tailor the product to your exact needs we are offering this kind of service for reasonable fees.

Getting Started on Unix Platforms

This chapter contains information about prerequisites, installing and uninstalling the product on UNIX operating systems.

Unix System Requirements

The CheckIT Management Platform is almost a full JAVA based solution from the application logic, through the database to the graphical user interface. It comes with a JRE and an Instance of a database. So it comes equipped with anything needed for its run, and it does not conflict with any existing JREs on your machine.

Installation on UNIX

This section contains information on how to get the installation package and on how to install it on UNIX operating systems.

Getting the software

To get the installation package, you can get it from the Jilroy web site www.jilroy.com, from the download section.

If you have a download ready for your type of machine, you do not need to prepare anything.

Installing the CheckIT Management Platform

The product comes compressed in a tar.gz format.


Use the following commands to open the downloaded file.

1. **gunzip filename.tar.gz**

This will extract the compressed installation file, into the tar format.

2. **tar -xvf filename.tar**

This will extract all the product files to the /opt/jilroy directory

 By default, the CheckIT Management Platform installs itself to the /opt/jilroy directory. To Install to a different file system, use the **ln -s** command to create a soft-link from /opt/jilroy to the new destination directory.

The package installation phase creates the directory structure shown in Table 2.1

Table 2.1:Unix Directory structure

Directory	Description
/opt/jilroy	main directory
/opt/jilroy/jre	jre for current operating system
/opt/jilroy/hsqldb	hsqldb database directory
/opt/jilroy/hsqldb/database	the location of the databases created. it will be empty on installation
/opt/jilroy/hsqldb/database/checkit	the directory that will contain the database of the CheckIT program.
/opt/jilroy/checkit	the home directory for the CheckIT application.
/opt/jilroy/checkit/bin	the bin directory of the application
/opt/jilroy/checkit/conf	the configuration files directory
/opt/jilroy/checkit/conf/discoveryrules	the discovery rules configuration files location
/opt/jilroy/checkit/lib	jar files used by the program
/opt/jilroy/checkit/rrd	rrd database files location
/opt/jilroy/checkit/data	loggers and additional data files
/opt/jilroy/checkit/resource	resource files used by the GUI
/opt/jilroy/checkit/lic	license information for open source software used
/opt/jilroy/checkit/log	logs directory
/opt/jilroy/checkit/temp	temporary files dir

i The Network Discovery product which is partly included in the CheckIT Management Platform product, kept its directory structure, but the root directory has changed from 'rjnetworkdiscovery' to 'checkit'.

License

The product comes with trial licenses for you to be able to test the product. For permanent licenses please contact us at sales@jilroy.com.

The License files should be located in the /opt/jilroy/checkit/conf directory.

Execution Permission

The management platform as a whole, does not require to run under root permissions. However in many UNIX platforms you must be an administrator to use raw sockets which are used in the ICMP monitoring method.

If you want to run all services from the same machine without any change, then install the product under the root user, if this is not permitted, then add sudo permissions to the following scripts:

```
%users ALL=(ALL) NOPASSWD: /opt/jilroy/checkit/bin/icmpcollectionagent.sh
```

```
%users ALL=(ALL) NOPASSWD: /opt/jilroy/checkit/bin/discoverycycle
```

```
%users ALL=(ALL) NOPASSWD: /opt/jilroy/checkit/bin/checkit_stop
```

Post Installation scripts on UNIX

To complete the installation, a script must be executed. This script will create the database of the product and the tables needed for its operation.

Post-Installation scripts


The post-installation script performs the following actions:

1. Start the database server.
2. Create the database needed for the application.
3. Create the tables needed for the application.
4. Fills system tables with default values

The script can be found in the binaries directory:

`/opt/jilroy/checkit/bin`

to execute the script just enter its name ***./install*** from the *bin* directory and press ENTER.

 The 'install' script launches some services that should be entered into the cron tab. The services are: The hsql Database, the tomcat application server, and the scheduler. Please remember to enter them into the cron tab.

Database configuration

The default database used by the program is hsql-db which is an open source project by the Apache team. It is a full JAVA database.

The database is installed with the product.

The connection information of the database can be found in:

`/opt/jilroy/checkit/conf/config.prop`

By default the connection information will work with the supplied installation file.

i It is possible to make the program run with different database such as Oracle or PostGreSQL. The only requirement is that the database will have a JDBC driver. Look later in the book on how to change the database used by the product.

Automatic Startup

The ***install*** script starts the database. However in order to start the database after a boot to the system, add to your scheduler (crontab) the following commands:

`/opt/jilroy/checkit/bin/startdb`

This will start the database server. Currently the port on which the database is running is an internal port with the number of 54345. This can be changed by the user. Look on later chapters on how to change the database listening port if needed.

`/opt/jilroy/checkit/bin/runscheduler`

This will launch the product scheduler. It runs the commands launched by the GUI, and the event handlers.

`/opt/jilroy/checkit/bin/runtomcat`

This command launches the application server used by the GUI.

Un-installation from UNIX

The product has only presence on the `/opt/jilroy` directory.

removing all presence of the product

If you have only installed the CheckIT product, and want to remove all its presence, then remove the jilroy directory using the command: `rm -r /opt/jilroy`

This will remove all the installed parts of the product

Remove only CheckIT binaries

IF you have installed more products from jilroy Software, then they are using the jre, and the hsqldb directories. In this case you must not delete those directories.

In order to delete only the CheckIT binaries do:

```
rm -r /opt/jilroy/checkit
```

in order to delete the CheckIT database too, then do:

```
rm -r /opt/jilroy/hsqldb/database/
```

remove the crontab additions

Remember to remove the commands related to the product which you entered into the crontab. Look at the Automatic startup section for the recommended crontab commands.

GUI only Installation on Unix Platforms

This chapter contains information about the actions needed in order to install only the CheckIT GUI on a remote machine from the CheckIT engine.

Preparing the system for install

Although the product can be installed as is on any platform it is recommended that some buffer changes will be done to the operating system, so that the product will work better.

TCP buffer settings

In /etc/sysctl.conf change add the following lines:

```
net.core.rmem_max=4194304  
net.core.wmem_max=4194304
```

These parameters set the max value of the read and write buffers allowed to be allocated by the tcp/ip software.

The parameters will take effect after boot or after executing the following commands:

```
su – root  
  
sysctl -p
```

As the product is based on sending many requests concurrently and waiting for their replies as they arrive, these parameters are required so that requests & responses will not get lost and that the product will not get blocked when its buffers are full.

In the relevant conf/config files (configxxx.xml) there are parameters that allow you to set the actual value of the send snd rcv buffer sizes.

The value given in the example is 4 megabytes, however it can be increased much more for very large buffers. This buffer should be tuned if performance is not statisfing.

Unix System Requirements

The GUI is a full JAVA application which works as a stand-alone application, and connects remotely to the CheckIT Management Platform application server. It comes with a JRE so it does not conflict with any existing JREs on your machine.

Installation on UNIX

This section contains information on how to get the installation package and on how to install it on UNIX operating systems.

Getting the software

To get the installation package, you can get it from the Jilroy web site www.jilroy.com, from the download section.

If you have a download ready for your type of machine, you do not need to prepare anything.

Installing the GUI

The product comes compressed in a tar.gz format.

Use the following commands to open the downloaded file.

1. **gunzip filename.tar.gz**
This will extract the compressed installation file, into the tar format.
2. **tar -xvf filename.tar**
This will extract all the product files to the /opt/jilroy directory

i By default, the CheckIT Management Platform GUI installs itself to the /opt/jilroy directory. To Install to a different file system, use the **ln -s** command to create a soft-link from /opt/jilroy to the new destination directory.

If you have already a Network Discovery server installed. Do NOT install the GUI too. It will override some of the server files

The package installation phase creates the directory structure shown in Table 2.1

Table 2.1: Unix Directory structure

Directory	Description
/opt/jilroy	main directory
/opt/jilroy/jre	jre for current operating system
/opt/jilroy/checkit	the home directory for the CheckIT GUI application.
/opt/jilroy/checkit/bin	the bin directory of the application
/opt/jilroy/checkit/conf	the configuration files directory
/opt/jilroy/checkit/lib	jar files used by the program
/opt/jilroy/checkit/lic	license information for open source software used
/opt/jilroy/checkit/log	logs directory
/opt/jilroy/checkit/temp	temporary files dir

License

The GUI does not need a license file.

Execution Permission

The CheckIT Management Platform GUI does not require to run under root permission.

Post Installation scripts on UNIX

After the opening of the tar file, the only thing needed is to launch the GUI using the checkitgui.sh script.

By default the GUI tries to find the CheckIT Management Platform Application server on the local machine. You have to change the ServerURL configuration parameter and point to the address of the right sever. This is done from the main menu at: Administration -> Edit GUI Config File

See the chapter on the GUI for detailed explanation on how to do this.

Un-installation on UNIX

The product has only presence on the /opt/jilroy directory.

removing all presence of the product

If you have only installed the CheckIT Management Platform v.3.0 GUI, and want to remove all its

presence, then remove the jilroy directory using the command:

```
rm -r /opt/jilroy
```

This will remove all the installed parts of the product

Getting Started on Windows Platforms

This chapter contains information about prerequisites, installing and uninstalling the product on the windows operating systems.

Windows System Requirements

The CheckIT Management Platform v.3.0 is a full JAVA based solution from the application to the database. It comes with a JRE and an Instance of a hsql database. So it comes equipped with anything needed for its run, and it does not conflict with any existing JREs on your machine.

Installation on Windows

This section contains information on how to get the installation package and on how to install it on a Windows operating systems.

Getting the software

To get the installation package, you can get it from the Jilroy web site www.jilroy.com, from the download section.

If you have a download ready for your type of machine, you do not need to prepare anything.

Installing the CheckIT Management Platform v.3.0


The product comes with an executable program that contains the installation product. The product is called: JilroyCheckIT.vxy.yyymmdd.exe.

Where

Vxy – stands for version x, sub-version y.

yyymmdd - is the date of generation of the installation.

You have to run the installation executable, and the installation program will guide you through.

 The installation process asks you to select an installation path for the product.

You must not select a path that contains spaces in the directory names. If you do, there might be problems when running the product.

The package installation phase creates the directory structure shown in Table 3.1

Table 3.1:Windows Directory structure

Directory	Description
[INSTALLDIR]\jilroy	main directory
[INSTALLDIR]\jilroy\jre	jre for current operating system
[INSTALLDIR]\jilroy\hsqldb	hsqldb database directory
[INSTALLDIR]\jilroy\hsqldb\database	the location of the databases created. it will be empty on installation
[INSTALLDIR]\jilroy\hsqldb\database\checkit	the directory that will contain the database of the CheckIT program.
[INSTALLDIR]\jilroy\checkit	the home directory for the CheckIT application.
[INSTALLDIR]\jilroy\checkit\bin	the bin directory of the application
[INSTALLDIR]\jilroy\checkit\conf	the configuration files directory
[INSTALLDIR]\jilroy\checkit\conf\discoveryrules	the discovery rules configuration files location
[INSTALLDIR]\jilroy\checkit\lib	jar files used by the program
[INSTALLDIR]\jilroy\checkit\rrd	rrd database files location
[INSTALLDIR]\jilroy\checkit\data	loggers and additional data files
[INSTALLDIR]\jilroy\checkit\resource	resource files used by the GUI
[INSTALLDIR]\jilroy\checkit\lic	license information for open source software used
[INSTALLDIR]\jilroy\checkit\log	logs directory
[INSTALLDIR]\jilroy\checkit\temp	temporary files dir

The installation process


After double clicking the installation executable you will see the following screen.

This is the first screen of the install process. Press Next to continue.

At this stage the following screen will appear:

This is the license agreement document. Please read it carefully and only if you agree then type I agree to continue.

Now you are requested to select the installation path.

 You must NOT select a path that contains spaces in the directory names. If you do, there might be problems when running the product.

Now press Install to start the actual install of the product. During the installation process.

The following screen will appear.

The pop-up window instructs you to copy the license files you got from Jilroy Software to the [INSTALLDIR]\conf directory, if you have them, and if not you can still use the product with its trial license which time limited. The time limit is for 30-45 days from the download date of the installation executable.

During the install process the following command window will appear.

This happens as the install process does a few actions during the install process. The actions are:

1. Install 4 services that are used by the product:
 - JilroyDbServer: This is a service that runs the database used by the product. The database is a known open-source java database called hsql.
 - JilroyScheduler: This is a service that runs jobs scheduled by the user or by the application it self.
 - JilroyWatchDog: This is a service that makes sure that all the services and agents of the monitoring platform are running well. In case of an error it sends snmp traps and mail messages about the status.
 - JilroyTomcat: This is the application server used to serve the GUI clients of the application.
2. Initiate the database scheme and create the tables used by the product
3. Register the application GUI in the computer's start-up programs.

At the end of the install process the following screen appears:

Press Finish to end the installation.

 You can look at the install script in the [INSTALLDIR]\bin\install.bat.

License

The product comes with trial licenses for you to be able to test the product. They usually last for 30-45 days from the download time of the install program. For permanent licenses please contact us at sales@jilroy.com.

The License files should be located in the [INSTALLDIR]\checkit\conf directory.

Database configuration


The default database used by the program is hsql-db which is an open source project by the Apache team. It is a full JAVA database.

The database is installed with the product.

The connection information of the database can be found in:

[INSTALLDIR]\checkit\conf\config.prop

By default the connection information will work with the supplied installation file.

 It is possible to make the program run with different database such as Oracle or PostgreSQL. The only requirement is that the database will have a JDBC driver. Look later in the book on how to change the database used by the product.

Automatic Startup

The **install** script defines the services used by the product. Make sure not to change that they will start

automatically when the computer starts.

Un-installation on Windows

The product has multiple methods for un-installing it. The best way is to use the Add/Remove programs from the control panel.

The following screens will appear.

Press Next to continue.

The following screen will appear. It tells you what you are about to do:

Press the Uninstall button.

During the un-install process the following screen will appear:

The un-install will remove the defined services and the defined launch commands.

In the end of the un-install program the following screen will appear:

Press Finish to end the install process.

i The product keeps a backup of its conf directory in the %TEMP%/jilroy. This is done as the user may add functionality to the product that might be important to him.

GUI only Installation on Windows Platforms

This chapter contains information about the actions needed in order to install only the CheckIT Management Platform GUI on a remote machine from the CheckIT Management Platform engine.

Windows System Requirements

The GUI is a full JAVA application which works as a stand-alone application, and connects remotely to the CheckIT Management Platform application server. It comes with a JRE so it does not conflict with any existing JREs on your machine.

Installation on Windows

This section contains information on how to get the installation package and on how to install it on a Windows operating systems.

Getting the software

To get the installation package, you can get it from the Jilroy web site www.jilroy.com, from the download section.

If you have a download ready for your type of machine, you do not need to prepare anything.

Installing the CheckIT Management Platform v.3.0 GUI

The product comes with an executable program that contains the installation product. The product is called: JilroyCheckITGUI.vxy.yyymmdd.exe.

Where

Vxy – stands for version x, sub-version y.

yyymmdd - is the date of generation of the installation.

You have to run the installation executable, and the installation program will guide you through.

i The installation process asks you to select an installation path for the product.

You must not select a path that contains spaces in the directory names. If you do, there might be problems when running the product.

The package installation phase creates the directory structure shown in Table 3.1

Table 3.1:Windows Directory structure

Directory	Description
[INSTALLDIR]\jilroy	main directory
[INSTALLDIR]\jilroy\jre	jre for current operating system
[INSTALLDIR]\jilroy\checkit	the home directory for the CheckIT application.
[INSTALLDIR]\jilroy\checkit\bin	the bin directory of the application
[INSTALLDIR]\jilroy\checkit\conf	the configuration files directory
[INSTALLDIR]\jilroy\checkit\lib	jar files used by the program
[INSTALLDIR]\jilroy\checkit\data	loggers and additional data files
[INSTALLDIR]\jilroy\checkit\resource	resource files used by the GUI
[INSTALLDIR]\jilroy\checkit\lic	license information for open source software used
[INSTALLDIR]\jilroy\checkit\log	logs directory
[INSTALLDIR]\jilroy\checkit\temp	temporary files dir

The installation process

After double clicking the installation executable you will see the following screen.

This is the first screen of the install process. Press Next to continue. At this stage the following screen will appear:

This is the license agreement document. Please read it carefully and only if you agree then type I agree to continue.

Now you are requested to select the installation path.

❗ You must NOT select a path that contains spaces in the directory names. If you do, there might be problems when running the product.

Now press Install to start the actual install of the product. During the installation process.

During the installation process, the GUI files are copied and there is a registration of the application GUI in the computer's start-up programs.

At the end of the install process the following screen appears:

Press Finish to end the installation.

 You can look at the install script in the [INSTALLDIR]\bin\install.bat.

License

No license is needed for the GUI installation, so you may install any number of clients as needed. However the product itself requires a license.

Un-installation on Windows

The product has multiple methods for un-installing it. The best way is to use the Add/Remove programs from the control panel.

The following screens will appear.

Press Next to continue.

The following screen will appear. It tells you what you are about to do:

Press the Uninstall button.

The un-install will remove the program files and the defined launch commands.

In the end of the un-install program the following screen will appear:

Press Finish to end the install process.

Configuration

In this chapter we will explain how to configure CheckIT from the most common parts to the lesser common ones.

i In this chapter you will not find explanation on the configuration of general components (like the scheduler). The information about them will be found in their specific chapter.

i The documentation will not include the files and parameters that are specific to the Network Discovery product. You can find a detailed explanation on them in the Network Discovery users guide.

Config.prop

[General]

Verbose=FINEST

ServerURL=http://127.0.0.1:54346/

Not_up_status_cycle_timeout_in_sec=600

[Database]

DriverName=org.hsqldb.jdbcDriver

Manager=jdbc:hsqldb:hsq://127.0.0.1:54345/checkit

User=jilroy

Password=jilroy

[snmp_communities]

Community1=public0

Community2=public1

Community3=public2

Community4=public3

Community5=

Community6=

[Discovery]

launch_freq=24:00:00

rediscovery_timeout=14:00:00:00

[DiscoveryJobs]

Add_Node_job=[INSTALLDIR]bin/printline.sh [INSTALLDIR]/log/printline.log

Delete_Node_job=[INSTALLDIR]bin/printline.sh [INSTALLDIR]/log/printline.log

[Default_monitor_parameters]

Timeout_in_sec=3

Retry_count=3

Monitoring_cycle_timeout_in_sec=60

Monitoring_count_in_cycle=1

Inter_cycle_monitor_delay=1

Send_delay_count=10

Max_concurrent_requests=5000

Only_send_change_status_information=no

Start_monitoring_time=NOLIMIT

Stop_monitoring_time=NOLIMIT

Dayly_Status_Change_count_Limit=9

Backbone_failure_operational=0

[ICMP]

Change_status_job=/opt/jilroy/checkit/bin/printline.sh /opt/jilroy/checkit/log/printline.log

;trap_up_job=snmptrap parameters where the ip address is appended at the end of the string

;trap_down_job=snmptrap parameters where the ip address is appended at the end of the string

Event_job=/opt/jilroy/checkit/bin/printline.sh /opt/jilroy/checkit/log/printline.log

Status_Change_Limit=/opt/jilroy/checkit/bin/printline.sh /opt/jilroy/checkit/log/printline.log

[ICMP_BACKBONE_FAILURE]

Start_service_timeout_sec=600

Time_window_size_sec=180

Minor_threshold=1000

Major_threshold=1500

Recovery_count=800

Backbone_failure_change_status_command=/opt/jilroy/checkit/bin/printline.sh
/opt/jilroy/checkit/log/printline.log

[SNMP]

Change_status_job=/opt/jilroy/checkit/bin/printline.sh /opt/jilroy/checkit/log/printline.log

Event_job=/opt/jilroy/checkit/bin/printline.sh /opt/jilroy/checkit/log/printline.log

```
Status_Change_Limit=/opt/jilroy/checkit/bin/printline.sh /opt/jilroy/checkit/log/printline.log
```

[SNMP_BACKBONE_FAILURE]

```
Start_service_timeout_sec=600
```

```
Time_window_size_sec=180
```

```
Minor_threshold=1000
```

```
Major_threshold=1500
```

```
Recovery_count=800
```

```
Backbone_failure_change_status_command=/opt/jilroy/checkit/bin/printline.sh  
/opt/jilroy/checkit/log/printline.log
```

[WatchDog]

```
Change_status_command=/opt/jilroy/checkit/bin/printline.sh /opt/jilroy/checkit/log/printline.log
```

/conf/Config.prop is the user configuration file, for the batch commands of the discovery process, the DataCollectorServer and some internal commands

General

This section defines general parameters regarding the operation of the software.

Currently the parameters that can be set are:

Verbose

Defines the detail level of the logs produced by the software. valid values are: severe , info, fine, finer and finest.

i If you want to change the Verbose value, you have to restart the relevant processes after the change as they read this value only on the 'init' phase. used by the product.

ServerURL

This parameter specifies the location of the Application server of the product. It is used when working in a distributed environment to distribute changes in Jar data to the remote processes.

Not_up_status_cycle_timeout_in_sec

The product collects how many not up interfaces exist in a given time period. There is an RRD report on this information. This parameter defines the frequency of data collection for this parameter.

Database

This section defines the connection string info for the discovery database. Out of the box, the system comes with the Apache hsql Database, running in its server mode, on port 54345.

Currently the parameters that can be set are:

DriverName

The Class name for the database client.

Manager

The JDBC connection string of the database

User

The User Name used to access the database

Password

The Password value used to access the database

You can alter all these parameters to fit your environment.

i Note that currently the ./startdb and ./stopdb scripts are pointing to the hsql Db scripts. You will have to fix them too so they will fit your database.

Default_monitor_parameters

This section contains default parameters for the monitoring agents. They can be overridden by values put in the specific monitor config.prop file, see later a description on those files.

Timeout_in_sec

The timeout value for a request.

Retry_count

The number of retry operations if a timeout occurred.

Monitoring_count_in_cycle

This parameter is used by the ICMP monitor, to determine how many ICMP requests will be sent in a given monitoring cycle. If the number is larger than one, then the product can be used to measure Jitter on the line to the target. See a later chapter about this option.

Inter_cycle_monitor_delay

In the ICMP monitor (Jitter mode), this parameter determines how long to wait in a given cycle between each ICMP request.

Send_delay_count

This is a flow control parameter that determines how many requests will be sent before a small delay is entered. It is used to prevent over flooding the network card with too many requests.

Max_concurrent_requests

This is a flow control parameter that determines how many requests will be active concurrently. It defines a window size for concurrent requests.

Only_send_change_status_information

In the ICMP monitor, this parameter defines if to send request statistics (i.e reply time) or only status

change messages.

Start_monitoring_time

This parameter is used to define the start monitoring time in the format of HH:MM:SS or NOLIMIT which says all day.

Stop_monitoring_time

This parameter is used to define the stop monitoring time in the format of HH:MM:SS or NOLIMIT which says no limit.

Dayly_Status_Change_count_Limit

This parameter sets a limit to the times an interface may change its status in a single day. Above this limit an event handler is launched.

Backbone_failure_operational

This parameter specifies if the backbone failure feature is activated (1) or not (0).

ICMP

This section defines parameters that are directly related to the ICMP monitor, and effect the data collector server.

Change_status_job

The job that will be launched on each ICMP interface change status.

trap_up_job

A job that is used for sending traps to the management platform notifying an IP address is up. Its hard coded parameters should be the parameters of the snmptrap command documented in the utilities section. The IP address string will be appended to the command string.

Note: the snmptrap is a job that is run internally without forking by the scheduler.

trap_down_job

A job that is used for sending traps to the management platform notifying an IP address is down. Its

hard coded parameters should be the parameters of the snmptrap command documented in the utilities section. The IP address string will be appended to the command string.

Note: the snmptrap is a job that is run internally without forking by the scheduler.

Event_job

The job that will be launched on every event that occurs on data relating the ICMP monitor. see defining events in later chapters.

Status_Change_Limit

The job that will be launched on when the limit of status changes for a given interface is reached.

ICMP_Backbone_failure

This section defines the parameters that define the backbone failure conditions for the ICMP monitoring method

Start_service_timeout_sec

A timeout from starting an agent that monitors with this method, that within it, no backbone failure events will be detected. The logic behind this parameter is that when a monitoring agent starts, it discovers a lot of changes that occurred since its last round. We might not want to consider these changes as backbone failure.

Time_window_size_sec

A Backbone failure is discovered when a large number of interfaces go down, in a small period of time. The amount of time in which the count of interfaces which go down is calculated, is specified by this parameter.

Minor_threshold

When this value of failing interfaces is reached within the given time window, we get a notification in the 'change status' job that a Possible backbone failure situation is occurring.

Major_threshold

When this value of failing interfaces is reached within the given time window, we stop getting notifications about failing interfaces. We will get a single message that we entered a backbone failure situation. We keep an indication on every interface that went down during the backbone failure

situation.

Recovery_count

If we are in a backbone failure situation, then only when the number of interfaces that go down in the time window size is below this value, we get out of the backbone failure state.

Backbone_failure_change_status_command

This command is launched when we enter a backbone failure or we leave it.

SNMP

This section defines parameters that are directly related to the SNMP monitor, and effect the data collector server.

Change_status_job

The job that will be launched on each SNMP interface change status.

Event_job

The job that will be launched on every event that occurs on data relating the SNMP monitor. see defining events in later chapters.

Status_Change_Limit

The job that will be launched on when the limit of status changes for a given interface is reached.

SNMP_Backbone_failure

This section defines the parameters that define the backbone failure conditions for the SNMP monitoring method. Please refer to the ICMP_Backbone_failure section (above) for a detailed description about the section's parameters.

WatchDog

This section defines parameters that are directly related to the WatchDog process.

Change_Status_command

The job that will be launched when a service changes its status

config.xml

The conf/config.xml file is an internal configuration file, that would hardly have to be processed by the user. However this is the main configuration file passed as a parameter to many batch parameters such as the data collector server.

configGUI.prop

```
[General]

Verbose=FINEST

ServerURL=http://127.0.0.1:54346/
```

//conf/ConfigGUI.prop is the user configuration file, for the CheckIT GUI. It contains the the following sections:

General

This section defines general parameters regarding the operation of the software.

Currently the parameters that can be set are:

Verbose

Defines the detail level of the logs produced by the software. valid values are: severe , info, fine, finer and finest.

i If you want to change the Verbose value, you have to restart the relevant processes after the change as they read this value only on the 'init' phase. used by the product.

ServerURL

This parameter points to the address of the Application server of the product. This value should not be changed if the GUI is launched on the server's machine.

configGUI.xml

The conf/configGUI.xml file is an internal configuration file, that would hardly have to be processed by the user. However this is the main configuration file passed as a parameter to the GUI

configSNMPDataCollectorAgent.prop

This file defines parameters for the SNMP data collector agent.

i The parameters that will be explained in this sections are those that do not appear in the config.prop section.

```
[General]

Verbose=FINEST

ServerURL=http://127.0.0.1:54346/


[DataCollection]

Push_data_timeout_in_sec=15


[SNMP]

Timeout_in_sec=3

Retry_count=3
```

```
Monitoring_cycle_timeout_in_sec=300  
Monitoring_count_in_cycle=1  
Inter_cycle_monitor_delay=1  
Send_delay_count=10  
Max_concurrent_requests=5000  
Only_send_change_status_information=no  
Start_monitoring_time=NOLIMIT  
Stop_monitoring_time=NOLIMIT  
Max_concurrent_requests_for_node=1  
Max_OID_count_in_request=5
```

DataCollection

This section defines parameters that relate to the communication with the data collection server.

Push_data_timeout_in_sec

The parameter defines the timeout value after it a push is done to messages intended to the data collection server. This option enables buffering of data before sending it to the server.

SNMP

This section defines parameters that effect the operation of the snmp data collector agent. Most of the parameters were explained in the section of the config.prop, however there are some new parameters specific to the SNMP data collector.

Max_concurrent_requests_for_node

This parameter defines how many requests can be sent concurrently to a single node. It was found that some nodes do not support concurrent requests.

Max_OID_count_in_request


This parameter defines how many OIDs will be put in a single SNMP request. It was found that some devices have a limited buffer to receive SNMP requests. A large number of requests increases the size of the request, but reduces the line utilization.

configSNMPDataCollectorAgent.xml

The conf/configSNMPDataCollectorAgent.xml file is an internal configuration file, that would hardly have to be processed by the user. However this is the main configuration file passed as a parameter to the SNMP Data Collector Agent.

configICMPDataCollectorAgent.prop

This file defines parameters for the ICMP data collector agent.

 All the parameters in this file were explained either in the config.prop section or in the configSNMPDataCollectorAgent.prop file.

[General]

Verbose=FINEST

ServerURL=http://127.0.0.1:54346/

[DataCollection]

Push_data_timeout_in_sec=60

[ICMP]

Send_delay_count=10

```
Max_concurrent_requests=5000


Only_send_change_status_information=no
```

configICMPDataCollectorAgent.xml

The conf/configICMPDataCollectorAgent.xml file is an internal configuration file, that would hardly have to be processed by the user. However this is the main configuration file passed as a parameter to the ICMP Data Collector Agent.

configSQLDataCollectorAgent.prop

This file defines parameters for the SQL data collector agent.

 All the parameters in this file were explained either in the config.prog section or in the configSNMPDataCollectorAgent.prop file.

```
[General]

Verbose=FINEST

ServerURL=http://127.0.0.1:54346/


[DataCollection]

Push_data_timeout_in_sec=15


[SQL]

Max_concurrent_requests=100
```


configSQLDataCollectorAgent.xml

The conf/configSQLDataCollectorAgent.xml file is an internal configuration file, that would hardly have to be processed by the user. However this is the main configuration file passed as a parameter to the

SQL Data Collector Agent.

configSNMPTrapDataCollectorAgent.prop

This file defines parameters for the SNMP traps data collector agent.

 All the parameters in this file were explained either in the config.prog section or in the configSNMPDataCollectorAgent.prop file.

```
[General]

Verbose=FINEST

ServerURL=http://127.0.0.1:54346/


[DataCollection]

Push_data_timeout_in_sec=15


[SNMP]

SnmptTrapPort=162

ReceiveBufferSize=4000000
```

SnmptTrapPort

The port the agent will bind to, and expect to get traps in.

ReceiverBufferSize

The size of the receive buffer in the UDP socket, on which we get the traps.


configSNMPTrapDataCollectorAgent.xml

The conf/configSNMPTrapDataCollectorAgent.xml file is an internal configuration file, that would

hardly have to be processed by the user. However this is the main configuration file passed as a parameter to the SQL Data Collector Agent.

configCSVDataCollectorAgent.prop

This file defines parameters for the CSV data collector agent.

 All the parameters in this file were explained either in the config.prog section or in the configSNMPDataCollectorAgent.prop file.

```
[General]

Verbose=FINEST

ServerURL=http://127.0.0.1:54346/


[DataCollection]

Push_data_timeout_in_sec=15


[CSV]

Max_concurrent_requests=100
```

configCSVDataCollectorAgent.xml

The conf/configCSVDataCollectorAgent.xml file is an internal configuration file, that would hardly have to be processed by the user. However this is the main configuration file passed as a parameter to the CSV Data Collector Agent.

AgentsList.txt

Please refer to the agents technology chapter about explanation of this file.

This file is currently configured for a single server implementation of the product.

The SQL Filtering Files

The method to select which nodes will be monitored using a given monitoring agent is determined by a SQL filtering file specific for the given agent, and pointed by the agent's entry in the config.xml file of the agent's data collector.

Every agent has a specific format for its filtering instructions that determines who, what and when will be monitored, and what will be done with the monitoring results.

In the following sections we will list the format of some of the filtering files. Samples of all the files are found in the [installdir]/conf/ of the server.

Each file contains documentation on the parameters that should be passed, and a sample on how to use the file.

Common features

- All the filtering commands can have multiple SQL within them that are processed sequentially.
- Each selected line represents a single monitored node
- For each selected monitored node, you can specify when it should be monitored and at what frequently.

Additional commands

There are a few special commands that can be added to the filtering file. These commands are used to prepare for the SQL commands

\$LoadCsvToDatabase

This command loads a CSV file to a table in the database, that can later be queried.

The format is:

`$LoadCsvToDatabase:filename,tbl_name, Separator, is first line fields name (0|1)`

where the parameters are:

- the full path to the CSV file
- The table name in the database that will contain the content of the CSV file. This file is created each time the file is loaded.

- The separator between fields in the CSV file. If the separator is a comma, then write the string 'comma' instead of the value
- A flag indicating if the first line in the CSV file contains the field names. If yes then put the value -1, otherwise put 0. You can use the field names in the file as selection fields in the SQL command reading from the given table. If no field names are supplied then the fields in the table are labeled: F1, F2,... Fn

\$ DeleteTable

This command deletes a table from the database.

The format is:

`$DeleteTable:tbl_name`

where the parameters are:

- The table name in the database that needed to be deleted

ICMPSQLFilteringFile.txt

This file acts as a filtering file on top of the Network Discovery discovered network. It determines which interfaces will be monitored using the ICMP Data Collection Agent.

```
# This file contains sql commands for selecting the addresses to be ICMP pinged from the
# discovery process.
# the commands are seperated by ;
#
# default values will be values not selected (if they are in the end of the selection, or
# fields containing the string '(null)'
#
# The parameters selected should be:
```

```

#      1. nObjectId of strType = Address to monitor
#      2. strIpAddr to monitor
#      3. strHostName of address
#      4. Monitoring Frequency in seconds (optional, otherwise default taken from config.xml)
#      5. Monitoring Start Time HH:MM(optional, otherwise default taken from config.xml)
#      6. Monitoring Stop Time HH:MM(optional, otherwise default taken from config.xml)
#      7. Monitoring Timeout (optional, otherwise default taken from config.xml)
#      8. Monitoring Retry count (optional, otherwise default taken from config.xml)
#      9. Condition group to use on this node (optional, otherwise default taken from config.xml)
#      10. Max Dayly Status Change Count before an event on it
#
# ICMP specific parameters
#
#      11. Monitoring Count In Cycle (optional, otherwise default taken from config.xml)
#      12. Monitoring In Cycle Delay Time in sec (optional, otherwise default taken from config.xml)
select  nObjectId,strIpAddr,strDNSName  from  tbObject  where  strType  =  'Address'  and
strMonitoringMethod = 'ICMP';

```

The selection is done using one or more SQL commands that return the selected entries in the tbObject (which contains the outout of the discovery process).

The returned records must be in a fixed format, as described in the file itself.

SNMPSQLFilteringFile.txt

This file acts as a filtering file on top of the Network Discovery discovered network. It determines which interfaces will be monitored using the SNMP Data Collection Agent.

```

# This file contains sql commands for selecting the addresses and the OIDs to be collected using

```

```
SNNMP from the
# discovery process.
# the commands are seperated by ;
#
# default values will be values not selected (if they are in the end of the selection, or
# fields containing the string '(null)'
#
#
# The parameters selected should be:
#     1. nObjectId of strType = Node of the Node to monitor
#     2. strIpAddr of Node to monitor
#     3. strHostName of Node
#     4. Monitoring Frequency in seconds (optional, otherwise default taken from config.xml)
#     5. Monitoring Start Time HH:MM(optional, otherwise default taken from config.xml)
#     6. Monitoring Stop Time HH:MM(optional, otherwise default taken from config.xml)
#     7. Monitoring Timeout (optional, otherwise default taken from config.xml)
#     8. Monitoring Retry count (optional, otherwise default taken from config.xml)
#     9. Condition group to use on this node (optional, otherwise default taken from config.xml, or
NONE)
#         This condition group will work on all OIDs on the same node.
#     10. Max Dayly Status Change Count before an event on it
#
# snmp specific parameters
#     11. SNMP Version
```

```
# 12 SNMP Community/User
# 13. SNMP Password (used by SNMPV3 can be split to auth_path#priv_path)
# 14. OID Nic-Name
# 15. OID Type (Field|Table)
# 16. OID Value - if Field then the full value
# 17. OID_Key - value which is appended to OID if there is one, for example ifIndex
```

```
SELECT
```

```
    Interface.nObjectId,
    Node.strIpAddr,
    Node.strDNSName,
    '(null)' ,
    '(null)' ,
    '(null)',
    '(null)',
    '(null)',
    '(null)',
    '(null)',
    Node.nSNMPVersion,
    Node.strCommunity,
    Node.strPassword,
    'JILROY_IF_STATUS',
    'Field',
    '1.3.6.1.2.1.2.2.1.7.#1.3.6.1.2.1.2.2.1.8.'
```

```
        Interface.nIfIndex,
        Interface.strDescription
FROM
        tblObject Node,
        tblObject Interface
WHERE
        Node.strType = 'Node' AND
        Interface.strType = 'Interface' AND
        Node.nObjectId = Interface.nParentObjectId AND
        Node.strMonitoringMethod = 'SNMP'
```

The selection is done using one or more SQL commands that return the selected entries in the tblObject (which contains the output of the discovery process).

The returned records must be in a fixed format, as described in the file itself.

Note that if you do not want to enter values that can be taken from defaults in the server, you have to enter the value of '(null)'.

SQLCommandsSQLFilteringFile.txt

This file acts as a filtering file on top of the Network Discovery discovered network. It determines which databases will be monitored using the SQL Data Collection Agent.

```
# This file contains sql commands for selecting the nodes that we want to monitor using the SQL
JDBC from

# the discovery process output tables.

# the commands are separated by ;

#
```

```
# default values will be values not selected (if they are in the end of the selection, or
# fields containing the string '(null)'
#
#
# The parameters selected should be:
#     1. nObjectId
#     2. strIpAddr of strType = Node of the Node to monitor
#     3. strHostName of Node
#     4. JDBC Driver name
#     5. Connection string (IP and port can represented by @IP_ADDRESS@ @PORT@)
#     6. User name
#     7. password
#     8. Indicate what we want to do with data. OnlyMonitor|OnlyLog|MonitorAndLog
#     9. Database Port number
#     10. Monitoring Frequency in seconds (optional, otherwise default taken from config.xml)
#     11. Monitoring Start Time HH:MM(optional, otherwise default taken from config.xml)
#     12. Monitoring Stop Time HH:MM(optional, otherwise default taken from config.xml)
#     13. Monitoring Timeout (optional, otherwise default taken from config.xml)
#     14. Monitoring Retry count (optional, otherwise default taken from config.xml)
#     15. Condition group to use on this node (optional, otherwise default taken from config.xml, or
NONE)
#     16. Max Dayly Status Change Count before an event on it(optional for oper+admin status)
#     17. SQL Command unique NickName
#     18. SQL Command to execute
```

```
SELECT
```

```
    Node.nObjectId,  
    Node.strIpAddr,  
    Node.strDNSName,  
    'org.apache.derby.jdbc.ClientDriver' ,  
    'jdbc:derby://@IP_ADDR@:@PORT@/checkit' ,  
    'Jilroy',  
    'jilroy',  
    'OnlyLog',  
    54345,  
    '(null)',  
    '(null)',  
    '(null)',  
    '(null)',  
    '(null)',  
    '(null)',  
    '(null)',  
    'Counters',  
    'select count(*) from tblObject'
```

```
FROM
```

```
    tblObject Node
```

```
WHERE
```

```
    Node.strType = 'Node' AND
```

```
Node.strIpAddr = '127.0.0.1'
```

The selection is done using one or more SQL commands that return selected entries from tables found in the pointed database. For example in this case we use selected entries in the tblObject (which contains the output of the discovery process).

The returned records must be in a fixed format, as described in the file itself.

Note that if you do not want to enter values that can be taken from defaults in the server, you have to enter the value of '(null)'.

CSVCommandsSQLFilteringFile.txt

This file acts as a filtering file on top of the Network Discovery discovered network. It determines which CSV files will be monitored using the CSV Data Collection Agent.

```
# This file contains sql commands for selecting the CSV files that we want to monitor using the CSV
agent

# and selects the data in them. The selection is done using SQL commands when it is assumed that

# the CSV is loaded into a table named #TEMP_TABLE# and can be queried

#

# You can write multiple commands to select the files.

# the commands are separated by ;

#

# default values will be values not selected (if they are in the end of the selection, or

# fields containing the string '(null)'

#

#

# The parameters selected should be:

#     1. nObjectId

#     2. strIpAddr or strType = Node of the Node to monitor
```

```
# 3. strHostName of Node
# 4. URL of file locatoin (IP and port can represented by @IP_ADDRESS@ @PORT@)
# 5. Is First Line a Header Line (0|1)
# 6. User name (if URL auth is needed)
# 7. password (if URL auth is needed)
# 8. URL port
# 9. CSV Seperator
# 10. Indicate what we want to do with data. OnlyMonitor|OnlyLog|MonitorAndLog
# 11. Monitoring Frequency in seconds (optional, otherwise default taken from config.xml)
# 12. Monitoring Start Time HH:MM(optional, otherwise default taken from config.xml)
# 13. Monitoring Stop Time HH:MM(optional, otherwise default taken from config.xml)
# 14. Monitoring Timeout (optional, otherwise default taken from config.xml)
# 15. Monitoring Retry count (optional, otherwise default taken from config.xml)
# 16. Condition group to use on this node (optional, otherwise default taken from config.xml, or NONE)
# 17. Max Dayly Status Change Count before an event on it(optional for oper+admin status)
# 18. CSV Command unique NickName
# 19. SQL Command to execute

SELECT
    Node.nObjectId,
    Node.strIpAddr,
    Node.strDNSName,
    'file:///opt/jilroy/checkit/data/DNSName.csv',
    '1',
```

```
'(null)',
'(null)',
80,
'',
'OnlyLog',
'(null)',
'(null)',
'(null)',
'(null)',
'(null)',
'(null)',
'(null)',
'(null)',
'CSVTest',
'select count(*) from #TEMPTABLE#'
FROM
tblObject Node
WHERE
Node.strType = 'Node' AND
Node.strIpAddr = '127.0.0.1'
```

The selection is done using one or more SQL commands that return selected entries from tables found in the pointed database. For example in this case we use selected entries in the tblObject (which contains the outout of the discovery process).

The returned records must be in a fixed format, as described in the file itself.

Note that if you do not want to enter values that can be taken from defaults in the server, you have to

enter the value of '(null)'.

HTTPCommandsSQLFilteringFile.txt

This file acts as a filtering file on top of the Network Discovery discovered network. It determines which Web sites will be monitored using the HTTP Data Collection Agent.

```
# This file contains sql commands for selecting the web URLs that we want to monitor using the HTTP
agent

# and selects the data in them. The selection is done using SQL commands when it is assumed that

# the Web pages returned are scanned using a regular expression

#

# You can write multiple commands to select the files.

# the commands are seperated by ;

#

# default values will be values not selected (if they are in the end of the selection, or

# fields containing the string '(null)'

#

#

# The parameters selected should be:

#     1. nObjectId

#     2. strIpAddr of strType = Node of the Node to monitor

#     3. strHostName of Node

#     4. URL of page locatoin (IP and port can represented by @IP_ADDRESS@ @PORT@)

#     5. User name (if URL auth is needed)

#     6. password (if URL auth is needed)
```

```
# 7. URL port
# 8. Post | Get (indicates what type of http request will be done)
# 9. Post/Get data
# 10. Indicate what we want to do with data. OnlyMonitor|OnlyLog|MonitorAndLog
# 11. Monitoring Frequency in seconds (optional, otherwise default taken from config.xml)
# 12. Monitoring Start Time HH:MM(optional, otherwise default taken from config.xml)
# 13. Monitoring Stop Time HH:MM(optional, otherwise default taken from config.xml)
# 14. Monitoring Timeout (optional, otherwise default taken from config.xml)
# 15. Monitoring Retry count (optional, otherwise default taken from config.xml)
# 16. Condition group to use on this node (optional, otherwise default taken from config.xml, or NONE)
# 17. Max Dayly Status Change Count before an event on it(optional for oper+admin status)
# 18. HTML Command unique NickName
# 19. Parsing Regular expression

SELECT
    Node.nObjectId,
    Node.strIpAddr,
    Node.strDNSName,
    'http://www.jilroy.com/home.html',
    '(null)',
    '(null)',
    80,
    'GET',
    '(null)',
```

```
'OnlyLog',
'(null)',
'(null)',
'(null)',
'(null)',
'(null)',
'(null)',
'(null)',
'(null)',
'HTMLTest',
"
FROM
    tblObject Node
WHERE
    Node.strType = 'Node' AND
    Node.strIpAddr = '127.0.0.1'
```

The selection is done using one or more SQL commands that return selected entries from tables found in the pointed database. For example in this case we use selected entries in the tblObject (which contains the outout of the discovery process).

The returned records must be in a fixed format, as described in the file itself.

Note that if you do not want to enter values that can be taken from defaults in the server, you have to enter the value of '(null)'.

SNMPTrapsSQLFilteringFile.txt

This file acts as a filtering file for traps sent to the agent's location.

```
# This file contains sql commands for selecting the traps that we want to process using the Snmp
Trap agent

#

# the commands are seperated by ;

#

# default values will be values not selected (if they are in the end of the selection, or
# fields containing the string '(null)'

#

# In the mask fields:

#           In source Ip Address you can put mask within the address. For example 10.10.0.*
#           In other fields * means all values

#

# The parameters selected should be:

#   1. strSourceIpAddress - the source ip of the trap
#   2. strCommunity - the community in the trap
#   3. strEnterpriseld - enterprise id of trap
#   4. strGenericId - generic id of the trap
#   5. strSpecificId - specific id of the trap
#   6. strOld - OID found in the trap
#   7. strCondition - LT | LE | EQ | GE | GT
#   8. strValue: value of OID
#   9. strAction: Drop | Forward:IpAddressToForwardTo | DoAction
#  10. Monitoring Start Time HH:MM(optional, otherwise default taken from config.xml)
```

```
#      11. Monitoring Stop Time HH:MM(optional, otherwise default taken from config.xml)

#      12. Condition group to use on this node (optional, otherwise default taken from config.xml, or
NONE)

#

#

SELECT DISTINCT

    '1',

    'public',

    '1',

    '1',

    '1',

    '1',

    '1',

    '1',

    'Jilroy_ForwardTrap:127.0.0.1:162',

    '(null)',

    '(null)',

    '(null)'

FROM tblUniqueCounter
```

The selection is done using one or more SQL commands that return selected entries from tables found in the pointed database. In the given example we select constants, and forward all the received traps with a community = 'public' to a trap server located on the data collector machine.

The returned records must be in a fixed format, as described in the file itself.

Note that if you do not want to enter values that can be taken from defaults in the server, you have to enter the value of '(null)'.

The Agents Technology

The CheckIT product is built on top of a client-server technology that enables its components to communicate with one another through a secured and encrypted protocol.

This technology simplifies the scaling of the product and enables the existence of multiple instances of the same collectors, which can be distributed in the organization.

The components that use this technology are:

- The data collection server
- The ICMP data collecting agent
- The SNMP data collecting agent
- The SQL data collecting agent
- The logger daemon
- The RRD daemon
- The Scheduler
- The WatchDog process

and more...

Types of processes from the communication point of view

Using this technology each process can act either as:

listener

A process that acts as an acceptor, is able to accept connection requests from defined processes

Connector

A process that acts as an connector, is initiating a connect requests to pre-defined processes that are listeners

Both

A process can have both those capabilities enabled.

The service capabilities are defined in its conf/config.xml file

The reason for this flexibility is to support different security concepts when implementing the product.

 Please contact jilroy representatives if the set defaults do not suite you.

AgentsList.txt

This file defines the targets that the process should connect or accept connections from.

```
# Agent list file for the server to connect to

# format: IP, port, flag that tells what type of agent it is(Listener = true, Connector = false)

# data collector agent

127.0.0.1,54347,listener,/home/raphyz/projects/RJServerAgent/dist/conf/enckey.txt

# watchdog agent

127.0.0.1,54344,listener,/home/raphyz/projects/RJServerAgent/dist/conf/enckey.txt

127.0.0.1,-1,Connector,/home/raphyz/projects/RJServerAgent/dist/conf/enckey.txt
```

The file is built of lines which represent partners that the process should communicate with.

The partner line format

The line is CSV based with these parameters:

Field Type	Description
Ip Address	Destination IP address
Port number	Destination port. when our process is a listener, then the port is usually unknown and in this case we use -1, to allow any port
process type	The type of the target process (listener,Connector). This will determine if we connect to it or we wait for his connection.
Master Encryption Key	The communication between the processes is encrypted using a key that is generated at the beginning of the session. This key is used (not directly) to pass the session key securely. The encryption algorithm is AES (256)

The ICMP Data Collector agent logic

The ICMP data collector agent is one type of data collector supported by the CheckIT. It sends ICMP echo requests and collects ICMP echo replies from IP based devices.

It is a light weight agent that does not require any additional software locally (It does not require any database for its operations). It communicates with its data collection server, using a secured protocol on top of the tcp protocol.

The collector functionality

The collector gets a list of monitored elements which are a list of IP addresses. Based on operational parameters it sends the ICMP requests to those addresses using an asynchronous algorithm and in parallel.

It does not wait for a reply from a given interface to send the request to another interface. It sends the requests as a bulk and then correlates the responses.

The data collector server, extracts the list of monitored elements from the CMDB built by the Network Discovery product using SQL filtering commands as defined in the conf/ICMPSQLFilteringFile.txt in the server side.

Status monitoring

By default the product detects if an IP address responds to an Echo request. It monitors changes in the IP address status, and notifies the data collection server when the status changes.

The data collection server launches a change status job, on this event.

Sending Response time information

The agent has a run mode that enables it to send monitoring information which contains response time to the echo request, to the data collection server. This information may be used for response time reports that are supplied by the product.

Flow control parameters

The agent has multiple flow control parameters that enable it to fit any type of backbone.

Max concurrent requests

This mechanism defines the number of pending reply echo requests that can exist at a given time. When a node answers or a timeout is determined than a new address is echoed.

Send Delay

This mechanism determined how many ICMP echo requests will be sent as a bulk. It was built to prevent flooding the NIC card of the machine running the agent.

Start Time & Stop Time

The user can control when to start and when to stop monitoring the node.

This parameter can be set on the IP address level, where defaults can be set too.

Timeout & Retry

The user can control what is the timeout value that beyond it, it is considered that the device does not respond. It can control the number of retries that will be done until it is determined that the device does not respond.

This parameter can be set on the IP address level, where defaults can be set too.

Monitoring cycle timeout

The user can control the time between every measurement that is done on an IP address. The time is measured for every address separately, so it enables the spreading of the measurement evenly during time. The time is measured from the end of the previous cycle.

This parameter can be set on the IP address level, where defaults can be set too.

Jitter measurement

The ICMP data collector agent has the capability of measuring semi jitter information on the remote IP addresses.

This information is collected by sending multiple ICMP requests to the target IP in a single monitoring cycle, and monitoring the deference in the response times between the different requests.

The user can determine the number of requests that compose a single cycle. The output returned by the agent, is then processed, and average, mean,Max, Median and standard deviation are calculated.

The user can define limits to those values that when reached will launch an event handler.

The SNMP Data collector logic

The SNMP Data collector agent was designed to collect large numbers of SNMP parameters from large numbers of devices in very short cycles.

It sends SNMP requests and parses the replies based on user defined instructions, and processes and logs the results.

It is a light weight agent that does not require any additional software locally (It does not require any database for its operations). It communicates with its data collection server, using a secured protocol on top of the tcp protocol.

The collector functionality

The collector gets a list of instructions which specify what to monitor and from where to extract the SNMP data. The commands are located in the conf/SNMPSQLFilterFile.txt in the server side.

It then sends the requests in parallel and asynchronous way to the defined targets based on the flow control parameters specified to it. It does not wait for a reply from a given device to send a request to another device. If there are more than one request to a given node, it accumulates them to a single request to improve performance.

Status monitoring

The SNMP Data collector agent comes by default with instructions on how to detect the statuses of interfaces of devices using the information extracted from the IF_OperStatus and IF_AdminStatus of the interfaces MIB.

This is a complementary operation of the ICMP data collection agent for status monitoring. It

the product detects what is the status of the interface. It does so by either not receiving any response from the SNMP request or by the oper/admin statuses of the interface. When the product detects a status change it notifies the data collection server about it. The data collection server launches a change status job, on this event.

Collecting SNMP data

The SNMP Data collector agent was designed to collect any SNMP data, either specific OIDs or Table values (i.e. all OIDs starting with a given prefix). The data collected is then accumulated on the agent and sent in bulks in a compressed and encrypted way to the data collection server. The server then

check if there are any conditions defined on the parameters selected values, and if yes, generates events on them, and logs the data, by sending it to its logging daemon.

There is also an RRD Daemon, that runs on the loggers log and builds RRD databases according to definitions supplied to it.

Flow control parameters

The agent has multiple flow control parameters that enable it to fit any type of backbone. It supports all the parameters supported by the ICMP data collector agent (please refer to the previous section for an explanation on them).

It also supports a few more parameters needed because of the SNMP protocol.

Max_concurrent_requests_for_node

This mechanism defines how many requests can be sent concurrently to a single node. It was found that some nodes do not support concurrent requests.

Max_OID_count_in_request

This mechanism defines how many OIDs will be put in a single SNMP request. It was found that some devices have a limited buffer to receive SNMP requests. A large number of requests increases the size of the request, but reduces the line utilization.

The SNMP Trap Data collector logic

The SNMP Trap Data collector agent was designed to collect & filter large numbers of SNMP traps from large numbers of devices in very short cycles.

It sends the SNMP traps that match its filters to the data collector for forwarding to other SNMP managers or for internal processing..

It is a light weight agent that does not require any additional software locally (It does not require any database for its operations). It communicates with its data collection server, using a secured protocol on top of the tcp protocol.

The collector functionality

The collector gets a list of filters which specify what to accept and what to drop, and according to the instructions in the filters list, it either drops or forwards the received traps.

The SQL Data collector logic

The SQL Data collector agent was designed to collect large numbers of SQL results from large numbers of databased in very short cycles.

It creates a thread pool that handles concurrently the SQL requests needed to be executed. It collects the output of each request and logs it on the server side , and if requested checks if the values have changes since the last time the query was done, and if so, it notifies the server about the change. It will also identify and notify if the request failed or the server did not respond.

It is a light weight agent that does not require any additional software locally (It does not require any database for its operations). It communicates with its data collection server, using a secured protocol on top of the tcp protocol.

The collector functionality

The collector gets a list of instructions which specify what to monitor and from where to extract the SQL queries result sets. The commands are located in the conf/SQLCommandsSQLFilterFile.txt in the server side.

It then queues the requests to an input queue for the worker threads that do the actual SQL queries. The threads wait on the input queue and extract the commands. The mechanism built supports the reservation of a connection pool to the remote databases. This option prevents the creation of a connection for every query done to the database.

When the worker thread completes the SQL request assigned to it, it writes the result into an output queue which is later processed by the agent main thread.

The number of worker threads is determined by the max_concurrent_requests value specified in the .prop file.

Status monitoring

The SQL Data collector agent can be instructed to collect and forward status information about the requests it is supposed to execute. The agent reports is the server is Up or Down based on the fact that he got a reply or not from the database server. It also reports if the record set that was returned in this cycle was different from the previous time a record set was returned. If this is the case, then the agent will report to the data collection the 'Changed' status.

This option enables the product to become a watcher on important database fields.

On top of what was said, the user can define conditions on the values returned by the query which are evaluated on the server side.

Collecting SQL data

The SQL Data collector agent was designed to collect any SQL query result set periodically. The complete result set is returned to the data collecting server and is logged there. The data collected is accumulated on the agent and sent in bulks in a compressed and encrypted way to the data collection server. The server then checks if there are any conditions defined on the parameters selected values, and if yes, generates events on them, and logs the data, by sending it to its logging daemon.

There is also an RRD Daemon, that runs on the loggers log and builds RRD databases according to definitions supplied to it.

Flow control parameters

The agent has multiple flow control parameters that enable it to fit any type of backbone. It supports most of the parameters supported by the ICMP data collector agent (please refer to the previous section for an explanation on them).

The CSV Data collector logic

The CSV Data collector agent was designed to collect information from large numbers of CSV files using SQL commands to query the CSV files. It was designed to perform its actions on large numbers of files in very short cycles.

It creates a thread pool that handles concurrently the CSV related SQL requests needed to be executed. It collects the output of each request and logs it on the server side, and if requested checks if the values have changes since the last time the query was done, and if so, it notifies the server about the change. It will also identify and notify if the request failed or the server did not respond.

It is a light weight agent that does not require any additional software locally (It does not require any database for its operations). It communicates with its data collection server, using a secured protocol on top of the tcp protocol.

The collector functionality

The collector gets a list of instructions which specify what to monitor and from where to extract the CSV files and which SQL queries to perform. The information is located in the conf/CSVCommandsSQLFilterFile.txt in the server side.

It then queues the requests to an input queue for the worker threads that do the actual HTTP requests to bring the CSV files and the SQL queries. The threads wait on the input queue and extract the commands. The mechanism built supports the reservation of a connection pool to perform the local SQL commands.

When the worker thread completes the SQL request assigned to it on the requested CSV file, it writes the result into an output queue which is later processed by the agent main thread.

The number of worker threads is determined by the max_concurrent_requests value specified in the .prop file.

Status monitoring

The CSV Data collector agent can be instructed to collect and forward status information about the requests it is supposed to execute. The agent reports is the server is Up or Down based on the fact that he got a reply or not from the web server. It also reports if the record set that was returned in this cycle was different from the previous time a record set was returned. If this is the case, then the agent will report to the data collection the 'Changed' status.

This option enables the product to become a watcher on important CSV files fields.

On top of what was said, the user can define conditions on the values returned by the query which are evaluated on the server side.

Collecting CSV data

The CSV Data collector agent was designed to collect any data based on the SQL queries performed on the selected CSV files, periodically. The complete result set is returned to the data collecting server and is logged there. The data collected is accumulated on the agent and sent in bulks in a compressed and encrypted way to the data collection server. The server then checks if there are any conditions defined on the parameters selected values, and if yes, generates events on them, and logs the data, by sending it to its logging daemon.

There is also an RRD Daemon, that runs on the loggers log and builds RRD databases according to definitions supplied to it.

Flow control parameters

The agent has multiple flow control parameters that enable it to fit any type of backbone. It supports most of the parameters supported by the ICMP data collector agent (please refer to the previous section for an explanation on them).

The Discovery Field Data collector logic

The Discovery Field Data collector agent is the most powerful agent supplied. It uses Discovery rules defined for the Jilroy Active Inventory product, and can be used to run those discovery rules in parallel and periodically to extract information using all the protocols supported by the Jilroy Active Inventory product.

This agent requires connection to the Inventory definitions and database in order to work, so usually it will be located on the Data collection server where the active inventory is also installed. There is a way to install the active inventory product on a remote machine and point the agent to that machine.

The collector functionality

For each monitored element the collector runs the discovery rule defined for it, and extracts the data from the field passed as a parameter with the monitored element (see the SQL filtering file for this agent for more details).

The agent can run multiple discovery operations concurrently, when the agent uses worker threads to perform the requested actions

As the discovery rules can define actions, the agent supports a periodical Active operation that will not fill any database, or can send the collected data to the center for logging or processing.

The number of worker threads is determined by the `max_concurrent_requests` value specified in the `.prop` file.

Status monitoring

The Discovery Field Data collector agent can be instructed to collect and forward status information about the requests it is supposed to execute. The agent reports is the server is Up or Down based on the fact that he got a reply or not from the Discovery rule. It also reports if the record set that was returned in this cycle was different from the previous time a record set was returned. If this is the case, then the agent will report to the data collection the 'Changed' status.

This option enables the product to become a watcher on important data fields in the network using any of the protocols supported by the Active Inventory product(ICMP, SNMP, HTTP, WMI, Tlenet, SSH, SQL, Open ports,...)

On top of what was said, the user can define conditions on the values returned by the query which are evaluated on the server side.

Collecting CSV data

The Discovery Field Data collector agent was designed to collect any data based on the Discovery rules defined to the Active Inventory product. The field which is defined to be returned to the data collecting server can be logged there. The data collected is accumulated on the agent and sent in bulks in a compressed and encrypted way to the data collection server. The server then checks if there are any conditions defined on the parameters selected values, and if yes, generates events on them, and logs the data, by sending it to its logging daemon.

There is also an RRD Daemon, that runs on the loggers log and builds RRD databases according to definitions supplied to it.

Flow control parameters

The agent has multiple flow control parameters that enable it to fit any type of backbone. It supports most of the parameters supported by the ICMP data collector agent (please refer to the previous section for an explanation on them).

The HTTP Data collector logic

The HTTP Data collector agent was designed to collect information from large numbers of web site using HTTP get or post requests. It was designed to perform its actions on large numbers of web sites in very short cycles.

It creates a thread pool that handles concurrently the HTTP requests. It collects the output of each request and optionally runs a regex operation on it. It logs the results on the server side, and if requested checks if the values have changes since the last time the query was done, and if so, it notifies the server about the change. It will also identify and notify if the request failed or the server did not respond.

It is a light weight agent that does not require any additional software locally (It does not require any database for its operations). It communicates with its data collection server, using a secured protocol on top of the tcp protocol.

The collector functionality

The collector gets a list of instructions which specify what to monitor and from where to extract the web pages and which regex to perform. The information is located in the -

conf/HTTPCommandsSQLFilterFile.txt in the server side.

It then queues the requests to an input queue for the worker threads that do the actual HTTP requests to bring the web pages and the regex operation. The threads wait on the input queue and extract the commands.

When the worker thread completes the HTTP request assigned to it and optionally the regex operation on the returned web page, it writes the result into an output queue which is later processed by the agent main thread.

The number of worker threads is determined by the max_concurrent_requests value specified in the .prop file.

Status monitoring

The HTTP Data collector agent can be instructed to collect and forward status information about the requests it is supposed to execute. The agent reports is the server is Up or Down based on the fact that he got a reply or not from the web server. It also reports if the data that was returned in this cycle was different from the previous time data was returned. If this is the case, then the agent will report to

the data collection the 'Changed' status.

This option enables the product to become a watcher on important web pages fields.

On top of what was said, the user can define conditions on the values returned by the query which are evaluated on the server side.

Collecting HTTP data

The HTTP Data collector agent was designed to collect any data based on the regex operations results on selected web pages, periodically. The result is returned to the data collecting server and is logged there. The data collected is accumulated on the agent and sent in bulks in a compressed and encrypted way to the data collection server. The server then checks if there are any conditions defined on the parameters selected values, and if yes, generates events on them, and logs the data, by sending it to its logging daemon.

There is also an RRD Daemon, that runs on the loggers log and builds RRD databases according to definitions supplied to it.

Flow control parameters

The agent has multiple flow control parameters that enable it to fit any type of backbone. It supports most of the parameters supported by the ICMP data collector agent (please refer to the previous section for an explanation on them).

The WMI Data collector logic

The WMI Data collector agent was designed to collect information from large numbers of Windows stations supporting the WMI protocol. It was designed to perform its actions on large numbers of stations in very short cycles.

It creates a thread pool that handles concurrently the WMI requests. It collects the output of each request and optionally runs a regex operation on it. It logs the results on the server side, and if requested checks if the values have changes since the last time the query was done, and if so, it notifies the server about the change. It will also identify and notify if the request failed or the server did not respond.

It is a light weight agent that does not require any additional software locally (It does not require any database for its operations). It communicates with its data collection server, using a secured protocol on top of the tcp protocol. As WMI is a Microsoft proprietary protocol, the agent is required to run on a windows machine. There is no requirement as for the location for the data collector or the monitoring platform server it self, the requirement is only for the agent.

The collector functionality

The collector gets a list of instructions which specify what to monitor and from where to extract the the data . The information is located in the -

conf/WMICommandsSQLFilterFile.txt in the server side.

It then queues the requests to an input queue for the worker threads that do the actual WMI requests to bring the WMI information and the regex operation. The threads wait on the input queue and extract the commands.

When the worker thread completes the WMI request assigned to it and optionally the regex operation on the returned data, it writes the result into an output queue which is later processed by the agent main thread.

The number of worker threads is determined by the max_concurrent_requests value specified in the .prop file.

Status monitoring

The WMI Data collector agent can be instructed to collect and forward status information about the requests it is supposed to execute. The agent reports if the station is Up or Down based on the fact

that he got a reply or not from the station. It also reports if the data that was returned in this cycle was different from the previous time that data was returned. If this is the case, then the agent will report to the data collection the 'Changed' status.

This option enables the product to become a watcher on important WMI fields.

On top of what was said, the user can define conditions on the values returned by the query which are evaluated on the server side.

Collecting WMI data

The WMI Data collector agent was designed to collect any data based on the instructions passed by the SQL filtering file, periodically. The result is returned to the data collecting server and is logged there. The data collected is accumulated on the agent and sent in bulks in a compressed and encrypted way to the data collection server. The server then checks if there are any conditions defined on the parameters selected values, and if yes, generates events on them, and logs the data, by sending it to its logging daemon.

There is also an RRD Daemon, that runs on the loggers log and builds RRD databases according to definitions supplied to it.

Flow control parameters

The agent has multiple flow control parameters that enable it to fit any type of backbone. It supports most of the parameters supported by the ICMP data collector agent (please refer to the previous section for an explanation on them).

The CheckIT database scheme

In this chapter we list the additions to the database scheme that were made due to the monitoring platform product. It does not list the tables described in the Network Discovery product documentation.

Name prefix conventions:

str – refers to string

n – refers to an Integer

tblStatusMonitoring

TblStatusMonitoring holds the status of the IP addresses and interfaces monitored by the different agents.

It contains entries from several type:

- Node – A network node entry
- Interface – An Interface entry (each SNMP discovered node) has one or more interfaces.
- Address – An Address entry, maps an IP address to a Node. A node might have several IP addresses.

nObjectId int,

strMonitoringMethod varchar NULL,

strIpAddr varchar NULL,

strStatus varchar NULL,

nCycle int NULL,

nIfIndex int NULL,

nDailyStatusChangeCount int NULL,

nLastStatusChangeTime int NULL,

nBackBoneFailureNode int NULL

Field Name	Usage
nObjectId	A unique ID of the “Address” Type entry for ICMP data collector or “Interface” entry for the SNMP data collector
strIpAddr	The IP address of the node on it the measurement is done.
strStatus	The Status of the object
nCycle	The monitoring cycle number
nIfIndex	The IF Index for SNMP data collector status
nDailyStatusChangeCount	This field counts the number of status changes that occurred for the given node in the past day
nLastStatusChangeTime	The time the last status change occurred.
nBackBoneFailureNode	A flag of (0 1) that indicates if this element was considered down due to a backbone failure. See the backbone failure chapter for more details.

tblServiceMessages

tblServiceMessages is used by the application server to send notification to the data collectors. It is an internal table.

Utility commands

snmptrap

The snmptrap command sends a trap to a target SNMP manager based on the parameters passed in the command line

Usage: snmptrap strIpAddr, nPort, strCommunity, strEnterprise, nGeneric, nSpecific, strOID , stringValue

Example: snmptrap 127.0.0.1 162 public

.1.3.6.1.4.1.9.17888.1.2.1 4 5 1.3.6.1 'trap message'

Parameter Description

- strTargetIp - IP address of traps receiver.
- nTargetPort - Port number of traps receiver.
- strCommunity - Community of traps receiver.
- strEnterprise - Enterprise OID of sent trap.
- nGeneric - Generic number of trap.
- nSpecific - Specific number of trap.
- strOID – oid of variable
- strValue – string value of variable

The CheckIT GUI

The CheckIT product comes with a Java based Graphical User Interface (GUI). The GUI can run on any platform that supports Java.

The Client uses HTTP(s) to access the application server. It can be located remotely to the product's core.



The product menu

The products menu has the following structure

- Administration – The main administration sub menu
 - Edit Product Config File – opens the products conf/config.porp file, and allows to edit it
 - Edit Provisioning File – opens the default provisioning file, pointed by the conf/config.xml file. See an explanation on this option in the Network Discovery product documentation.
 - License info – shows the current license information
 - Edit GUI Config File – opens the GUI's conf/configGUI.porp file, and allows to edit it
 - Edit Connection File – shows the local connections information. Please refer to the security chapter for more information
 - System status – This report currently lists the length of the SQL updates queue waiting for processing
 - Discovery rules – See Inventory documentation
 - Permissions Management – The permissions management sub menu
 - Users Management – opens a query panel with the users information. Please refer to the security chapter for more information

- Commands permission management – opens a query panels with the commands permissions information. Please refer to the security chapter for more information
- Exit – exit thg GUI
- Discovery Jobs – The sub-menu that handles the discovery operations. It is equivalent to the 'run' sub-menu in the 'Network Discovery' product. Please refer to the 'Network Discovery' product documentation for a detailed explanation on this sub-menu.
 - Run Discovery – This menu item launches the discovernet command with the ALL option.
 - Add Arp Info to Provisioning File – This menu item launches only the “Add Arp Info to Provisioning File” option of the “discovernet” command line.
 - Run Arp Provisioning and Discovery – This menu items run a command line that performs the 2 actions of adding the arp table to the provisioning file, and performing the discovery.
 - Discovery Jobs Status – This menu item, launches a panel, with the jobs status of the jobs launched by this sub-menu.
- Discovery Reports – This sub-menu lists all the discovery reports of the product. It is equivalent to the 'Reports' sub-menu in the 'Network Discovery' product. Please refer to the 'Network Discovery' product documentation for a detailed explanation on this sub-menu.
 - Nodes – This report lists all the discovered nodes
 - Interfaces – This report lists all the Interfaces discovered on the SNMP discovered nodes.
 - Vlans – This report lists the vlans discovered
 - Arp – This report lists the ARP entries discovered
- Monitoring Jobs – This sub-menu handles operations related to the monitoring actions.
 - Suspend
 - Suspend ICMP monitors – suspends the monitoring action of all ICMP agents. It does not stop the agent, but stops the monitoring action.
 - Suspend SNMP monitors – suspends the monitoring action of all SNMP agents. It does not stop the agent, but stops the monitoring action.
 - Request Cycle count information – In the tblMonitoringStatus we update the entieres of the monitored entries only when there is a status change. However sometimes the user wants

to know what is the actual cycle count. Selecting this option will cause all the agents to report the current cycle count, and all the data collectors will update the relevant monitored element entries with the current cycle count. In large scale monitoring this operation is very time and CPU consuming so it is recommended not to use it frequently. Currently there is no report that shows this information.

- Local Services Jobs status – A report that lists the status of all the jobs in the 'Services' scheduler category.
- WatchDog Report – A report that list status report of all processes which are watched by the WatchDog. These services can be remote.
- All Jobs Status – This report lists all the jobs in the scheduler's db. For more information look in the scheduler chapter.
- Monitoring Reports – This sub-menu contains monitoring related reports
 - Status summary – This report lists the total counts of elements and their status for each monitoring method
 - Interfaces status – This report lists the statuses of interfaces/addresses monitored by the ICMP & SNMP monitors
 - Service Level Report – A report that lists the statuses of the interfaces, addresses based on their service level agreement.
- Help – A sub menu with help information
 - About – Shows the product information
 - Documentation – Displays this documentation

The Product's Panels

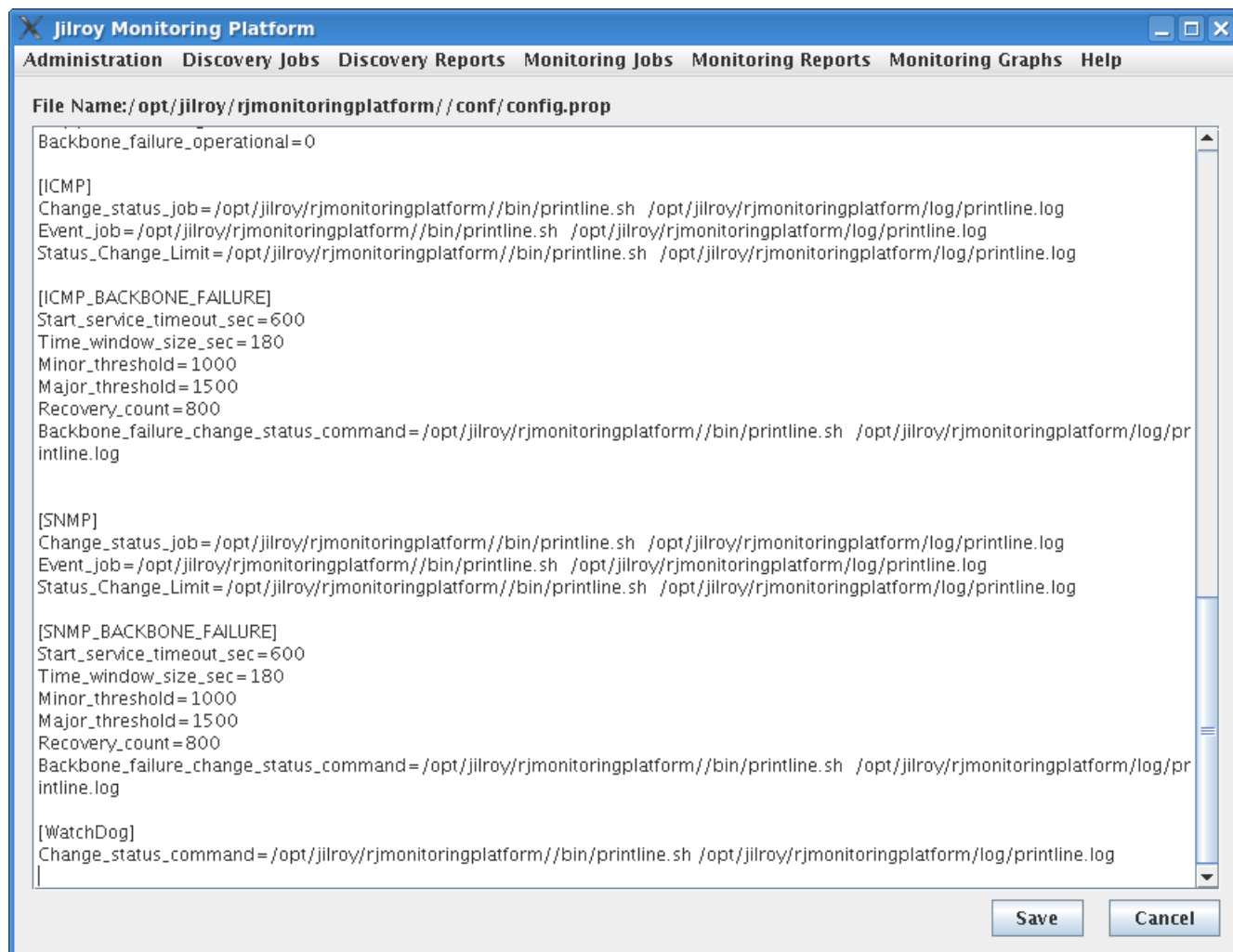
This chapter will describe the panels relevant for the CheckIT product. It will not explain about panels which are specific to the Network Discovery product. Please refer to the Network Discovery product documentation for explanation on these screens.

Edit Product Config File

This Panel will only function for the administrators role.

It displays the conf/config.prop file in the product's install directory.

It is reached from: Administration -> Edit Product Config File



The supported options are:

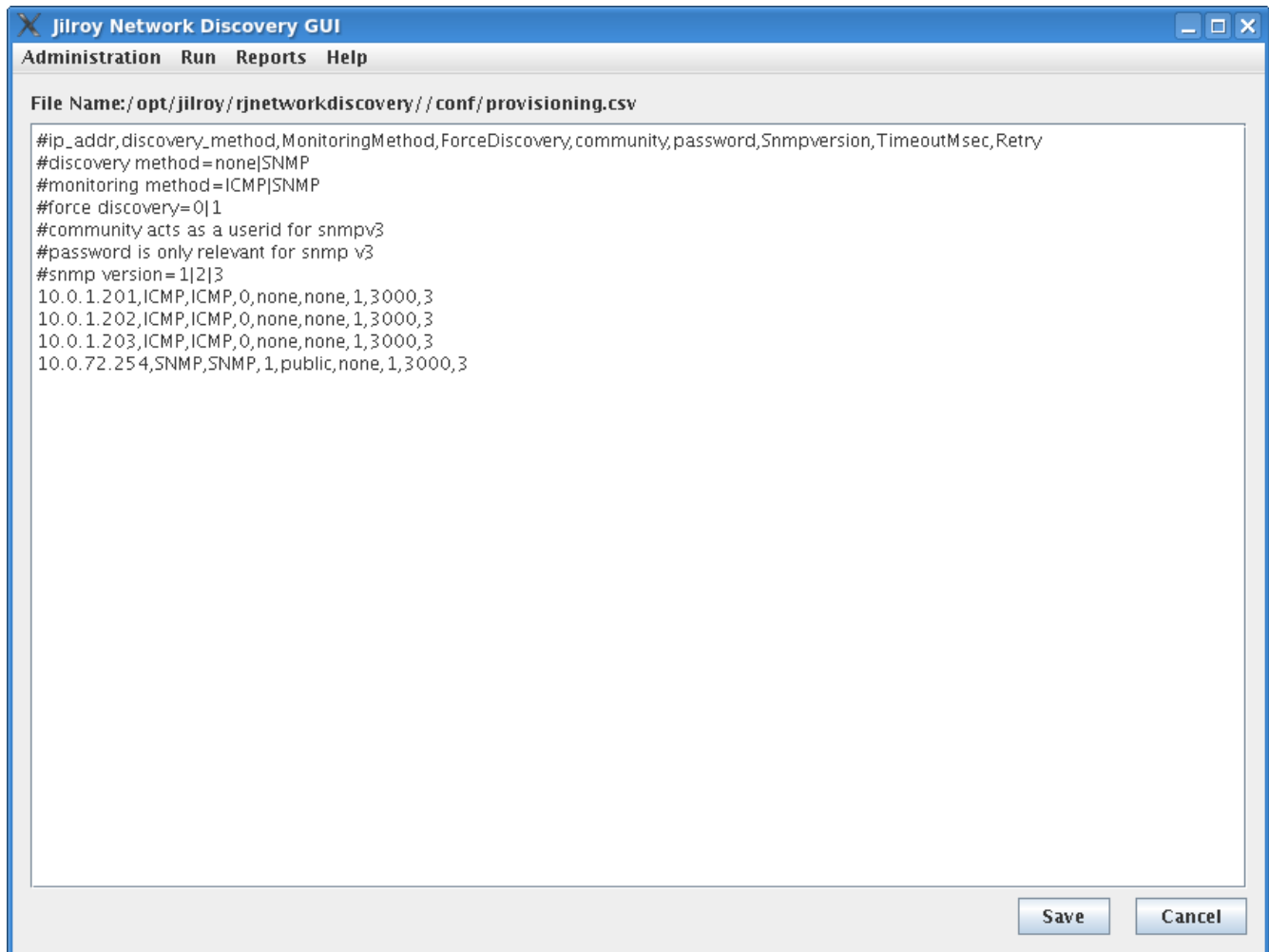
- Save – to save the changes done in the file
- Cancel – to leave the screen.

Edit Provisioning File

This Panel will only function for the administrators role.

It displays the default provisioning file pointed by conf/config.xml file. Its name is: conf/provisioning.csv file in the product's install directory.

It is reached from: Administration -> Edit Provisioning File



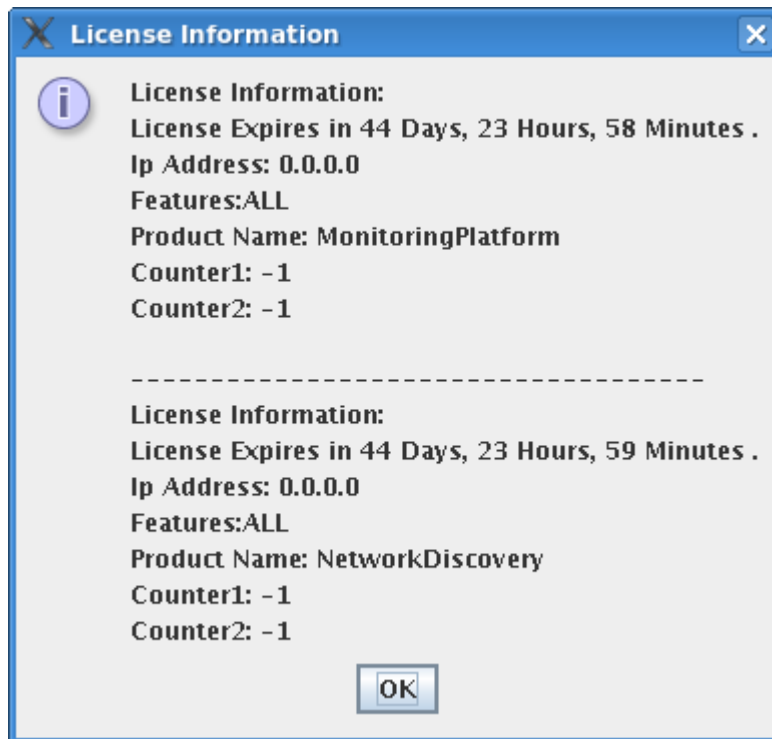
The supported options are:

- Save – to save the changes done in the file
- Cancel – to leave the screen.

License Info

This option will a message box with the current license information.

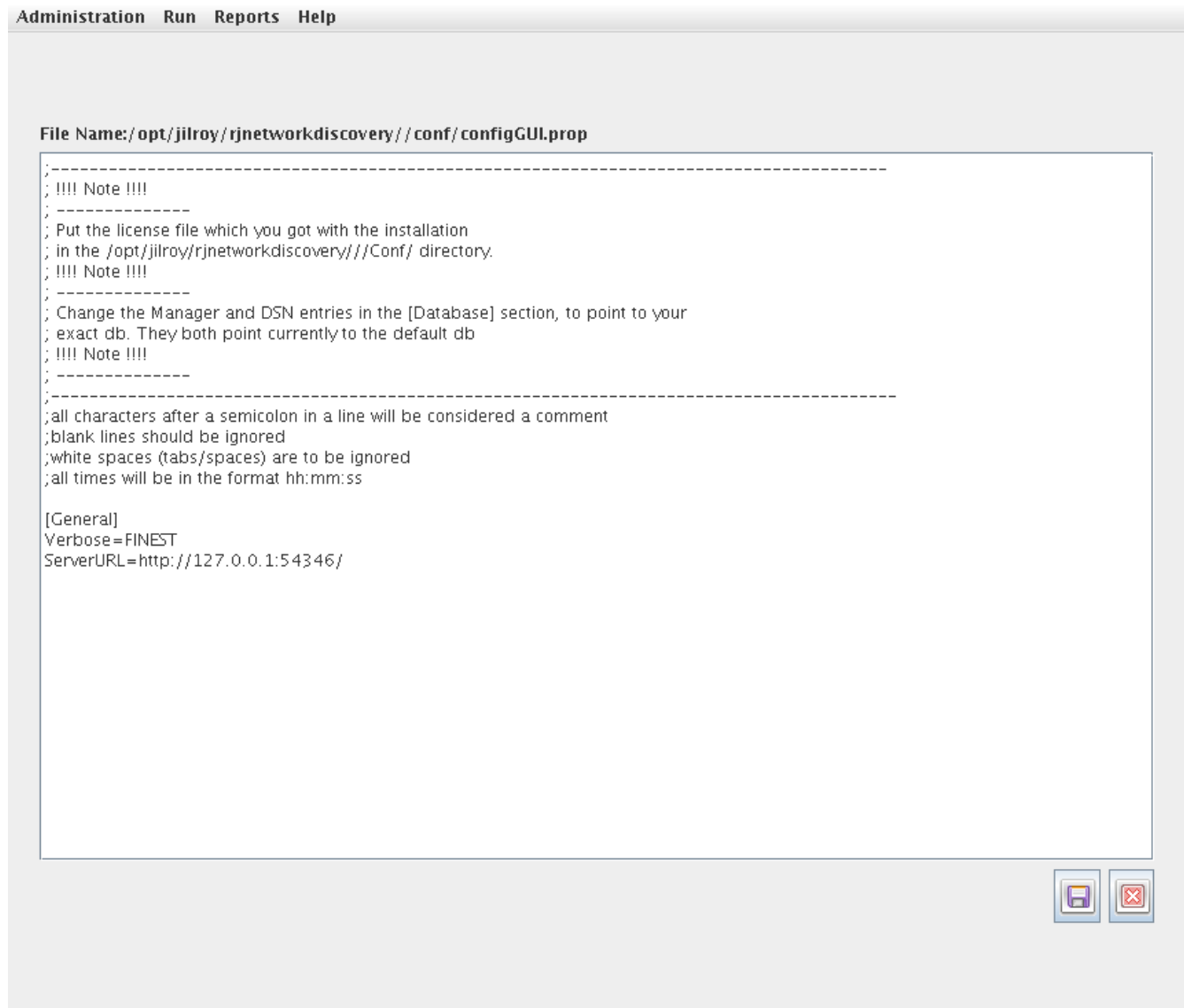
It is reached from: Administration -> License Info



Edit GUI Config File

This Panel displays the conf/configGUI.prop file in the GUI's install directory.

It is reached from: Administration -> Edit GUI Config File



The supported options are:

- Save – to save the changes done in the file

- Cancel – to leave the screen.

System Status



This Panel shows the number of SQL update requests pending for execution. The product has SQL worker threads that do the I/O asynchronously to the monitoring gathering process. This number should be low and is usually so after the system stabilizes.

[Administration](#) [Discovery Jobs](#) [Discovery Reports](#) [Monitoring Jobs](#) [Monitoring Reports](#) [Monitoring Graphs](#) [Help](#)

Report Name: System Status

Query Fields:

Actions:

  Lines Limit:

Query Results:

SQL_REQUESTS_QUEUE_SIZE
0

Actions:

- refresh – performs the query again
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of rows returned.

Local Services Job status

This Panel shows the status of the local services of the CheckIT product run by the scheduler. It shows the status of all the jobs in the “service” category of the scheduler. It is reached from: Monitoring Jobs -> Local Services Job Status








Administration Run Reports Monitoring Jobs Monitoring Reports Monitoring Graphs Help

Report Name: Local Services Job Status

Query Fields:

Job Category: Job Description: Job Id: Job Command:

Actions:

 Kill/Delete Type:       Lines Limit:

Query Results:

JOB_ID	JOB_NAME	JOB_CAT...	DESCRIPT...	JOB_COM...	START_TI...	STOP_TIME	Select
2	datacollectio...	Services	data collecti...	/opt/jilroy/jr...	Tue Apr 29 ...	null	<input type="checkbox"/>
3	icmpcollectio...	Services	ICMP collecti...	/opt/jilroy/jr...	Tue Apr 29 ...	Tue Apr 29 ...	<input type="checkbox"/>
4	snmpcollecti...	Services	SNMP collecti...	/opt/jilroy/jr...	Tue Apr 29 ...	null	<input type="checkbox"/>
5	sqlcollection...	Services	SQL collectio...	/opt/jilroy/jr...	Tue Apr 29 ...	null	<input type="checkbox"/>
6	csvcollection...	Services	CSV collectio...	/opt/jilroy/jr...	Tue Apr 29 ...	null	<input type="checkbox"/>
7	httpcollectio...	Services	HTTP collecti...	/opt/jilroy/jr...	Tue Apr 29 ...	null	<input type="checkbox"/>
8	loggingdaem...	Services	Logging Dae...	/opt/jilroy/jr...	Tue Apr 29 ...	null	<input type="checkbox"/>
9	rrddaemon	Services	RRD Daemo...	/opt/jilroy/jr...	Tue Apr 29 ...	null	<input type="checkbox"/>

The Panel shows the selected jobs based on the following query fields:

- Job description
- Job Id
- Job Command
- Job category (with the filter of services forced to it)

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

The fields are the ones described in the tblJobs database table.

Actions:

- refresh – performs the query again
- kill selected job
- Delete selected job – marks the job for deletion. It will be deleted after it ends, by the scheduler process itself
- Restart selected job – restart the selected job, only if it already ended.
- Select All – select all jobs returned.
- Un-select ALL – unselect all selected jobs.
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of rows returned.

WatchDog Report

The WatchDog report lists the status of the services defined to the watchdog. The location of the services can be on remote computers.

It is reached from: Monitoring Jobs -> WatchDog Report



Administration **Discovery Job: Discovery Report** **Monitoring Job: Monitoring Report** **Monitoring Graph** **Help**

Report Name: WatchDog Report

Query Fields:

Service Name: **Status:**

Actions:

  **Lines Limit:**

Query Results:

SERVICE_NAME	STATUS	CHANGE_TIME
ICMPDataCollectionAgent1	Down	Tue Apr 29 16:26:21 IDT 2008
SNMPDataCollectionAgent1	Running	Tue Apr 29 16:23:49 IDT 2008
SQLDataCollectionAgent1	Down	Tue Apr 29 16:26:21 IDT 2008
CSVDataCollectionAgent1	Running	Tue Apr 29 16:23:49 IDT 2008
HTTPDataCollectionAgent1	Down	Tue Apr 29 16:26:21 IDT 2008
WMIdataCollectionAgent1	Down	Tue Apr 29 16:26:21 IDT 2008
DbScheduler1	Running	Tue Apr 29 16:24:02 IDT 2008
DataCollectionServer1	Running	Tue Apr 29 16:23:49 IDT 2008
RRDDaemon1	Running	Tue Apr 29 16:23:49 IDT 2008
LoggingDaemon1	Running	Tue Apr 29 16:23:49 IDT 2008

The Panel shows the selected Nodes based on the following query fields:

- Service Name
- Status

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

The fields are the ones described in the tblObject database table.

Actions:

- refresh – performs the query again
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of raws returned.

All Jobs Status

This report is explained in the scheduler chapter.

It is reached from: Monitoring Jobs -> All Jobs Status

Status Summary

The Status Summary report lists the total count of all the monitored elements in each status and in each monitoring method.



[Administration](#) [Discovery Jobs](#) [Discovery Reports](#) [Monitoring Jobs](#) [Monitoring Reports](#) [Monitoring Graphs](#) [Help](#)

Report Name: Status Summary

Query Fields:

Status: Monitoring Method:

Actions:

  Lines Limit:


Query Results:

MONITORING_METHOD	STATUS	COUNT	CHANGE_TIME
CSV	UNKNOWN	1	Sat Sep 06 09:04:14 IDT 2008
DiscoveryField	UNKNOWN	1	Sat Sep 06 09:04:14 IDT 2008
HTTP	UNKNOWN	1	Sat Sep 06 09:04:14 IDT 2008
ICMP	DOWN	1	Sat Sep 06 09:08:38 IDT 2008
ICMP	UP	3	Sat Sep 06 09:08:38 IDT 2008
SNMP	ADMIN_DOWN	14	Sat Sep 06 09:04:34 IDT 2008
SNMP	DOWN	3	Sat Sep 06 09:04:34 IDT 2008
SNMP	UP	26	Sat Sep 06 09:04:34 IDT 2008
SQL	UNKNOWN	1	Sat Sep 06 09:04:14 IDT 2008

This report is updated before the interfaces status is updated. This is because the updates on the elements level are queued for the SQL worker threads to process, and these counters are updated within the monitoring data collection process synchronously.

The Panel shows the selected status counts based on the following query fields:

- Status
- Monitoring Method

 The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

Actions:

- refresh – performs the query again
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of rows returned.

Interfaces status

The Interfaces status report lists the status of all interfaces selected by the query parameter.

It is reached from: Monitoring Reports -> Interfaces Status








Administratior Discovery Job: Discovery Report Monitoring Job: Monitoring Report Monitoring Graph Help

Report Name: All Jobs Status

Query Fields:

Job Category: Job Description: Job Id: Job Command:

Actions:

 Kill|Delete Type: **Selected**       Lines Limit:

Query Results:

JOB_ID	JOB_NA...	JOB_CA...	DESCRIP...	JOB_CO...	START_...	STOP_TI...	Select
1	discovernet...	Default	discover ne...	/opt/jilroy/j...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
2	datacollecti...	Services	data collect...	/opt/jilroy/j...	Tue Apr 29...	null	<input type="checkbox"/>
3	icmpcollecti...	Services	ICMP collect...	/opt/jilroy/j...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
4	snmpcollect...	Services	SNMP collec...	/opt/jilroy/j...	Tue Apr 29...	null	<input type="checkbox"/>
5	sqlcollectio...	Services	SQL collecti...	/opt/jilroy/j...	Tue Apr 29...	null	<input type="checkbox"/>
6	csvcollectio...	Services	CSV collecti...	/opt/jilroy/j...	Tue Apr 29...	null	<input type="checkbox"/>
7	httpcollecti...	Services	HTTP collec...	/opt/jilroy/j...	Tue Apr 29...	null	<input type="checkbox"/>
8	loggingdae...	Services	Logging Da...	/opt/jilroy/j...	Tue Apr 29...	null	<input type="checkbox"/>
9	rrddaemon	Services	RRD Daem...	/opt/jilroy/j...	Tue Apr 29...	null	<input type="checkbox"/>
59	StatusChan...	SNMPChang...	Status Chan...	/opt/jilroy/...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
60	StatusChan...	SNMPChang...	Status Chan...	/opt/jilroy/...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
61	StatusChan...	SNMPChang...	Status Chan...	/opt/jilroy/...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
62	StatusChan...	SNMPChang...	Status Chan...	/opt/jilroy/...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
63	StatusChan...	SNMPChang...	Status Chan...	/opt/jilroy/...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
64	StatusChan...	SNMPChang...	Status Chan...	/opt/jilroy/...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
65	StatusChan...	SNMPChang...	Status Chan...	/opt/jilroy/...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
66	StatusChan...	SNMPChang...	Status Chan...	/opt/jilroy/...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>
67	StatusChan...	SNMPChang...	Status Chan...	/opt/jilroy/...	Tue Apr 29...	Tue Apr 29...	<input type="checkbox"/>

The Panel shows the selected Interfaces status based on the following query fields:

- IP Address of the node
- DNS Name
- Interface IfIndex
- Status

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

The fields are the ones described in the tblObject & tblMonitoringStatus database tables.

Actions:

- refresh – performs the query again
- Update Status – The interfaces that their status will be changes. Valid Values:
 - All – All interfaces
 - NotSignaled – When using backbone failure handling, all those interfaces that have gone down and did not launch the status change event handler, at that time
 - Selected – Those interfaces that have been selected in the GUI
- Update Value – The value the interface status will change to. Valid values are:
 - Up
 - Down
 - Admin Down
 - Oper Down
- Update – The update key, that actually changes the statuses of the interfaces.
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of raws returned.

Service Level Report

The Service Level report lists the status of all interfaces selected by the query parameter, based on their service level category and

It is reached from: Monitoring Reports -> Service Level Report

Administrator Discovery Job: Discovery Report Monitoring Job: Monitoring Report Monitoring Graph Help




Report Name: Service Level Report

Query Fields:

Service Level: Group Level 1: Group Level 2: Group Level 3: Group Level 4:

IP Address: DNS Name: Monitoring Method: Status:

Actions:

  Update Type: Update Value:  Lines Limit:

Query Results:

OB...	SE...	GR...	GR...	GR...	GR...	H...	IP_...	IF_...	IF_...	M...	ST...	CH...	Se...
17	Platin...	Jilroy	US	NY	RND		10.0...	3	Vlan3	SNMP	UP	Tue ...	<input type="checkbox"/>
18	Platin...	Jilroy	US	NY	RND		10.0...	10108	Giga...	SNMP	UP	Tue ...	<input type="checkbox"/>
19	Platin...	Jilroy	US	NY	RND		10.0...	10122	Giga...	SNMP	UP	Tue ...	<input type="checkbox"/>
20	Platin...	Jilroy	US	NY	RND		10.0...	10111	Giga...	SNMP	UP	Tue ...	<input type="checkbox"/>
21	Platin...	Jilroy	US	NY	RND		10.0...	5011	Port-...	SNMP	UP	Tue ...	<input type="checkbox"/>
22	Platin...	Jilroy	US	NY	RND		10.0...	10105	Giga...	SNMP	UP	Tue ...	<input type="checkbox"/>
23	Platin...	Jilroy	US	NY	RND		10.0...	10127	Giga...	SNMP	ADMI...	Tue ...	<input type="checkbox"/>
24	Platin...	Jilroy	US	NY	RND		10.0...	10119	Giga...	SNMP	UP	Tue ...	<input type="checkbox"/>
25	Platin...	Jilroy	US	NY	RND		10.0...	10112	Giga...	SNMP	ADMI...	Tue ...	<input type="checkbox"/>
26	Platin...	Jilroy	US	NY	RND		10.0...	6	Vlan6	SNMP	UP	Tue ...	<input type="checkbox"/>
27	Platin...	Jilroy	US	NY	RND		10.0...	1	Vlan1	SNMP	UP	Tue ...	<input type="checkbox"/>
28	Platin...	Jilroy	US	NY	RND		10.0...	10110	Giga...	SNMP	ADMI...	Tue ...	<input type="checkbox"/>
29	Platin...	Jilroy	US	NY	RND		10.0...	10121	Giga...	SNMP	UP	Tue ...	<input type="checkbox"/>
30	Platin...	Jilroy	US	NY	RND		10.0...	10118	Giga...	SNMP	UP	Tue ...	<input type="checkbox"/>
31	Platin...	Jilroy	US	NY	RND		10.0...	5009	Port-...	SNMP	UP	Tue ...	<input type="checkbox"/>
32	Platin...	Jilroy	US	NY	RND		10.0...	5	Vlan5	SNMP	UP	Tue ...	<input type="checkbox"/>
33	Platin...	Jilroy	US	NY	RND		10.0...	10128	Giga...	SNMP	ADMI...	Tue ...	<input type="checkbox"/>

The Panel shows the selected Interfaces/Addresses based on their service level and group parameters. The entries displayed are based on the following query fields:

- Service Level

- Group level1 – level 4
- IP Address of the node they were last discovered on
- DNS Name
- Monitoring Method
- Status

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

The fields are the ones described in the tblObject & tblMonitoringStatus database tables.

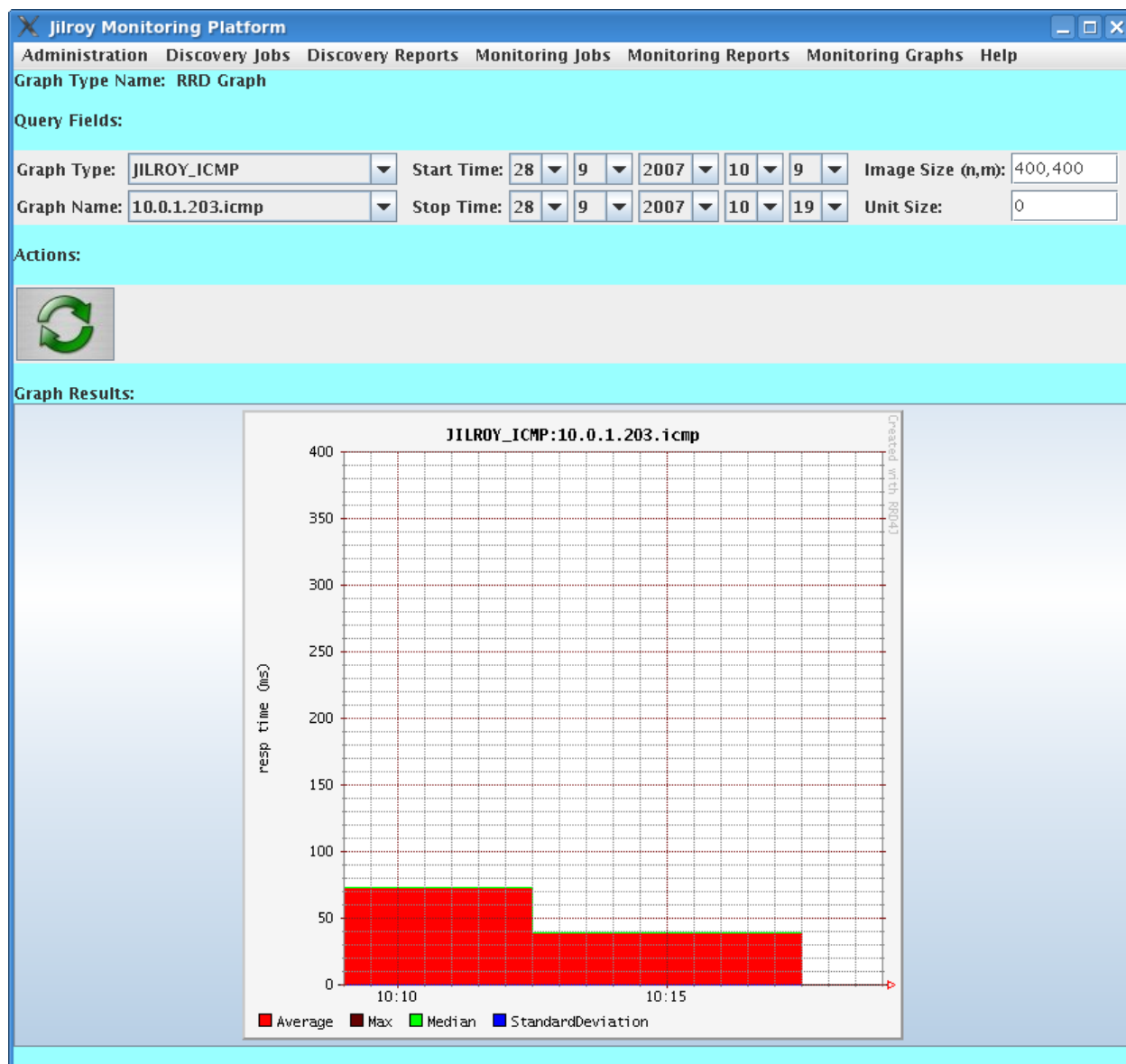
Actions:

- refresh – performs the query again
- Update Status – The interfaces that their status will be changes. Valid Values:
 - All – All interfaces
 - NotSignaled – When using backbone failure handling, all those interfaces that have gone down and did not launch the status change event handler, at that time
 - Selected – Those interfaces that have been selected in the GUI
- Update Value – The value the interface status will change to. Valid values are:
 - Up
 - Down
 - Admin Down
 - Oper Down
- Update – The update key, that actually changes the statuses of the interfaces. Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of raws returned.

RRD Graph

The RRD graph is a report that displays the output of the graphs collected using the RRD definitions specified in conf/rrd files, and in the monitored elements.

It is reached from: Monitoring Graphs -> RRD Graph



The Panel shows the selected graph based on the following query parameters:

- Graph Type – The name of the group of reports
- Graph Name – The exact specific name of the rrd report (for a specific element when it is relevant)
- Start Time
- Stop Time
- Image Size – specifies the size of the graph created
- Unit Size – specifies the units of the graph y axis.

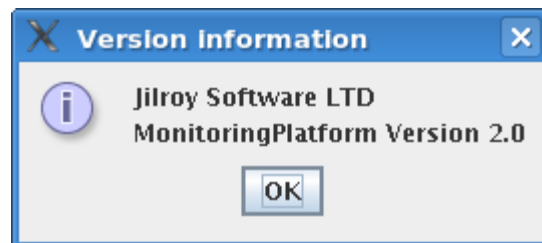
Actions:

- refresh – performs the query again

The graphs created are also saved as files in the temp/ dir on the application server computer.

About

The About menu option lists the product version information.



Backup and Restore for the Database

The product supports a way to backup its database into CSV files, and restoring the tables from the CSV files.

exporttable2csv

This command exports a table to a CSV file, that can be later loaded back to the database using a supplied utility

The following is the output of the command when no parameters are supplied.

```
format: exporttable2csv csv_file_name
```

where the *csv_file_name* is the full path to the file with instructions about the operation requested.

The file structure is:

- Line 1:Table name
- Line 2:CSV File full path
- Line 3:Separator String
- Line 4:Replace separator string

If separator is found in a table field then it replaces it with the replace string

importcsv2table

This command imports a CSV file in to a table.

The following is the output of the command when no parameters are supplied.

```
format: importcsv2table csv_file_name
```

where the *csv_file_name* is the full path to the file with instructions about the operation requested.

The file structure is:

- Line 1:Table name
- Line 2:CSV File full path
- Line 3:Separator String

Additional SQL related commands

The product is highly based on the database. It has several user commands which are related to the database.

Query analyzer for the database

In order to drill down into the database, and perform SQL queries on the product's database, we recommend to use DbVisualizer. Its free version is good enough for all required actions.

createdb

This command is creating the product's database. It extracts the relevant information about the name of the database from the `conf/config.xml` file and from the `env.db` script.

```
format: createdb  config.xml
```

createtables

This command creates the product's tables. It extracts the tables information from the definition files located in the `/conf/tables/` directory.

```
format: createtables  config.xml
```

dbconf

This command is recreating the database. It is used without parameters.

It can be reused when there are problems with the database.

loadtableinfo

This command is used to load data into a given table. It is similar to the importcsv2file, but is easier to use

```
format: loadtableinfo table_name csv_file_name separator
```

Where

1. Table_name - The table name which will be update
2. csv_file_name – The full path to the CSV file containing the data
3. separator – The separator string, in the csv file

sqlexec

This command is used to execute SQL commands on the database form a batch file.

```
format: sqlexec sql_commadns_file_name
```

The sql commands file requires that every command will be in a single line.

startdb

This command starts the database. It should be added to the crontab on the computer startup. It is currently tailored to use the hsql DB.

stopdb

This command stops the database. It is currently tailored to user the hsql DB.

truncatetables

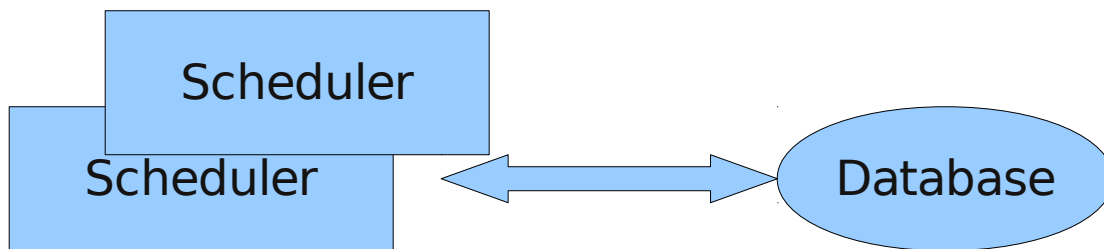
This command line truncates all the tables defined in the conf/tables configuration files.

Usage:truncatetables ALL table_name

The product Scheduler

The product comes with a Scheduler process, which handles the launching of the product related services and its event handlers.

This scheduler is based on the database and manages its scheduling based on the information stored in the database.



The scheduler runs jobs. A job is a single autonomic unit of work. By default the scheduler runs the jobs as external processes asynchronously, however there are commands that are run internally by the scheduler as part of the scheduler code.

The scheduler comes with tools to schedule jobs and on manage the scheduled jobs:

- Scheduler GUI
- Scheduler commands

The scheduler was designed in a way that will enable to limit the number of jobs that will run concurrently.

The scheduling process was design for scale. It is possible to split commands between multiple schedulers running on different machines in-order to improve performance.

The Scheduler's tables structure

tblJobs

tblJobs holds the scheduled jobs information.

Field Name	Usage
nSchedulerId	The Id of the scheduler that should process the current job.
nJobId	The ID of the Job
strCommand	The command line that is executed
strStatus	This field can contain the current status of the job, if the job it self supports its update
strName	The name of the job
strDescription	A description of the jobs work
nScheduleTime	The time the job should start (if null, then it will start immediately
nStartTime	The time the job started (or NULL)
nStopTime	The time the job ended (or NULL
nRestartFrequencyMin	Determines if the job should be restarted and if yes, how long after it has finished. The time is in minutes
strJobCategory	The category that the job belongs to
fDelete	Indicates if the job should be deleted from the table by the scheduler (values:0,1)

Categories file

The Scheduler has a mechanism for preventing over-flooding of the system with event-related jobs. The mechanism is based on Job category classes. You

can define the maximal amount of concurrent jobs that should run from a given category class. If the full capacity of the class has been reached, then the additional jobs are queued until the previous ones end.

The classes are defined in the file conf/categories.txt:

Categories.txt file

```
# Scheduler categories
Default,10
DiscoveryEvents,10
```

A remark line starts with #

The format of every raw is:

- Category name
- Maximum number of concurrent jobs from the given category.

After changing the file the Scheduler must be restarted.

GUI









Administration Run Reports Help

Report Name: All Jobs Status

Query Fields:

Job Category: Job Description: Job Id: Job Command:

Actions:

 Kill|Delete Type: 
      Lines Limit:

Query Results:

JOB_ID	JOB_NAME	JOB_CAT...	DESCRIP...	JOB_CO...	START_T...	STOP_TL...	Select
1	discovernet...	Default	discover net...	/opt/jilroy/r...	Fri Apr 25 1...	Fri Apr 25 1...	<input type="checkbox"/>
61	Run Discovery	Default	GUI Launche...	/opt/jilroy/r...	Fri Apr 25 1...	null	<input type="checkbox"/>

This panel is reached through:

Reports->All Jobs Status

The Panel shows the selected jobs based on the following query fields:

- Job category
- Job description
- Job Id
- Job Command

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

The fields are the ones described in the tblJobs database table.

Actions:

- refresh – performs the query again
- kill selected job
- Delete selected job – marks the job for deletion. It will be deleted after it ends, by the scheduler process itself
- Restart selected job – restart the selected job, only if it already ended.
- Select All – select all jobs returned.
- Un-select ALL – unselect all selected jobs.
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of rows returned.

Internal commands

Internal commands are commands that are executed synchronously in the scheduler process and are not forked and run as an external process.

There is currently only one specific internal command recognized by the scheduler, and a generic mechanism to run user defined classes that extend the “com.rj.dbscheduler.JavaInternalJob” abstract class.

snmptrap

The snmptrap command sends a trap to a target based on the parameters passed in the command

line. Look at the utilities chapter for a description on the parameters of this command.

JavaInternalJob abstract class

The mode to use this command is by writing in the job command line the following data:

JavaInternalJob InLine [Parameter [Parameter ...]]

Currently only inline run mode is supported. In the future we plan to support thread execution of commands.

Look at the supplied code sample named SampleJavaInternalJob.java found in our ftp site, for an example on how to write your on java internal jobs.

Loading jobs to the database

The Scheduler comes with a command that enables it to load jobs into the tblJobs.

loadjobsfromfile

This command load an XML file containing jobs information into the tblJobs.

Usage:loadjobsfromfile Jobs.xml.file

The XML file format is:

```
<ProcessList>

  <JobEntry>

    <strJobName>discovernetwitharp</strJobName>

    <strJobDescription>job description</strJobDescription>

    <strJobCategoryName>Default</strJobCategoryName>

    <strDelayTime>00:00:00</strDelayTime>

    <strJobCommandLine>command full path</strJobCommandLine>

    <strRestartFrequencyMin>24:00:00</strRestartFrequencyMin>
```

```
<nSchedulerId>1</nSchedulerId>

</JobEntry>

</ProcessList>
```

ProcessList

This is a wrapper tag for all entries of in the XML file

JobEntry

This tag contains the jobs definition tags

strJobName

The name of the job

strJobDescription

The job description

strJobCategory

The name of the job category

strDelayTime

The number of minutes after the load, that the job will be able to start.

strJobCommandLine

the full command line of the job

strRestartFrequencyMin

The time in minutes after the job end, that it should be restarted again

nSchedulerId

The number of the scheduler that will run the job.

i If the job has to update its status in the tblJobs strStatus field, you can pass to the job aparameter with the value @JobId@, that will be replaced with the Job Id. You can use this information to know the exact Job entry.

The product Security mechanism

The product has a security mechanism implemented in its Application Server (tomcat) and implemented through the GUI.

The permissions mechanism is built on the following concepts:

- Users belong to given roles
- Certain commands can be executed only by certain roles.

The authentication information is passed with each request by the GUI, based on a connection_info file held in the conf/connection.prop file.

The Users and Permissions table

In order to implement the security mechanisms 2 tables were added to the products database.

tblUsers

tblUsers contains the Users and Roles information

Field Name	Usage
strUser	The User Name
strPassword	The User's Password
strRole	The Role of the user.

A user can belong only to a single role.

Default values

The product comes with the admin user defined, where its password is admin, and its role is administrator.

Do not delete this entry, but you are allowed to change the password of the administrator.

tblPermissions

tblPermissions contains the information about the permitted commands for a given role

Field Name	Usage
strCommand	The Command Name
strRole	A Role that has of the user.

A given command can be allowed for multiple roles.

Default values

The product comes with the @ALL@ entry with the role administrator, which defines that the administrator can execute all commands.

Do not delete this entry.

The connection.prop file

This file contains the security information of the current user.

The format of the file is:

```
[General]

User=admin

Password=admin

EncryptionKey=0123456789abcdef
```

User

The User name of the GUI user.

Password

The password of the GUI user.

EncryptionKey

An optional parameters when there is communication between peers when the application server is acting as a relay.

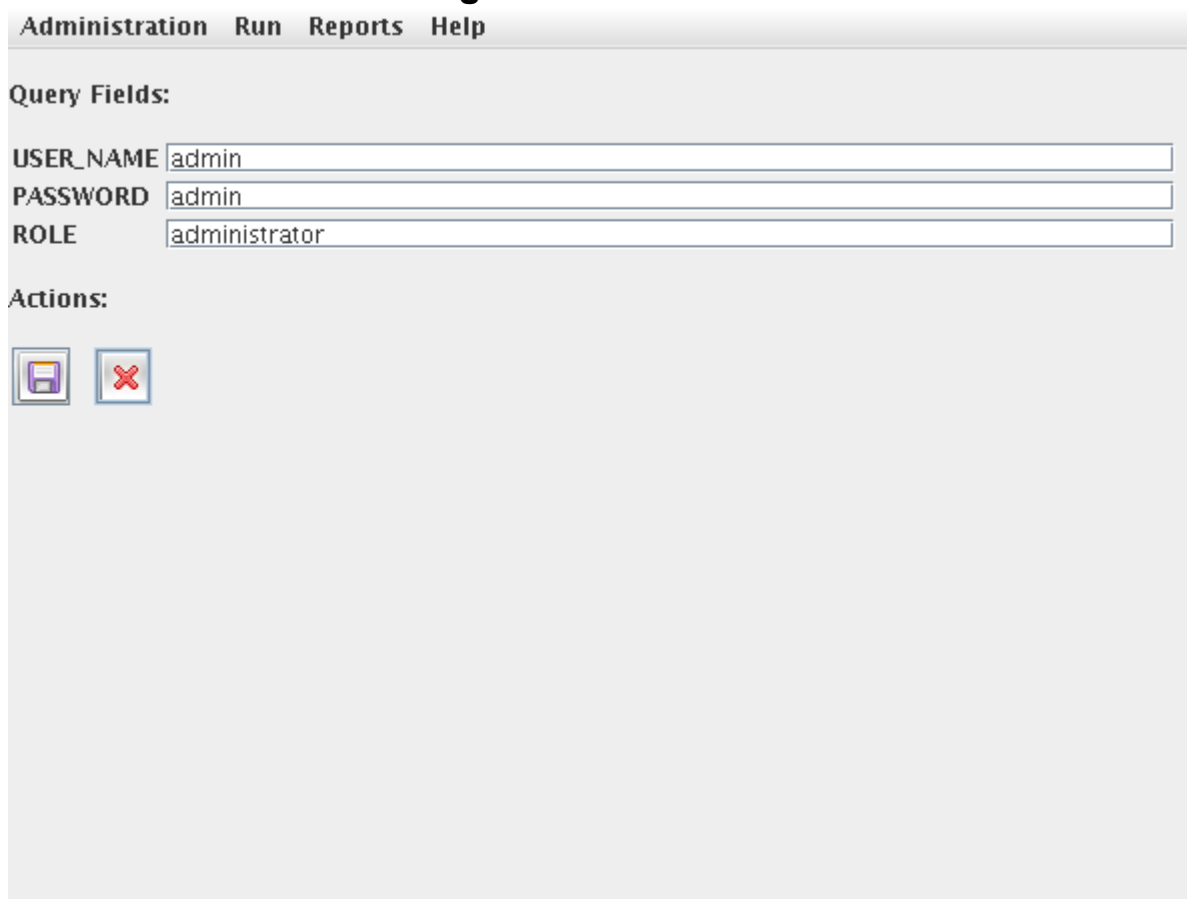
This parameter is not relevant with most of our products.

The Security GUI

The product comes with several screens to manage the security mechanism

Client Side

connection information management



Administration Run Reports Help



Query Fields:

USER_NAME

PASSWORD

ROLE

Actions:

This panel is reached through:

Administration->Edit Connection File

The fields are the ones described in the conf/connection.prop

Actions:

- Save – save values in the conf/connection.prop file
- Cancel – End without doing anything

Server Side

the users definition panel

This screen is used to define the users of the system.






Administration Run Reports Help

Report Name: Users management

Query Fields:

User: Role:

Actions:

     Lines Limit:

Query Results:

USER_NAME	PASSWORD	ROLE	Select
admin	admin	administrator	<input type="checkbox"/>

This panel is reached through:

Administration->Permissions Management -> Users Management

The Panel shows the selected users based on the following query fields:

- User – User name
- Role – Role Name

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

The fields are the ones described in the tblUser database table.

Actions:

- refresh – performs the query again
- Add - adds a new entry to the users table
- Delete – Delete the selected entries form the users table
- update – Update the first selected entry from the users table
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of raws returned.

the Commands permissions panel

This screen is used to define the commands permissions. It defines the roles that can execute given commands.

This panel is reached through:

Administration->Permissions Management ->
Commands Permissions Management

The Panel shows the selected users based on the following query fields:

- Command – Command name

- Role – Role Name

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .





Administration Run Reports Help

Report Name: Commands Permission management

Query Fields:

Command: Role:

Actions:

    Lines Limit:

Query Results:

COMMAND	ROLE	Select
@ALL@	administrator	<input type="checkbox"/>

The fields are the ones described in the tblUser database table.

Actions:

- refresh – performs the query again
- Add - adds a new entry to the users table
- Delete – Delete the selected entries form the users table
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of raws returned.

General Commands permitted

Following is a list of all the common application server commands:

RRD

A code for allowed the RRD related commands.

Reports commands

For each Report the user ask, you can define a security entry(It applies also to your extensions – see the GUI programmers guide). The name you specify as the query name in the form definition.

The Watch-Dog

The product comes with a built in watch-dog service. This service accesses all the processes that are used by the product regardless to their location, and notifies by launching an event handler and via a report about the current status of the services.

The watch-dog uses a keep-alive built in mechanism with in the services to validate that the processes are running and responsive.

How to define the watched services

There are 2 files that control what will be monitored by the watch dog.

WatchDogServicesList.txt

This file defines the names of the services that should be monitored.

```
# WatchDog Services Names & timeouts  
  
ICMPDataCollectionAgent1,300  
  
SNMPDataCollectionAgent1,300  
  
DbScheduler1,300  
  
DataCollectionServer1,300  
  
RRDDaemon1,300  
  
LoggingDaemon1,300
```

The name of the service is defined in its conf/config.xml file (as it was started) in the tag named <ProcessName>.

In each line you will write the process name and the timeout after it, it is assumed that the process is down.

The watchdog tries to “ping” the services that were connected to it every half timeout value.

WatchDogAgentList.txt

The second file, is the agent list of the watchdog that should be set to either connect or accept connections from all the services that should be watched.


```
# Agent list file for the server to connect to

# format: IP, port, flag that tells what type of agent it is(Listener = true, Connector = false)

127.0.0.1,54347,listener,/home/raphyz/projects/RJServerAgent/dist/conf/enckey.txt

127.0.0.1,-1,Connector,/home/raphyz/projects/RJServerAgent/dist/conf/enckey.txt
```

An explanation on the file is found in the “Agents Technology” chapter.

 You will have also to set the agent list file for the servers you create, or update it, for servers, you changed their location.

Watch-Dog Database table scheme

The Watch-dog keeps its status information in the tblServiceStatus table.

tblJobs holds the scheduled jobs information.

Field Name	Usage
strServiceName	The name of the service
strStatus	The current known status of the service
nStatusChangeTime	The time the last status change was discovered.

Watch-Dog GUI

The Watch-dog supplies a report which lists the defined services current statuses. This report is called

“WatchDog Report”.

The report does not allow to restart the services, as the services might not be located on the same platform as the application server.



Administration **Discovery Job: Discovery Report** **Monitoring Job: Monitoring Report** **Monitoring Graph** **Help**

Report Name: WatchDog Report

Query Fields:

Service Name: **Status:**

Actions:


  **Lines Limit:**

Query Results:

SERVICE_NAME	STATUS	CHANGE_TIME
ICMPDataCollectionAgent1	Down	Tue Apr 29 16:26:21 IDT 2008
SNMPDataCollectionAgent1	Running	Tue Apr 29 16:23:49 IDT 2008
SQLDataCollectionAgent1	Down	Tue Apr 29 16:26:21 IDT 2008
CSVDataCollectionAgent1	Running	Tue Apr 29 16:23:49 IDT 2008
HTTPDataCollectionAgent1	Down	Tue Apr 29 16:26:21 IDT 2008
WMIDataCollectionAgent1	Down	Tue Apr 29 16:26:21 IDT 2008
DbScheduler1	Running	Tue Apr 29 16:24:02 IDT 2008
DataCollectionServer1	Running	Tue Apr 29 16:23:49 IDT 2008
RRDDaemon1	Running	Tue Apr 29 16:23:49 IDT 2008
LoggingDaemon1	Running	Tue Apr 29 16:23:49 IDT 2008

The information presented by this report is taken from the tblServiceStatus. The Panel shows the selected services based on the following query fields:

- Service Name
- Service Status

 The values entered in the query fields can be prefixes. To enter a prefix or a wild character write

the string you want with % at its end or at the location of the wild character .

The fields are the ones described in the tblServiceStatus database table.

Actions:

- refresh – performs the query again
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of rows returned.

Backbone failure processing

The monitoring platform acting as a status monitor, has the capability of identifying backbone failures, and taking special actions in this case.

What is a backbone failure

A backbone failure is usually a case when a major junction in the backbone fails and this leads to a case of many interfaces going down in a short period.

In this case usually the user will not want to be notified on every interface going down as they will cause an event storm on the operator's console, and as most of the interfaces are not really down but unreachable. Instead he would like to know that a backbone failure occurred.

When the backbone failure is resolved the user will like to know the exact status of the interfaces that went down during the backbone failure.

As the monitoring platform does not hold a full layer2 map of the organization, it can not know the exact path from its monitor to the every interface. A trace route solution is not valid too, as it will take too much time to perform.

Because of the CheckIT uses a heuristic method to discover a backbone failure situation. The assumption that the product makes is that a backbone failure happens when a large number of interfaces go down at a relatively short period.

Backbone Failure configuration parameters

The backbone failure related parameters are found in the conf/config.prop file.

It is found in the following sections:

- `Default_monitor_parameters` – In this section we define if this feature is activated or not.
- `ICMP_BACKBONE_FAILURE` – Backbone failure parameters relating to the ICMP monitoring method.
- `SNMP_BACKBONE_FAILURE` – Backbone failure parameters relating to the SNMP monitoring method.

Please refer to the configuration chapter for a detailed description on the relevant parameters in these sections.

Backbone Failure feature logic

When an interface changes its status to down, the Data connection server, checks if a backbone failure situation exists for the given monitoring method.

The check is done by counting the number of interfaces that went down in the defined time window, for the given monitoring method.

If the minor threshold was not reached then in the status change job for the interface, an indication of no backbone failure state ("NoBackBoneFailure") is indicated.

If the minor threshold was reached then in the status change job for the interface an indication of a possible backbone failure situation occurs ("PossibleBackboneFailure").

If the major threshold is reached, the product indicates that it is in a Backbone failure state, and stops running event handlers for falling interfaces.

When the product enters the backbone failure state it runs an event handler for the backbone failure change status event.

When the product is within the backbone failure state, a check is done if the number of interfaces failing within the time window is below the recovery limit. If this is the case the backbone failure is reset and all the interfaces that have changed to a down state and for them the change status event was not launched, are changed to an up state. In this case if they are truly down, then an event on this situation will be launched now.

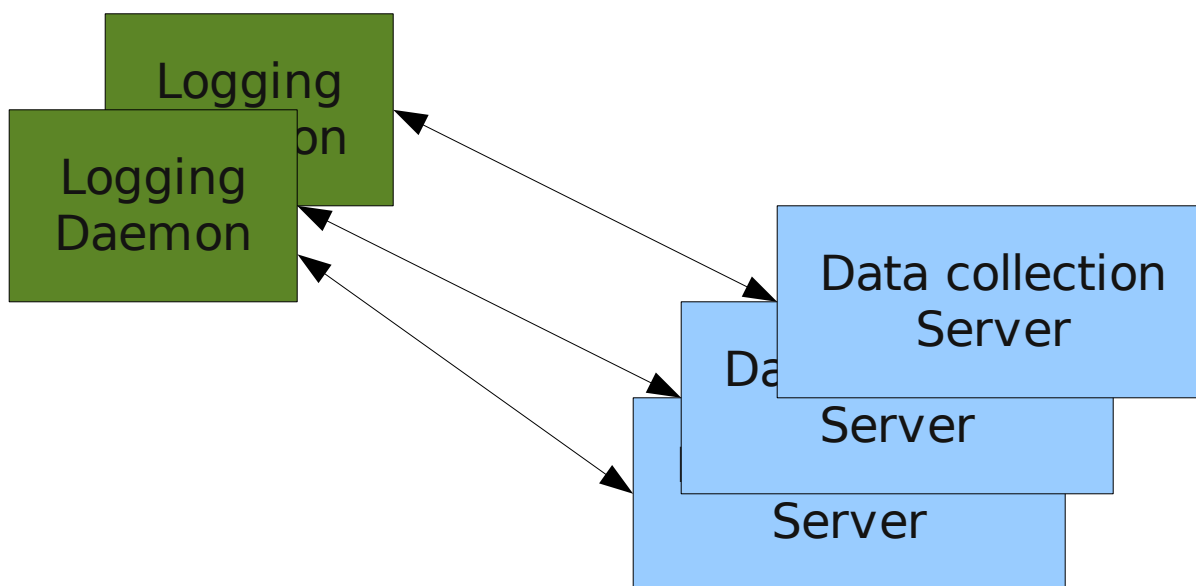
Logging Daemon

The product comes with a Logging daemon, that receives logging data from the data collectors servers and writes them to a log file. There could be multiple Logging daemons installed.

The Idea of separating the logging daemon from the data collector servers was to allow the product to scale with its data collector servers.

The logging data is passed to the logging server using a UDP port on which the logging daemon listens for messages.

When a message arrives to the logging daemon, it is written in a format of: a Length of the record separated by a new line, and then the data itself.



The current components in that use the logged data are the RRD Daemon, that are explained in an other chapter.

The output of the logging daemon is located in a the data/ directory. There is a size limit to a single log file, which is configured in the conf/configLoggingDaemon.xml. When the log file reaches the given size, a new file is generated with the same prefix but with a higher index as a suffix.

Configuring the Logging Daemon

The logging daemon is configured by the conf/configLoggingDaemon.xml file, and does not have frequently changed, user related parameters.

The most relevant parameter used by the daemon is its Port ID, and the parameters for the logging daemon clients are its location (IP address & Port number).

Additional parameters

- `LoggerFileName` – set the exact location of the logging files, and their prefix name.
- `MaxLoggerFileSize` – The maximum size of a logger file.

Event Log Daemon

The product comes with an Event log support. This support offers the following:

- An Event log console
- An Event log history report
- An Event actions report

Events

All the monitors report change status events to an event log, SNMP events are also reported to the event log, and there can be user reported events. All these can be processed by the event-log support.

All events are usually written to 2 tables: tblEventLog & tblEventLogHistory, where tblEventLog is dynamic and represents the current state of the system, when tblEventLogHistory represents all the events that were entered to the system.

tblEventLog has the following information:

- nTime: the time that the event was created
- strType: the type of the event
- strSubType: the sub-type category of the event
- strOrigin: the origin of the event
- strSubOrigin: the sub-type of the event
- strObjectType: the object type of the object that received the event
- strObjectId: the object id of the object that received the event
- strObjectName: the object name of the object that received the event
- strObjectSubName: the object sub-name of the object that received the event
- strSeverity: the severity of the event (normal, warning, severe)
- strStatus: the status of the event (open,closed)

- strOwner: Owner of the event
- strData: the full event data

Event Actions

The user can decide what will be done when an event arrives. TblEventActions determines the actions performed on each event, when multiple actions can be performed on a given event.

An Event action record contains the following fields:

- nExecutionOrder: Actions are executed by order, when the execution order is ascending
- strDescription: A description of the event action
- strType: the type of the event being processed
- strSubType: the sub-type of the event being processed
- strOrigin: the origin of the event being processed
- strSubOrigin: the sub-origin of the event being processed
- strObjectType: the object type of the object that event being processed handles
- strObjectId: the object id of the object that event being processed handles
- strObjectName: the object name of the object that event being processed handles
- strObjectSubName: the object sub-name of the object that event being processed handles
- strSeverity: the severity of the event being processed
- strOwner: the owner of the event being processed
- strDataSeperator: the type of the event being processed
- strDataCondition: the type of the event being processed
- strActionType: the action type that should be executed for the given event. Currently there are a few actions defined:
 - AddToConsoleDb: Add event to console database
 - WriteToEventlog: Write event to event log text file

- AddToHistoryDb: Add event to event log history table
- RemoveEvent: remove all matching events from the console
- CloseEvent: close all matching events in the console
- AssignEvent: assign event to an owner specified in the action data
- System: run a shell command that receives the event data
- strActionData: the action data for the event processed
- fContinueSearchForMatches: indicate if to continue search for events
- fRegexMatch indicate if the matching of the parameters in the selection fields.

The Event log console

The Event log console displays the events sent to the product. It is a query based report that can filter the events based on a list of parameters, and can perform actions on the events.

Check-IT Network Management System: EventLogConsoleReport

Administration Discovery Jobs Discovery Reports Monitoring Jobs Monitoring Reports Maps Monitoring Graphs Help

Query Fields:

From Time: 21/08/10 08:09:03 TO Time: 22/08/10 04:09:03 Origin: Sub Origin: Severity: Event Type:

Actions:

Owner: admin Severity Type: Normal Lines Limit: 500

Query Results:

EVENT_TIME	OWNER	ORIGIN	SUB_ORIGION	EVENT_TYPE	SEVERITY_COLOR	NAME	SUB_NAME	MESSAGE	Select
Sat Aug 21 18:10:35 IDT 2010	null	WatchDog	null	StatusChange	Severe	ICMPDataCollectionAgent 1	null	WatchDog Change Status for:ICMPDataCollectionA	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	172.16.1.37	null	IP: 172.16.1.37 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	172.16.1.34	null	IP: 172.16.1.34 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	192.168.80.1	null	IP: 192.168.80.1 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	172.16.0.1	null	IP: 172.16.0.1 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	127.0.0.1	null	IP: 127.0.0.1 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:31 IDT 2010	null	MonitoringAgent	SNMP	StatusChange	Normal	172.16.1.33	null	IP: 172.16.1.33 Interface:pNA+ Loopback Driver changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:31 IDT 2010	null	MonitoringAgent	SNMP	StatusChange	Normal	172.16.1.33	null	IP: 172.16.1.33 Interface:Xerox Embedded Ethernet Controller, 10/100 Mbps, v1.0, UTP RJ-45, connector A1, Auto changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:08:25 IDT 2010	null	WatchDog	null	StatusChange	Normal	DbScheduler1	null	WatchDog Change Status for DbScheduler1 to Running	<input type="checkbox"/>
Sat Aug 21 18:08:09 IDT 2010	null	WatchDog	null	StatusChange	Normal	SNMPDataCollectionAgent 1	null	WatchDog Change Status for:SNMPDataCollection	<input type="checkbox"/>

in a console based , where assigning of events, and removing of events can be performed.

To enter to this report go to: Administration → Event Log → Event log console

The information presented by this report is taken from the tblEventLog. The Panel shows the events based on the following query fields:

- Start Date & Time
- Stop Date & Time
- Event Origin
- Event Sub-origin
- Event Type
- Event Owner
- Event Status

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

Actions:

- refresh – performs the query again
- Update Owner
- Update Severity
- Delete Event
- Add Event
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of rows returned.

The Event log history report

The Event log history report displays all the events sent to the product. Opposed to the ones listed in the console, the events can not be removed from this report, so that all events are displayed.

"Check-IT Network Management System":EventLogHistoryReport

Administration Discovery Jobs Discovery Reports Monitoring Jobs Monitoring Reports Maps Monitoring Graphs Help

Query Fields:

From Time: 21/08/10 08:18:37 Origin: Sub Origin: Severity: Event Type:

TO Time: 22/08/10 04:18:37 Owner:

Actions:

Lines Limit: 600

Query Results:

EVENT_TIME	OWNER	ORIGIN	SUB_ORIGION	EVENT_TYPE	SEVERITY_COLOR	NAME	SUB_NAME	MESSAGE	Select
Sat Aug 21 18:10:35 IDT 2010	null	WatchDog	null	StatusChange	Severe	ICMPDataCollectionAgent1	null	WatchDog Change Status for:ICMPDataCollectionAgent1 to:Down	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	172.16.1.37	null	IP: 172.16.1.37 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	172.16.1.34	null	IP: 172.16.1.34 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	192.168.80.1	null	IP: 192.168.80.1 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	172.16.0.1	null	IP: 172.16.0.1 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:33 IDT 2010	null	MonitoringAgent	ICMP	StatusChange	Normal	127.0.0.1	null	IP: 127.0.0.1 changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:31 IDT 2010	null	MonitoringAgent	SNMP	StatusChange	Normal	172.16.1.33	null	IP: 172.16.1.33 Interface: p4+ Loopback Driver changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:09:31 IDT 2010	null	MonitoringAgent	SNMP	StatusChange	Normal	172.16.1.33	null	IP: 172.16.1.33 Interface: Xerox Embedded Ethernet Controller, 10/100 Mbps, v1.0, UTP RJ-45, connector A1, Auto changed its status to:UP	<input type="checkbox"/>
Sat Aug 21 18:08:25 IDT 2010	null	WatchDog	null	StatusChange	Normal	DbScheduler1	null	WatchDog Change Status for:DbScheduler1	<input type="checkbox"/>
Sat Aug 21 18:08:09 IDT 2010	null	WatchDog	null	StatusChange	Normal	SNMPDataCollectionAgent1	null	WatchDog Change Status for:SNMPDataCollectionAgent1 to:Running	<input type="checkbox"/>
Sat Aug 21 18:08:07 IDT 2010	null	WatchDog	null	StatusChange	Normal	LoggingEventlogDaemon1	null	WatchDog Change Status for:LoggingEventlogDaemon	<input type="checkbox"/>

To enter to this report go to: Administration → Event Log → Event log History

The information presented by this report is taken from the tblEventLogHistory.The Panel shows the events based on the following query fields:

- Start Date & Time
- Stop Date & Time
- Event Origin
- Event Sub-origin
- Event Type
- Event Owner

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

Actions:

- refresh – performs the query again
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of rows returned.

The Event Actions report

The user can define actions that will be executed when an event arrives. This report list all the Event log history report displays all the eve*nts sent to the product. Opposed to the ones listed in the console, the events can not be removed from this report, so that all events are displayed.

Query Fields:

Execution Order: Origin: Sub Origin: Severity:

Description: Object Type: Object Name: Object Sub Name:

Actions:

Lines Limit:

EXECUTION_ID	EVENT_TYPE	ORIGIN	SUB_ORIGION	SEVERITY	OWNER	OBJECT_TYPE	OBJECT_ID	OBJECT_NAME	SUB_NAME	DATA_SEPARATOR	DATA_CONDITION	ACTION_TYPE	ACTION_DATA	CONTINUE_ACTION	REGEX_MATCH	Select
99999	null	null	null	null	null	null	null	null	null	null	null	AddToMonitorObj	null	1	0	<input type="checkbox"/>
99999	null	null	null	null	null	null	null	null	null	null	null	AddToConsoleObj	null	1	0	<input type="checkbox"/>
99999	null	null	null	null	null	null	null	null	null	null	null	WriteToEventlog	null	1	0	<input type="checkbox"/>

To enter to this report go to: Administration → Event Log → Event Actions

The information presented by this report is taken from the tblEventActions. The Panel shows the events based on the following query fields:

- Execution order

- Event Origin
- Event Sub-origin
- Event Type
- Event Owner
- Description
- Object Type
- Object Name
- Object Sub-type

i The values entered in the query fields can be prefixes. To enter a prefix or a wild character write the string you want with % at its end or at the location of the wild character .

Actions:

- refresh – performs the query again
- Add Action
- Update Action
- Print to CSV – put query result into a CSV file for later processing
- Lines limit – sets the max number of raws returned.

RRD Daemon

RRD is a Round Robin Database. It is an open-source database used to collect monitoring data with a method that limits the size of the collecting database, and with methods to accumulate the data in methods defined by the database creator.

An explanation about the database can be found in: <https://rrd4j.dev.java.net/>

The RRD Daemon is a process which goes over the log files and builds RRD databases based on RRD definitions supplied to it, and the Logging data collected by the logging daemon.

The RRD Daemon should be located in a place close to the Logging Daemon log files as it reads them sequentially.

The RRD databases configuration files

The RRD Daemon reads as it goes up a set of RRD definition files that define which Log records to process, and what RRD daemon to build.

You must be familiar with the RRD concept to understand the logic behind it.

The RRD definition files are located in the conf/rrd/ directory.

A sample of an RRD configuration file

```
<RRDDefinitionEntries>

  <IniFile>/opt/jilroy/checkit//conf/config.prop</IniFile>

  <RRDDefinitionEntry>

    <Name>JILROY_ICMP</Name>

    <DBName>@strIpAddr@.icmp</DBName>

    <VerticalLabel>resp time (ms)</VerticalLabel>
```

```
<MonitoringCycleTimeoutSec
INIFILE="ICMP.Monitoring_cycle_timeout_in_sec">300</MonitoringCycleTimeoutSe
c>

<Parameters>

    <Parameter>

        <Name>Average</Name>

        <DisplayName>Average</DisplayName>

        <GraphType>Area</GraphType>

        <Color>255,0,0</Color>

        <Type>GAUGE</Type>

    </Parameter>

    <Parameter>

        <Name>Max</Name>

        <DisplayName>Max</DisplayName>

        <GraphType>Line</GraphType>

        <Color>100,0,0</Color>

        <Type>GAUGE</Type>

    </Parameter>

    <Parameter>

        <Name>Median</Name>

        <DisplayName>Median</DisplayName>

        <GraphType>Line</GraphType>

        <Color>0,255,0</Color>

        <Type>GAUGE</Type>
```

```
</Parameter>

<Parameter>

    <Name>StandardDeviation</Name>

    <DisplayName>Standard Deviation</DisplayName>

    <GraphType>Line</GraphType>

    <Color>0,0,255</Color>

    <Type>GAUGE</Type>

</Parameter>

</Parameters>

<Archives>

    <Archive>

        <Description>Raw Data</Description>

        <TimePeriodCount>1</TimePeriodCount>

        <Size>100</Size>

        <Type>Average</Type>

    </Archive>

    <Archive>

        <Description>Hourly Db</Description>

        <TimePeriodCount>12</TimePeriodCount>

        <Size>100</Size>

        <Type>Average</Type>

    </Archive>

    <Archive>
```

```
        <Description>Dayly Db</Description>

        <TimePeriodCount>288</TimePeriodCount>

        <Size>100</Size>

        <Type>Average</Type>

    </Archive>

</Archives>

</RRDDefinitionEntry>

</RRDDefinitionEntries>
```

RRDDefinitionEntries

A Wrapper tag that contains one or more RRDDefinitionEntry entries.

IniFile

This tag contains the location of a prop file, that can be used to configure parameters in the RRDDefinitionEntry elements.

RRDDefinitionEntry

This tag contains a single RRD Definition entry used by the RRD daemon. It contains 2 sets of tags, as defined in the definition of the RRD database (Parameters, Archives) and some additional parameters.

Name

This tag defines the name of the RRD entry. This name is the name of the log entry written to the log files.

This name should match the name of the SNMP OID Nick-name when collecting snmp data using the

SNMP data collector agent.

DBName

The Name of the file of the RRD database. The name can contain parameters, such as the Ip address of the monitored node, which will be replaced by the right value. Parameters are enclosed by @..@.

VerticalLabel

The Vertical label that will appear in the graphs generated by this RRD database.

MonitoringCycleTimeoutSec

The frequency of receiving logging information that will feed this database

Parameters

A tag that contains the set of parameters collected in this RRD database

```
<Parameters>

  <Parameter>

    ...

  </Parameter>

  ...

  <Parameter>

    ...

  </Parameter>

</Parameters>
```

Parameter

This tag defines a parameter for the RRD database.

```
<Parameter>

  <Name>Average</Name>
```

```
<DisplayName>Average</DisplayName>

<GraphType>Area</GraphType>

<Color>255,0,0</Color>

<Type>GAUGE</Type>

</Parameter>
```

Name

The name of the parameter

Display Name

The display name of the parameter appearing in the legend.

GraphType

The way the parameter will appear in the graph (Area|Line)

Color

The color of the parameter in the graph.

Type

The type of the parameter (Counter, GAUGE,...) see valid values in the RRD documentation.

Archives

A tag that contains the set of archives collected in this RRD database

```
<Archives>

    <Archive>

        ...

    </Archive>

</Archives>
```

```
</Archive>

...

<Archive>

...

</Archive>

</Archives>
```

Archive

This tag defines an archive for the RRD database.

```
<Archive>

  <Description>Raw Data</Description>

  <TimePeriodCount>1</TimePeriodCount>

  <Size>100</Size>

  <Type>Average</Type>

</Archive>
```

Description

This tag is used as a documentation tag

TimePeriodCount

This archive will accumulate this number of raw measurements. The time of an entry is the value specified in the `MonitoringCycleTimeoutSec` tag multiplied by this value.

Size

The number of entries kept in this archive

Type

The type of value kept in an entry, with regard to the source of the data. See RRD documentation.

Performance related tricks & Tips

This chapter deals with several tricks and tips that can be used to improve the performance of the monitoring platform

Initial status of an object

The product was designed to executable a job on every status change. This can be a very problematic issue when the product is running for the first time, as it does not know the initial status of the monitored elements.

If you can estimate what will be the most common status, you can set it as the default value, and in this case there will be only notification when the monitored element changes its status from the defined value.

You can set the default value in the [INSTALLDIR]/conf/config.xml file in the <MonitoringAgent> of the relevant agent:

```
<MonitoringAgent>
    <DefaultStatus>DOWN</DefaultStatus>
</MonitoringAgent>
```

The value for ICMP & SNMP agents can be UP|DOWN|UNKNOWN

Dynamic load of monitored elements in run time

The process of loading large numbers of monitored elements is a long process specially when we talk about 10Ks and more nodes.

All the monitoring agents are programmed to load the data once when they go up, and then ask for delta in the objects that should be monitored.

We can identify 2 types of load:

- A major change in the database leading to a new full load of all monitored elements and than

informing the agent about the changes. This process is also costly on the server side and should be done with caution.

- A small change which should be done fast

Major changes Delta load

Major changes in the list of monitored elements are considered changes that lead to change in more than a 1/3 of the monitored elements.

This is done when the database table `tlbObjectLastMonitoringTime` is updated. This is done automatically when there are changes in the objects table using the `NetworkDiscovery` utility or when running the `MonitoringInventory` script.

If you do not want to update the table run the 'inventory' script instead and do the operations as described in the next section

Minor changes in Delta load

When only small changes are done to the monitoring inventory (Add|Update|Delete) you can upload the changes by adding files into the `[INSTALLDIR]/conf/monitoringupdate` directory.

The files name convention is:

Action.[descriptive name.]nAgentId

where Action can be: Add| Update | Delete

the AgentId is the ID of the agent which we want to update as found in the `conf/config.xml` under the entry:

```
<MonitoringAgent>
    <AgentId>1</AgentId>
</MonitoringAgent>
```

The content of the file should be the same as the output of the selection performed by the filter file of the specific agent type when the first line is the field names selected and the following lines are the actual values.

No field name should be specified for fields that are not given value (fields that take default value).

There can be multiple files inserted for the same agent for a given action.

The files are renamed to .old when processed.

Final Page

More Information

More information about Jilroy Software, the CheckIT product, and our other products can be found on our web site. www.jilroy.com

Contact Us

For any information or problem, request for information or extension idea related to the CheckIT or any other product, please contact one of the following

email addresses.

Sales

sales@jilroy.com

Product Management

discovery pm@jilroy.com

FTP Site

All jilroy products can be downloaded from our Web site.