C/C++ Edition
Users Guide

Version 4.2.6673.0

December 17, 2021

Contents

Chapter 1

About MyARM

MyARM is an Application Response Measurement (ARM) 4.0 compliant agent. MyARM is derived from the work of tang-IT ARM agent. As a member of The Open Group the main developers of tang-IT ARM and therefore of MyARM worked actively on the ARM 4.0 standard within the ARM working group. A deep inside knowledge of how ARM 4.0 works influenced the design and implementation of MyARM and its components.

1.1   Features

- Language bindings
    - ARM 4.0 (ARM 4.1) for C
    - ARM 4.0 (ARM 4.1) for Java
    - ARM 4.0 (ARM 4.1) for C#
    - ARM 4.0 for Python (using ARM 4.0 C binding)

- Language frameworks
    - ARM 4.0 (ARM 4.1) for C++ (using ARM 4.0 C binding)
    - QArm4 ARM 4.0 (ARM 4.1) for C++/Qt4 (using ARM 4.0 C binding)
    - ARM 4.0 shell script commands (using ARM 4.0 C binding)

- Response time measurement
    - Parent/child transaction response time correlation using ARM correlator concept
    - Possibility to associate context information (properties, metrics) to a transaction response time measurement
    - Possibility to bind a thread (id and name) to a transaction response time measurement
    - Transaction status (good, failed, etc.)
    - Measure time spent waiting for some resource to be available (waiting for a response or printer ready)
    - Associate an user with a transaction response time
    - Inherit parent user identity by using the parent correlator within the child transaction measurement
    - Automatic stopping of root transaction response time measurements for ARM 4.1 event flows using UDP messages (MyARM 4.0)

 – Discard transaction measurements if response time is less than a configured threshold
 – Discard or stop (clear) transaction measurements if response time is greater than a configured threshold

- Management

 – Ability to enable/disable response time measurements from a central administration console
 – Execute user-defined actions (e.g. send notifications, starting a new server to reduce load on existing servers, etc) upon exceeding response time thresholds
 – Ability to monitor any response time measurement regarding its execution status (GOOD, FAILED, etc)

- Analysis

 – Statistical transaction response time metrics such as mean, median and deviation
 – Visualization of transaction chains and trees
 – Calculation of response time distribution within a transaction tree regarding application or host
 – Speedup calculation of parallel or distributed algorithms
 – Transaction response time histograms
 – Real time statistics calculation and presentation

1.2 Editions

MyARM is released in *Versions* as well as in *Editions*. All editions are based on the same source and therefore carry the same version number. New versions are released as new features and enhancements are available in MyARM. This document refers to version 4.2.6673.0 of MyARM.

This brief overview summarizes different *Editions* and its features:

MyARM Community Edition
 Free to use fully compliant ARM 4.0 agent including bindings for C/C++, Java, C# and Python. Manager graphical user interface (UI) and a standalone HTTP-Server providing a modern web user interface (UI). All measured data is stored using SQLite database as backend.

MyARM C/C++ Edition
 Provides the ARM 4.0 (ARM 4.1) language binding for the C programming language. A C++ framework which uses the ARM 4.0 C language binding provides an easy to use object-oriented API. Beside the manager GUI and the web UI, command line tools are provided for batch processing of measured data stored in a supported (SQLite or MySQL) database.

MyARM Java Edition
 Provides the ARM 4.0 (ARM 4.1) language binding for the Java programming language. For analysing measured data the manager GUI and the web UI and command line tools are provided. Any measured data is stored in a supported (SQLite or MySQL) database.

MyARM C# Edition
 Provides the ARM 4.0 (ARM 4.1) language binding for the C# programming language. For analysing measured data the manager GUI and the web UI and command line tools are provided. Any measured data is stored in a supported (SQLite or MySQL) database.

MyARM Standard Edition
 Combines the C/C++ and Java Edition by providing the ARM 4.0 (ARM 4.1) language binding for the C/C++ and Java programming languages in a single edition. For analysing measured data the manager GUI, the web UI and command line tools are provided. Any measured data is stored in a supported (SQLite or MySQL) database.

MyARM Enterprise Edition
    Combines all three ARM 4.0 (ARM 4.1) language binding editions (C/C++, Java and C#) in one
    single edition. For analysing measured data the manager GUI, the web UI and command line tools
    are provided. Any measured data is stored in a supported (SQLite or MySQL) database. For high
    loads multiple database connections are used to write measured data into the atabase.

1.2.1   Add-ons

The functionality for each MyARM Edition (except MyARM Community Edition) can be extended by so-
called Add-ons. An Add-on is a special feature which is additional licensed and enhances the functionality
of the MyARM agent. The following Add-ons exists currently:

Real time statistics (RTS)

    Real time statistics are used to monitor executed transactions within a production environment to get
    in overall picture of the performance and health of the whole system including end-to-end response
    times. For detailed information please read section Real time statistics.

Runtime configuration (RTC)

    Transaction runtime configuration is used to manage and control instrumented applications remotely.
    For detailed information please read section Transaction runtime configuration.

Runtime notification (RTN)

    Runtime notifications are used to notify conditions of the running applications by executing a script
    based on transaction or RTS conditions. For detailed information please read section Runtime noti-
    fication.

Multiple database connections (MDB)

    The MySQL/MariaDB database data sink supports to open multiple connections to a server. For
    detailed information please read section MySQL database.

FastCGI web applications (FCGI)

    The MyARM web applications can be integrated into an existing web server infrastructure which
    supports the FastCGI protocol. For detailed information please read section FastCGI integration.

1.3   History

Here is a short overview of the MyARM development history.

October 2017
    Release of MyARM 4.1 introducing a new concept of an archive of MyARM binary files including
    compression with Zstandard compression algorithm to support high volumes of transaction measure-
    ments up to billions of transactions per day for particular customers.

May 2015
    Official release of MyARM 4.0 supporting disabling measurements from a central administration
    console, execution of actions (e.g. sending notifications) due to predefined response time conditions
    (e.g. response time mean is greater than 400 milliseconds).

October 2014
　　Official release of MyARM 3.1 with 64bit support for AIX and Solaris. Also Linux on ARM processors (32bit) is now supported.

March 2014
　　Official release of MyARM 3.0 including real time statistics Add-on.

October 2013
　　Release of version 3.0 of MyARM including real time statistics Add-on for particular customers.

January 2013
　　Release of version 2.1 of MyARM including new editions for single ARM 4.0 language bindings for C/C++, Java and C#.

June 2012
　　Release of version 2.0 of MyARM including web 2.0 analysis interface.

February 2011
　　Release of Version 1.4 of MyARM including support for Oracle database.

January 2010
　　Release of Version 1.3 of MyARM including support for C# and Python ARM 4.0 binding and AIX platform support.

April 2008
　　Release of Version 1.1 of MyARM including the Qt4 based Manager and Windows version.

May 2005
　　First release of MyARM.

1.4   License

THE FOLLOWING TERMS AND CONDITIONS CREATE A SOFTWARE LICENSE AGREEMENT
("**LICENSE AGREEMENT**") BETWEEN MYARM (REFERRED TO AS COMPANY) AND THE IN-
DIVIDUAL OR SINGLE ENTITY REFERRED TO HEREIN AS **"YOU"** OR **"LICENSEE"** FOR THE
"LICENSED SOFTWARE" (AS DEFINED BELOW). PLEASE READ THESE LICENSE TERMS
AND CONDITIONS CAREFULLY BEFORE INSTALLING OR USING THE LICENSED SOFTWARE.
COMPANY IS WILLING TO GRANT LICENSEE THE FOLLOWING LICENSE TO USE THE LI-
CENSED SOFTWARE ACCORDING TO THE TERMS AND CONDITIONS OF THIS LICENSE
AGREEMENT ONLY ON THE CONDITION THAT LICENSEE ACCEPTS ALL TERMS AND CON-
DITIONS IN THIS LICENSE AGREEMENT.

BY INSTALLING OR USING THE LICENSED SOFTWARE, YOU ACKNOWLEDGE THAT YOU
HAVE READ THIS LICENSE AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY
IT. IF YOU DO NOT AGREE TO ANY OF THE TERMS AND CONDITIONS BELOW, COMPANY IS
UNWILLING TO LICENSE THE LICENSED SOFTWARE TO YOU AND YOU HAVE TO DESTROY
ALL COPIES AND INSTALLATIONS OF THE LICENSED SOFTWARE. WITHIN TEN (10) DAYS
AFTER YOUR RECEIPT OF THE LICENSED SOFTWARE, RECEIVE A REFUND OF THE PUR-
CHASE PRICE. BY ASKING FOR A REFUND YOU STATE THAT YOU HAVE DESTROYED ALL
COPIES OF THE LICENSED SOFTWARE.

1.4.1   Definitions

1.4.1.1   Licensed Software

"Licensed Software" means the company software components in binary form for which Licensee has
paid the applicable per product or per module license fees together with user guides, build guides, ref-
erence manuals and other documentation accompanying such software components or otherwise made
available by company (collectively, the **"Documentation"**), any executables delivered with the software
components and any modified or updated versions of an of the foregoing made available to Licensee pur-
suant to Licensee's purchase of Upgrades and receipt of minor releases subject to companies maintenance
and support policies.

1.4.1.2   Upgrades

"Upgrades" means major releases providing bug fixes, platform updates and major product enhancements
or new features. Major releases are identified by a change in the first digit of the three digit release number
(x.0.buildid.0).

1.4.1.3   Maintenance and Support

"Maintenance and Support" means the technical support and software maintenance services on the Li-
censed Software. It includes bug fixes, platform updates and minor product enhancements as contained
in minor releases as well as email-support. Minor releases are identified by a change in the second digit
release number (1.x.buildid.0), indicating minor product enhancements. Bug fix releases are identified by
a change in the fourth digit number (1.0.buildid.x).

1.4.2   License Grants

1.4.2.1   Licenses

For each license that Licensee acquires to use the Licensed Software, Licensee will be provided with a
license key which may be provided by company or its authorized distributors or resellers. The license

key will contain information specific to the edition of Licensed Software, the type of license under which Licensee may use Licensed Software, an expire date when the license may expire and information about the node the Licensed Software may be executed. The type of license a Licensee may acquire according to this License Agreement is either a **"Single Node License"** or an **"Evaluation License"** all defined in this section of this License Agreement.

1.4.2.2   Single Node License

The terms of this section are applicable to the Licensee only if Licensee has purchased a Single Node License for the Licensed Software directly from company or a company authorized distributor or reseller as defined below. Subject to the terms and conditions of the License Agreement; Company grants to Licensee a non-exclusive, non-transferable, perpetual, limited right and license to:

1. permit to install and use the Licensed Software on a per product or per module basis on nodes the Licensee purchased a Single Node License for.

2. copy or have copied the Licensed Software as necessary for the purpose of exercising the rights granted under this section or for backup or disaster recovery purposes, provided that companies copyright notice and other proprietary rights notices are reproduced on each copy.

1.4.2.3   Evaluation License

In order to evaluate companies software components Licensee may install companies software components on a temporary basis for evaluation, non-commercial purposes only. An Evaluation License is limited to the period of time specified to Licensee by company. At the end of this period the license to use companies software components expires. With the Evaluation License company grants to Licensee a non-exclusive, non-transferable and limited right to use companies software components solely for evaluation purposes. The Evaluation License contains a timeout feature that disables its operation after its expiration. At the end of the evaluation period further use of the Evaluation License is prohibited without the purchase of a commercial license to obtain a valid license key for the Licensed Software. If Licensee does not purchase a license at the end of the evaluation period Licensee hereby agrees to permanently remove or delete the Evaluation License from all computer systems on which the Evaluation License was installed and destroy any software and documentation received.

1.4.3   License restrictions

1.4.3.1   General use limitations

All rights not specifically granted herein are retained by company. Licensee may not, nor may Licensee permit any other person or entity to use, copy, modify or distribute the Licensed Software (electronically or otherwise), or any copy, adaptation, transcription, or merged portion thereof, or the Documentation except as expressly authorized by company. Licensee may not modify or port the Licensed Software to operate on platforms other than those for which it has paid the appropriate fees. Licensee may not, nor may Licensee permit any other person or entity to, reverse assemble, reverse compile, or otherwise translate any binary forms of the Licensed Software, except to the extent applicable laws specifically prohibit such restriction. Licensee's rights may not be transferred, leased, assigned, or sub-licensed except for a transfer of the License Agreement in its entirety to

1. a successor in interest of Licensee's entire business who assumes the obligations of this License Agreement or

2. any other party who is reasonably acceptable to company, enters into a substitute version of the License Agreement and pays an administrative fee intended to cover attendant costs (new Single Node Licenses).

If Licensee uses, copies or modifies the Licensed Software or transfers possession of any copy, adaptation, transcription or merged portion thereof to any other party in any way not expressly authorized by company, all licenses under this License Agreement are automatically terminated.

1.4.3.2   Proprietary Protection

Company shall have sole and exclusive ownership of all right, title and interest in and to the Licensed Software and all modifications and enhancements thereof (including ownership of all trade secrets and copyrights pertaining thereto), subject only to the rights and privileges expressly granted to Licensee herein by company. This License Agreement does not provide Licensee with title or ownership of the Licensed Software but only a right of limited use.

1.4.4   Maintenance and Support of Licensed Software

Company or its authorized resellers shall provide Maintenance and Support on Licensed Software. Maintenance and Support services provided by company are provided in accordance with companies maintenance and support policies which are subject to change. Maintenance and support is limited to platforms listed on companies current product information which is also subject to change. Other support is available from company on the basis of special maintenance agreements to be negotiated.

1.4.5   Upgrades of Licensed Software

Company shall provide Upgrades on the Licensed Software on the basis of a separate Upgrade Subscription Agreement or on a case by case basis. The provision of Upgrades (major releases) occurs under sole discretion of company.

1.4.6   Limited Warranty, Disclaimer and Limitation of Liability

1.4.6.1   Limited Warranty

Company warrants to Licensee that the unaltered Licensed Software when used as permitted under the License Agreement and in accordance with the instructions in the Documentation will operate substantially as described in the Documentation for a period of ninety (90) days from the date of delivery (the **"Software Warranty Period"**. The Licensed Software is provided to Licensee as binary code and is for use by sophisticated individuals and company does not warrant that use of the Licensed Software will be uninterrupted or error-free, that all errors will be corrected or that use of the Licensed Software will meet Licensee's needs. Company will, at its own expense and as its sole obligation and Licensee's sole and exclusive remedy for an breach of this warranty, use commercially reasonable efforts to correct any reproducible error in the Licensed Software reported to company by Licensee in writing during the Software Warranty Period. Provided, however, that no such error correction provided to Licensee will extend the original Software Warranty Period. If company determines that it is unable to correct the error, company may upon approval by Licensee refund to Licensee the fees paid by the Licensee for the defective Licensed Software and terminate the License Agreement and all licenses granted herein. In the event Licensee does not approve of such refund and termination of the License Agreement, Licensee will be entitled to keep the Licensed Software and use it pursuant to the licenses granted herein. Provided, however, that company will not be obligated to provide Maintenance and Support for the Licensed Software that is impacted by the reported defect.

1.4.6.2   Exclusions

The limited warranty set forth above will not apply to defects resulting from or because of modifications made to the Licensed Software by anyone other than company. Misuse, failure of media not furnished by

company, operation with media, software or equipment not authorized by company in the Documentation or not meeting or not maintained in accordance with the supplier's specifications or causes other than ordinary use. The warranty set forth above will not be enlarged, diminished or affected by and no obligation or liability will arise from, company's rendering of technical advice, assistance or service in connection with Licensee's selection or use of the Licensed Software.

1.4.6.3   Disclaimer

EXCEPT FOR THE LIMITED WARRANTY SET FORTH IN PREVIOUS SECTION ABOVE, THE LICENSED SOFTWARE IS PROVIDED "AS IS", WITHOUT REPRESENTATIONS OR WARRANTIES OF ANY KIND. COMPANY SPECIFICALLY DISCLAIMS ALL OTHER PROMISES, REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ANY IMPLIED WARRANTY ARISING FROM COURSE OF PERFORMANCE OR COURSE OF DEALING.

1.4.6.4   Limitation of Liability

THE CUMULATIVE LIABILITY OF COMPANY TO LICENSEE FOR ALL CLAIMS RELATING TO THE LICENSED SOFTWARE AND THIS LICENSE AGREEMENT, INCLUDING ANY CAUSE OF ACTION SOUNDING IN CONTRACT TORT, OR STRICT LIABILITY, SHALL NOT EXCEED THE TOTAL AMOUNT OF ALL LICENSE FEES PAID TO COMPANY HEREUNDER. THIS LIMITATION OF LIABILITY IS INTENDED TO APPLY WITHOUT REGARD TO WHETHER OTHER PROVISIONS OF THIS LICENSE AGREEMENT HAVE BEEN BREACHED OR HAVE PROVEN INADEQUATE. COMPANY SHALL HAVE NO LIABILITY FOR LOSS OF DATA OR DOCUMENTATION, IT BEING UNDERSTOOD THAT LICENSEE IS RESPONSIBLE FOR REASONABLE BACKUP PRECAUTIONS. IN NO EVENT SHALL COMPANY BE LIABLE FOR ANY LOSS OF PROFITS; ANY INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES; OR ANY CLAIMS OR DEMANDS BROUGHT AGAINST LICENSEE. THIS LIMITATION UPON DAMAGES AND CLAIMS IS INTENDED TO APPLY WITHOUT REGARD TO WHETHER OTHER PROVISIONS OF THIS LICENSE AGREEMENT HAVE BEEN BREACHED OR HAVE PROVEN INEFFECTIVE. LICENSEE MAY HAVE ADDITIONAL RIGHTS UNDER CERTAIN LAWS (E.G. CONSUMER LAWS) THAT DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, OR THE EXCLUSION OR LIMITATION OF CERTAIN DAMAGES. IF SUCH LAWS APPLY, CERTAIN EXCLUSIONS OR LIMITATIONS MAY NOT APPLY TO LICENSEE; HOWEVER, ALL OTHER RESTRICTIONS AND LIMITATIONS SHALL REMAIN IN EFFECT.

1.4.7   Termination

1.4.7.1   Term

The term of this License Agreement will begin as of the date that Licensee receives the Licensed Software and will remain in effect perpetually unless terminated under this section.

1.4.7.2   Termination of Convenience

Licensee may terminate this License Agreement for any reason or for no reason by giving company thirty (30) days written notice.

1.4.7.3   Termination for Cause

Company may terminate this License Agreement if Licensee breaches its obligations hereunder. Company will effect such termination by giving Licensee notice of termination, specifying therein the alleged breach.

If the breach is curable, Licensee will have a grace period of thirty (30) days after such notice is served to cure the breach described therein. If the breach is cured within the thirty (30) day grace period, the this License Agreement will remain in effect. Otherwise, this License Agreement will automatically terminate upon the conclusion of the thirty (30) day grace period.

1.4.7.4   Effect of Termination

Upon termination of the License Agreement for any reason the following terms shall apply:

1. All rights granted under this License Agreement will immediately terminate and Licensee must immediately stop all use of the Licensed Software.

2. Licensee must destroy all copies of the Licensed Software provided to or made by or on behalf of Licensee, and will, within ten (10) days after the effective date of termination, provide company with written certification that all such copies have been returned or destroyed.

3. All provisions of this License Agreement with the exception of the licenses granted in section License Grants, Maintenance and Support obligations set forth in Section Maintenance and Support and Upgrade obligations set forth in Section Upgrades will survive termination of this License Agreement for any reason. Termination of the License Agreement will not affect Licensee's obligation to pay all amounts accrued hereunder prior to the effective date of termination.

1.4.8   Miscellaneous

1.4.8.1   Severability

If any term or provision of the License Agreement is found to be invalid under any applicable statute or rule of law, then, that provision notwithstanding the License Agreement will remain in full force and effect, and in such event, such provision will be changed and interpreted so as to best accomplish the objectives of such unenforceable or invalid provision within the limits of applicable law or applicable court decisions.

1.4.8.2   Governing Law/Forum selection

The License Agreement and the parties' rights and obligations hereunder shall be solely and exclusively construed, interpreted and enforced under and in accordance with the laws of Germany, without any reference to conflicts of law principles. Any and all disputes between the parties that cannot be amicably resolved, arising under, out of, or otherwise relating to the License Agreement, Licensed Software or any services provided by company to Licensee shall be brought and resolved solely and exclusively in the courts located in Frankfurt/Main, Germany. Both parties hereby irrevocably consent to the jurisdiction of such courts and service of process connection therewith. Any judgement rendered by such courts my be entered and enforced by any court having jurisdiction over the party against which an award is entered or its assets. Both parties hereby irrevocably waive any objections to the jurisdiction of such courts based on any ground, including without limitation improper venue or forum non convenience.

1.4.8.3   Export Law

Licensee may not use or otherwise export or re-exports the Licensed Software except as authorized by German law and the laws of the jurisdiction in which the Licensed Software was obtained.

1.4.8.4   English Language

The parties hereto have expressly required that the present License Agreement be drawn up in the English language.

1.4.9   Third party licenses

1.4.9.1   Apache Software Foundation License

The licensed software includes software (APR) licensed under the Apache License Version 2.0 (see ASF-LICENSE).

1.4.9.2   GNU Lesser General Public License

The licensed software includes software (libmariadbclient) bdeveloped by SkySQL Corporation Ab and licensed under the GNU Lesser General Public License (LGPL) (see LGPL-LICENSE).

1.4.9.3   Public domain license

This product includes software (MD5) developed by L. Peter Deutsch of Aladdin Enterprises.

1.4.10   Copyright notice

THE LICENSED SOFTWARE IS PROTECTED BY COPYRIGHT LAW AND INTERNATIONAL TREATY. UNAUTHORIZED REPRODUCTION OR DISTRIBUTION IS SUBJECT TO CIVIL AND CRIMINAL PENALTIES.

Copyright 2005-2021 MyARM. All Rights Reserved.

Chapter 2

Getting started

2.1 Introduction

This document describes the installation of MyARM. First the requirements and environment needed to deploy MyARM for C/C++, Java, C# and Python are described. Followed by a detailed description how to install MyARM on your system and finally setting up the required license key.

2.1.1 Requirements

The following conditions must be fulfilled to deploy the MyARM agent and you need at least 160 MB (enterprise edition) of free disk space. MyARM supports 32-bit and 64-bit environments. If you need support for any other operating system, platform, database or compiler feel free to contact us.

- Intel i586 or higher CPU (Pentium, AMD), ARMv7 or higher

- Linux kernel 2.4.x, 2.6.x or 4.x

- glibc 2.13.x or higher.

- 32bit (ARMv7, x86) and 32/64-bit (amd64) support

2.1.1.1 Databases

1. MySQL 5.0-5.6

2. MariaDB 5.5 or 10.x

3. SQLite3 (file based)

2.1.1.2 Compiler

1. Any ANSI-C/C++ compliant C/C++ Compiler.

2. Java 1.5 RTE or higher

3. C#: Mono 3.8 or higher

2.1.2 Naming of MyARM archives

MyARM is only distributed over the net by downloading the appropriate distribution from the MyARM download page: https://myarm.com/support.html#support-dl.

The following naming scheme applies to all MyARM distribution archive files:

myarm-***version***-***edition***-linux-***arch***-***type***.tar.gz

where the parts set in ***bold and italic*** are:

version – the version of the MyARM distribution. The MyARM version is build with the following pattern: "<major>.<minor>.<buildid>.<revision>".

edition – the edition of MyARM. Either ***community*** for the community edition, ***c-cpp*** for the C/C++ edition, ***java*** for the Java edtion, ***csharp*** for the C# edtion, ***standard*** for the standard edition or ***enterprise*** for the enterprise edition.

arch – the architecture (CPU) the distribution was build for. Currently

- ***armv7*** for ARMv7 CPU based systems
- ***x86*** for Intel/AMD 32bit CPU based systems
- ***amd64*** for Intel/AMD 64bit CPU based systems

type – the distribution type. Either ***agent*** containing a libraries and programs needed to deploy instrumented applications, ***tools*** containing libraries and programs to analyse measured data, ***docs*** containing the documentation and manual pages of MyARM programs or ***sdk*** containing header files and examples how to use MyARM or to instrument applications with ARM. A type of ***bin*** is an archive containing the all above described archives for easy downloading.

2.2 Installing the MyARM distribution on a linux workstation

2.2.1 Install directory

MyARM was build using the /opt/myarm directory and if used shared libraries are found automatically. We recommend to install the MyARM distribution under the directory /opt as this is the standard for programs from third-party vendors. The install procedure installs MyARM into a parent directory called myarm-4.2.6673.0. All you have to do after an installation is to create a symbolic link from /opt/myarm to the current version of MyARM you want to use:

```
# cd /opt
# ln -s myarm-4.2.6673.0 myarm
```

Therefore its easy to change to a new version of MyARM.

2.2.2 Archive extraction

All files and directories belonging to the MyARM distribution reside under a common parent directory named myarm-4.2.6673.0. The MyARM distribution is divided into several parts. Therefore a two step extraction exists:

1. extract the MyARM distribution archive into a temporary directory mostly in your home directory.

2. install the appropriate MyARM parts onto your system using the provided installation script.

If you have downloaded a MyARM standard edition for linux for Intel 64-bit you have an archive named:

```
myarm-4.2.6673.0-standard-linux-amd64-x86-bin.tar.gz
```

The following command sequence will extract the archive in a new directory named `myarm-4.2.6673.0` and starts the installation process with the provided installation script:

```
# tar xvzf myarm-4.2.6673.0-standard-linux-amd64-x86-bin.tar.gz
# cd myarm-4.2.6673.0
# ./install.sh
```

Next section will guide you through the installation script. If you want to extract and install MyARM manually skip it.

2.2.3   Installing

Executing the provided install script will present you the following greeting page. It asks if you want to proceed with the installation. Enter 'I' or hit return and you will be prompted to answer few questions about which parts of MyARM you want to install.

```
MyARM Standard Edition 4.2.6673.0                          17.12.2021
=====================================================================

Welcome to the MyARM Standard Edition 4.2.6673.0 installation process.

MyARM - an Application Response Measurement Version 4.0 compliant agent!

This install script will guide you to install MyARM on your system.

MyARM is split into the following different parts:
 1. Agent components and files needed to run ARM instrumented
    applications.
 2. Tool components and files needed to analyse recorded ARM data.
 3. Documentation for MyARM.
 4. SDK files needed to instrument own applications.
 5. Contributions to MyARM (Apache HTTPD mod_arm4, python Arm4Module,
    QArm).

This script will now ask you which of these parts should be installed.
It will also ask you for your MyARM license key which you should have
received via email.

If you want to install MyARM manually note that there is an archive
named myarm-4.2.6673.0-linux-amd64-x86-extlibs.tar.gz which is needed by
the agent and tools components of MyARM.

I)nstall MyARM
q)uit script
What do you want to do (for default in UPPER case hit return) [I/q]?
```

Installation script greeting

You will be asked to install:

1. the agent part; needed to deploy instrumented applications

2. the tools part; needed to analyse the measured data

3. the documentation part

4. the SDK part (needed to instrument own applications)

5. the contribution part

6. the license key you received via email

7. the directory where to install MyARM to

8. creating a symbolic link to easily switch between MyARM versions

Answering all questions with yes is shown in the following snippet:

```
I)nstall MyARM
q)uit script
What do you want to do (for default in UPPER case hit return) [I/q]? i

Install agent part [Y/n]? y
Install tools part [Y/n]? y
Install documentation [Y/n]? y
Install SDK part [Y/n]? y
Install contribution part [Y/n]? y

Where do you want to install MyARM [/home/ruppert/myarm-4.2.6673.0]?
Enter your license key: 4fd89ee566b183803ae6b083a7b687cdfd4c78dc
Create a symbolic link '/home/ruppert/myarm' to
  '/home/ruppert/myarm-4.2.6673.0' [Y/n]? y
```

Installation questions answered

After answering all questions the install script copies all files to the destination directory:

```
Installing agent to /home/ruppert/myarm-4.2.6673.0
Installing tools to /home/ruppert/myarm-4.2.6673.0
Installing documentation to /home/ruppert/myarm-4.2.6673.0
Installing sdk to /home/ruppert/myarm-4.2.6673.0
Installing contributions to /home/ruppert/myarm-4.2.6673.0
Writing license key to file /home/ruppert/myarm-4.2.6673.0/conf/myarm.key
Setting templates/sqlite3.conf as default config (symbolic link to myarm.conf)
Creating directory /home/ruppert/myarm-4.2.6673.0/var
Eetting read/write permissions for /home/ruppert/myarm-4.2.6673.0/var
Creating symbolic link '/home/ruppert/myarm' to '/home/ruppert/myarm-4.2.6673.0'

Installation completed!

You will find MyARM in your "/home/ruppert/myarm-4.2.6673.0" directory. MyARM
log files will be found in the "/home/ruppert/myarm-4.2.6673.0/var" directory.

Source the script in /home/ruppert/myarm-4.2.6673.0/scripts/setup.sh to setup
the needed environment to execute instrumented applications
and/or MyARM tools. Execute the following line in the shell:

cd /home/ruppert/myarm-4.2.6673.0; . scripts/setup.sh

If you use MySQL or MariaDB database use the following command
to create all needed databases and tables for the first time
(note you need the appropriate database privileges):

myarminitdb --create
```

Installation finished

Chapter 3

Agent

3.1 Overview

The MyARM agent implements the ARM version 4.0 interface standard of The Open Group. Figure 3.1 gives an overview how ARM is used:
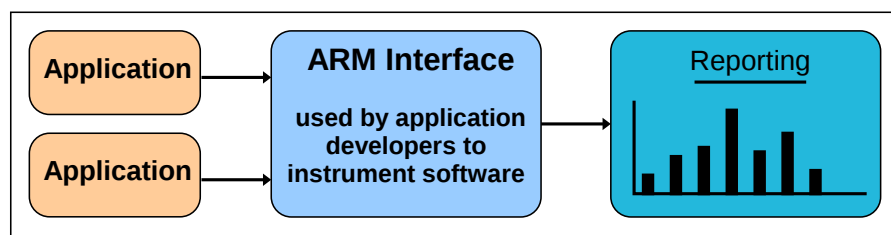
Figure 3.1: ARM overview

- Applications use the ARM interface to measure response times.

- An ARM interface agent implementation measures the response times and processes or stores the results for analysis.

- ARM implementations provide tools to analyse response times and present the results to the user.

This chapter describes only the MyARM agent specifics and not the ARM interface standards. For a detailed description of the ARM interface standard and its concepts please read the appropriate Open Group standard documents [ARM4C] and [ARM4J].

The main objectives of the MyARM ARM 4.0 agent implementation are:

- Minimal influence to the instrumented application using a threading pipeline (Figure 3.4) for processing the ARM data within the instrumented application

- Decoupled delivery of transaction data to the storage part through a dedicated transaction data collection process (daemon) per host

- Ability to deliver transaction data to the storage part directly from the application, if there is no chance to run a separate daemon on the host the application runs on

- Modular architecture for transport, storage and retrieval of measured data

- Choice between different storage databases (SQLite, MySQL or MariaDB)

3.1.1   Agent architecture

The MyARM agent architecture is designed to be flexible, efficient and robust. Figure 3.2 gives an overview:
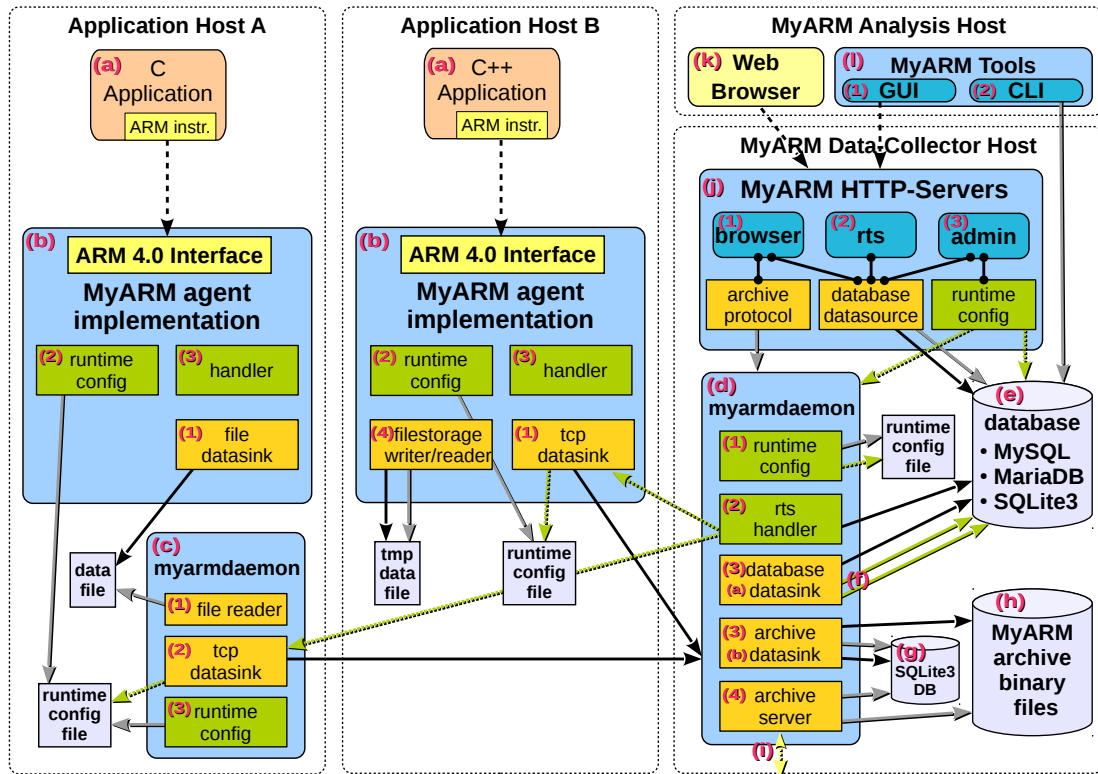
Figure 3.2: MyARM overall architecture
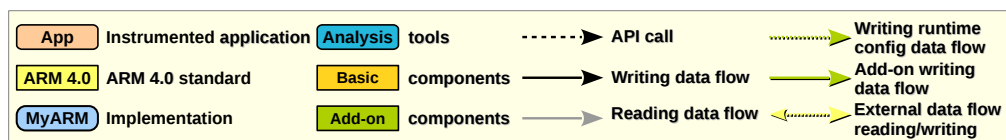
Figure 3.3: MyARM overall architecture legend

(a) The ARM instrumented C/C++ applications uses the ARM 4.0 interfaces to measure ARM transactions

(b) The MyARM agent implements the ARM 4.0 interfaces and is linked (dll) against the instrumented application

   (1) The MyARM agent implementation uses a datasink (backend) concept to store the measured ARM data. The following different kinds of datasinks can be used.

     file datasink

       The file datasink writes the measured ARM data to a file which will be collected by a `myarmdaemon`

     tcp datasink

       The tcp datasink writes the measured ARM data over a TCP/IP connection to a `myarmdaemon`

(2) The runtime configuration is used to change the behaviour of a running instrumented application. The runtime configuration is distributed to the instrumented application through a `myarmdaemon` process which receives runtime configuration data through a TCP/IP connection from the `myarmadmin` web application

(3) A handler uses runtime configurations to adopt the behaviour of the MyARM agent

(4) Filestorage component to write/read ARM data to/from a temporary file if the datasink is currently not able to store the data (TCP/IP connection down)

(c) The `myarmdaemon` is configured

(1) to read in any ARM data files which were written by the file datasink of an instrumented application

(2) It will forward the ARM data to a connected `myarmdaemon` configured as a TCP server

(3) reading runtime configurations from disk when new configuration were stored by the TCP datasink

(d) The `myarmdaemon` collects and aggregates ARM data received and provides interfaces for MyARM web applications

(1) receives runtime configuration data from the `myarmadmin` web application using a TCP/IP connection, stores the data into a local file and distributes the runtime configuration to connected `myarmdaemon` processes

(2) aggregates the received ARM data into so-called real time statistics (rts handler)

(3) stores any received ARM data depending on configuration

(1) into a database

(2) or in the filesystem based MyARM archive

(e) Database support is provided for SQLite3, MySQL and MariaDB

(f) Multiple database connections can be opened to the MySQL or MariaDB database for high transaction loads

(g) MyARM archive SQLite3 database used to store ARM metadata (such as application and transaction names)

(h) filesystem based MyARM archive to store ARM transaction measurements efficiently and a scalable search using by the `myarmbrowser` web application

(i) CSV export TCP/IP interface to export ARM transaction measurements exactly once

(j) MyARM HTTP server started by the `myarmdaemon`

(1) `myarmbrowser` for individual case analysis

(2) `myarmrtsbrowser` and `myarmrtsmonitor` for statistical analysis and 24 hours monitoring

(3) `myarmadmin` to configure the MyARM system globally

(k) A standard web browser can be used to access all MyARM web applications

(l) MyARM applications

(1) `myarmmanager` provides a simple graphical user interface to access all MyARM web applications

(2) Command line tools (such as `myarmquery`) are able to connect to the database directly (besides SQLite3 database)

3.1.2   Agent data processing

The MyARM agent implementation measures and processes timing information about the instrumented application. To minimize the interference the MyARM agent implementation uses a threading pipeline as shown in Figure 3.4 to process the measured data.
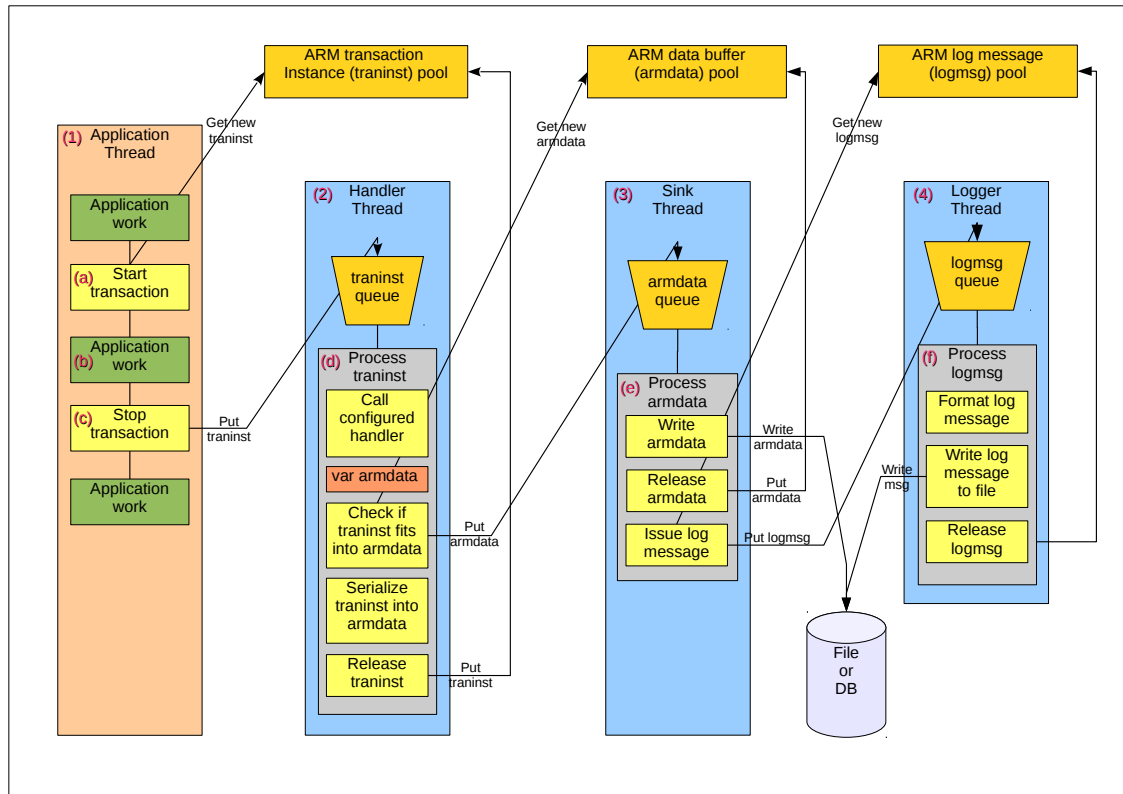
Figure 3.4: Agent data processing thread pipeline

Threads are shown in the light orange (application thread) and blue (MyARM agent threads) boxes marked with the following digits in parenthesis:

(1) *Application thread* executing application code which is instrumented with calls to the ARM 4.0 interface API.

(2) *Handler thread* which will process by various so-called handler) and serialize the measured information into a so-called ARM data buffer.

(3) *Sink thread* which will export the measured data to a destination (database, file or daemon using a TCP/IP connection)

(4) *Logger thread* which will write any log message which is generated by the MyARM agent implementation to a file or logging service (syslog, eventlog).

A typical ARM response time measurement (ARM transaction) is shown in Figure 3.4 with the following letters in parenthesis:

(a) Within the *Application thread* the ARM 4.0 API is called to start a transaction in the ARM agent implementation. The MyARM agent will get a so-called transaction instance from its internal transaction instance memory pool.

(b) The application will continue its work which will be measured by ARM.

(c) After the application has finished its work it will call the ARM 4.0 API indicating that the current ARM transaction has finished by calling the stop transaction function (C) or method (Java). The MyARM agent will record timing and provided context information within the transaction instance object obtained within (a). Then it will put this transaction instance in the queue of transaction instances of the *Handler thread*.

(d) The *Handler thread* will be notified that there is a transaction instance in its queue and will process it:

- It will get this transaction from the queue

- Call any configured handler and if a handler discarded or postponed transaction instance the processing is stopped here

- Check if the transaction instance fits into the current so-called ARM data buffer. If not, the current ARM data buffer is full and is put into the queue of the *Sink thread*.

- Check if there is a valid ARM data buffer and if not (ARM data buffer was full or its the first transaction instance) it will get a new ARM data buffer from the ARM data buffer memory pool.

- Now we have a valid ARM data buffer and will serialize the contents of the transaction instance into the ARM data buffer.

- The last step is to release the transaction instance and to put it in the transaction instance memory pool again.

(e) The *Sink thread* will be notified that there is an ARM data buffer in its queue and will process it by writing the data into the configured datasink:

- It will get the ARM data buffer from the queue

- Write the contents of the data buffer to the configured datasink (file, tcp or database)

- Release the ARM data buffer by putting the ARM data buffer back to its memory pool.

- Finally, it will issue a log message containing the number of processed ARM data entries (application and/or transactions). To do this, a new logmsg object is obtained from the log message memory pool. It will be filled with all relevant information and put into the *Logger thread* queue.

(f) The *Logger thread* will be notified that there is a log message in its queue:

- It will get the logmsg object from the queue.

- Format the message to a readable string.

- Write the message to the configured logging destination.

- Release the logmsg object by putting it into the log message memory pool back.

3.1.3 Agent features

The MyARM agent implements the full set of features defined by the ARM 4.0 interface specification which can be summarized as follows:

- Transaction response time measurement with micro- or nanosecond granularity (platform dependent)

- Supports implicit (start/stop) and explicit (report) response time measurement

- Transaction parent/child relationship correlation

- Application measurement support (application running time)

- Up to 20 name/value context properties (*name=value*) for applications and transactions

- URI support for transactions

- Transaction metrics like counters, gauges, numeric IDs and simple strings

- Error handling support on API level

- Supports finer granularity for transaction measurements

  - measuring time spent waiting for some resource to be available (blocked transactions)
  - bind a thread to a transaction measurement

- Supports single- and multi-threading environments

3.1.4 Agent overhead

Measuring response times within an application is not for free. Getting time stamps, process context data, transfer measured data to external hard drive or databases needs addtional processing time. To get an overview of the performance impact using MyARM read the appendix Agent overhead.

3.2 Bindings

This section describes the ARM language bindings supported by MyARM. It provides a more detailed description of the MyARM set of ARM features.

3.2.1 ARM 4.0 C Binding

The ARM 4.0 C Binding published by The Open Group defines a C API to instrument applications to measure performance data. MyARM implements this API and can be used with any standard conforming instrumented application. This section does not describe the C API as it is published by The Open Group (see [ARM4C] for an API description), it describes the concrete implementation and special characteristics of MyARM.

3.2.1.1 C versus C++

The ARM 4.0 standard defines only a C API which can be used in C++ as well. The reason why there is no real C++ API is due to the fact how ARM is deployed within a system.

One major goal of ARM is that any application which is instrumented with a certain version of ARM should be able to run with any installed ARM agent of that version. But currently there does not exist any real application binary interface (ABI) standard for the C++ language. A compiler vendor can choose its own model how the compiler layouts objects or mangles class/method names to linker symbol names. So different ARM implementations could have different object layouts and different linker symbol names. This problem does not arise with plain ANSI-C programs.

3.2.1.2 ARM 4.0 C++ framework

However, MyARM provides a complete ARM 4.0 C++ framework using the ARM 4.0 C binding (See https://api.myarm.com/arm40cpp/). This framework is provided as source code which can be inlined or separately compiled. Therefore the ABI problem can be solved by the user of the ARM 4.0 C++ framework.

3.2.1.3 ARM 4.0 Optional features

In the ARM 4.0 document many features defined in the standard are marked as optional and may or may not be implemented by an agent. Besides the mandatory features MyARM supports the following optional features:

- Transaction correlation with the *arm_start_transaction()*, *arm_report_transaction()* and *arm_generate_correlator()* function calls

- Error reporting support using the *arm_get_error_message()* call

- Support for adjusting the start time for a transaction using the *arm_get_arrival_time()* call and the ARM_SUBBUFFER_ARRIVAL_TIME sub-buffer

- Support for blocking transactions using the *arm_block_transaction()* and *arm_unblock_transaction()* calls

- Binding threads to a transaction using *arm_bind_thread()* and *arm_unbind_thread()* or the ARM_FLAG_BIND_THREAD

- ARM metrics are fully supported using the sub-buffers ARM_SUBBUFFER_METRIC_VALUES and ARM_SUBBUFFER_METRIC_BINDINGS

- Full support of ARM identity properties using the sub-buffers ARM_SUBBUFFER_APP_IDENTITY and ARM_SUBBUFFER_TRAN_IDENTITY

- Full support of ARM context properties using the sub-buffers `ARM_SUBBUFFER_APP_CONTEXT` and `ARM_SUBBUFFER_TRAN_CONTEXT`

- Support of diagnostic detail information for failed or aborted transactions using the `ARM_SUBBUFFER_DIAG_DETAIL` sub-buffer

- Support of ARM 4.1 `ARM_SUBBUFFER_BLOCK_CAUSE` sub-buffer

- Basic support of ARM 4.1 `ARM_SUBBUFFER_MESSAGE_RCVD_EVENT` sub-buffer indicating end of flow measurement. In conjunction with `ARM_FLAG_TRACE_REQUEST` flag which indicates to wait for event flow auto response messages (see Event flow auto response handler) it is possible to automatically stop one-way messages

3.2.1.4   ARM 4.0 unsupported features

As described in the previous section many features are optional. Currently MyARM does not provide the following features:

- Turning tracing on and off using the `ARM_FLAG_TRACE_REQUEST` flag. MyARM by default traces all transactions.

  If a `ARM_FLAG_TRACE_REQUEST` flag is encountered MyARM enables the event flow auto response feature (see Event flow auto response handler).

3.2.1.5   Agent details

3.2.1.5.1   C Agent shared library architecture

The main part of the ARM agent is the shared library (object) which implements the ARM 4.0 C API. Each ARM instrumented application is linked against this library. The library base name is `arm4` and for example `libarm4.so` is used under Linux platforms, whereas `libarm4.dll`, is used on Windows platforms.
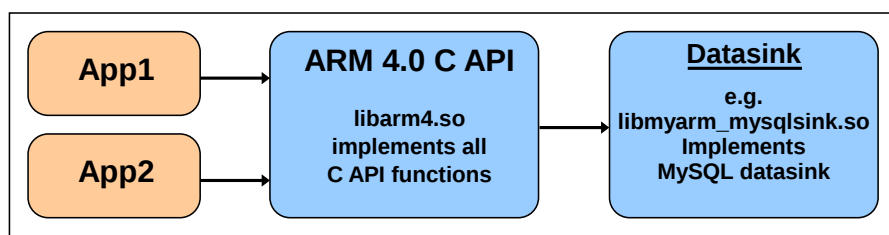
Figure 3.5: Agent shared library overview

Figure 3.5 also shows that the `libarm4.so` library in turn uses different datasink shared libraries which are loaded at runtime according to the current configuration. Here the `mysql` datasink library `libmyarm_mysqlsink.so` is shown which stores ARM data which originating in the `libarm4.so` library into a MySQL database.

3.2.1.5.2   Multi-threading versus Single-threading

The ARM 4.0 standard states that any ARM implementation has to be thread-safe. The MyARM implementation uses threads of their own to process measured data (Figure 3.4). Therefore if you use MyARM in a single-threaded program it is not really single-threaded but the interference between the instrumented application and the MyARM agent is reduced to a minimum regardless which configuration is used.

3.2.1.5.3   Response times

The ARM standard defines response times measured in nanoseconds. Not all systems today provide such a high granularity of time stamps. Table 3.1 shows on which platform MyARM provides which granularity of response time measurements.

PLATFORM	GRANULARITY	DESCRIPTION
Linux (x86, amd64)	nanoseconds	using `clock_gettime()` system call
Windows (x86, amd64)	microseconds	using `GetSystemTimeAsFileTime()` system call

Table 3.1: **Response time granularity**

3.2.1.5.4   Return codes

The ARM 4.0 C Binding describes that a negative return code for any of the API calls indicates an error, a zero return code a success and a positive return code a warning. The MyARM specific return codes are documented in the agent reference appendix.

3.2.1.5.5   Overhead

An overview of the performance impact by using the MyARM ARM 4.0 C Binding is documented in the appendix ARM 4.0 C Binding agent overhead.

3.3 Concepts

All MyARM supported ARM 4.0 language binding agents and analysis tools use the following concepts.

3.3.1 Configuration

The adaption of the MyARM agent to the current hard- and software environment is done using a set of configuration files. Each file contains name-value pairs, so-called configuration properties. For example setting the `basic.time.use_utc` property to `true` will instruct MyARM to show time information only in UTC.

```
basic.time.use_utc = true
```

Configuration 3.1: **Use UTC time**

For more and detailed information read the configuration chapter.

All MyARM agents and tools are using the MYARM_CONFIG_URL environment variable to get the configuration file.

3.3.2 Logging

The ARM standard is designed to transparently plugin into an instrumented application. Also, the ARM agent should never interfere with the application. The MyARM ARM agent uses log messages to report the current configuration, state changes, warnings and errors. The log messages can be written to different kinds of output media such as normal files, Windows event log or Unix syslog.

MyARM supports the following levels of log messages:

error
    Error log messages to report errors

warning
    Warning log messages to report some unusual situations

status
    Status log messages to report state changes within the ARM agent

info
    Only informational messages

config
    Logs the configuration used by the ARM agent

For more information read the section "Configuring Log facility".

3.3.3 ARM data buffer

When executing an instrumented application measured and recorded ARM data is serialized and written to a so-called ARM data buffer. Therefore such an ARM data buffer can contain any kind of data like transaction response time measurements, log messages or even other kind of data like CPU utilization.

An ARM data buffer has a defined size in bytes which can be specified by a configuration property. Avoiding memory shortage due to a high rate transaction measurements, a maximum number of allocated ARM data buffers is defined. For more detailed information about configuring the ARM data buffer concept read the configuration properties section.

3.3.4   Handler

A handler is a technical concept for implementing filters, monitors or calculators for transaction measurements. It is used to process a measured transaction instance according to the current runtime configuration or performs additional tasks such as sending an auto response message for event flow measurements.

For a detailed description of all features implemented by handler read section handler.

3.3.5   Datasink

The datasink concept is used to transport and store any kind of measured and recorded ARM data in a consistent way. Datasinks within MyARM are plugins which are loaded during initialization phase according to the current configuration.

With this concept it is possible to use a chain of datasinks which will:

1. store any measured data onto disk (filesink)

2. read the measured data from the disk and send it to a central server using TCP/IP (tcpsink)

3. finally storing the measured data into a MySQL database (mysqlsink).

For a detailed description of all available datasinks read section datasinks.

3.3.6   Datasource

The datasource concept defines a consistent interface to read in and manage recorded ARM data stored in a database. Like datasinks they are implemented as plugins which are loaded during runtime. Each analysis tool uses this datasource concept to retrieve the measured ARM data from the database. A datasource exists for each supported database as described in section Databases.

3.3.7   File storage

The file storage concept is a sub-component of the MyARM agent which is used to read and write ARM data files. The reading and writing be used separately or combined within a program. It is used to efficiently store measured ARM data onto a hard disk and retrieve the data later from the hard disk within the following programs:

`myarmdaemon`
>   uses the file storage concept to collect all measured ARM data for the whole host. The `myarmdaemon` can be started as a Windows service or an Unix daemon process to collect all measured ARM data for the whole host.

`myarmdaemon` TCP server part
>   can use the file storage concept to save any data received through TCP/IP onto disk. Therefore it is secured that any incoming data is first stored onto local hard drive before further processing. The `myarmdaemon` itself will read in the stored ARM data if its configured datasink (database or another `myarmdaemon` accepting TCP data connections) is ready to accept data.

ARM agent
>   uses this concept to temporarily store ARM data if the configured datasink is currently not available (database or `myarmdaemon` are down). Later if the datasink resource is available again it will read in the ARM data and write it to the datasink.

For detailed information about configuring the file storage concept read the configuration properties section.

3.3.8 Archive

The archive concept uses a set of MyARM binary files to store any measured ARM data. These files are created by the archive datasink and can be queried using the myarmarchive tool or by using the archive component (TCP protocol) of the `myarmdaemon`.

To efficiently store and retrieve data within the archive, files are created using time stamp information within the filename of the measured data. For example the file `transactions/myarm.info/20170801_091014.myarm` contains transaction measurements from our myarm.info demo site recorded on 1st of august 2017 at 10 minutes and 14 seconds past nine (UTC).

The archive uses a directory sub-tree to store all its data, starting at its configured root directory. Currently the archive distinguishes between the following main ARM data and uses a configureable file pattern relative to the archive root to create files for each of it:

ARM definitions
    normally, the following file pattern is used `definitions/$s_defs`, where the `$s` part is replaced by the system address of the host the definitions were created. The file pattern can be configured using the `basic.archive.definition.filepattern`.

ARM transaction measurements
    this is the most used file pattern and therefore it offers many pattern keys to create a file. The transaction file in the above example was generated by the following file pattern: `transactions/$s/%Y%m%d_%H%M%S`. This pattern can be configured by using the `basic.archive.transaction.filepattern`.

ARM threshold data
    for threshold data the following file pattern is used: `thresholds/${LOCATION}_-thresholds_%Y%m%d%H%M`. This pattern can be configured by using the `basic.archive.threshold.filepattern`.

    Note: Thresholds are only generated if `basic.archive.threshold.enable` is set to true.

For a complete list of basic archive configuration properties read section Archive configuration propertes.

3.3.9 Resource watchdog

During deployment MyARM needs some resources (mainly memory) to operate correctly. The amount of memory needed depends highly on the number and frequency of transaction measurements. However MyARM should never interfere with expend the resources of the deployed application. Therefore MyARM provides various configuration properties to limit the amount of used resources. To monitor the used resources by MyARM the resource watchdog provides a simple mechanism to log the used resources.

The resource watchdog can be separately configured for monitoring of resources of following programs:

- the `myarmdaemon`

- any instrumented application

3.3.10 Runtime configuration and notification

With the release 3.0 MyARM supports so-called runtime configuration and runtime notifications starting with 4.0 of MyARM. The runtime configuration can be created and managed by the `myarmadmin` web application and send to appropriate configured `myarmdaemon` processes.

The following runtime configurations can be used:

Real Time statistics (RTS)
 selects transaction definitions to calculate statistics and to provide them in real time

Transactions
 selects transaction definitions to filter or monitor current measurements

Notifications
 based on transaction or RTS conditions a configured notification can be issued

3.3.10.1 Real time statistics

Real time statistics are used to monitor executed transactions within a production environment to get in overall picture of the performance and health of the whole system including end-to-end response times. Real time in this sense means that performance data is available 5 to 10 minutes after current time (presence). With this data just few minutes ago it is possible to react on hard or software failures very quickly!

To use the real time statistics MyARM needs to know which transactions are key performance indicators (KPI). Such a KPI can be directly derived from the ARM transaction definition by adding some important information. Within MyARM a key performance indicator is formed from the following data:

ARM transaction definition
 is used to form the base data set for the key performance indicator. This transaction definition is defined by the application and therefore represents the applications typical character

Filter criteria
 Besides the response time each ARM transaction measurement has several attributes associated. Therefore the following attributes can be used to form a specialized KPI for the given ARM transaction definition:

 System address
  defines the host or system the transaction was executed on. To monitor the performance on a specific system this filter criteria can be used.

 URI
  within web environments each transaction measurement can be associated with the appropriate URI. Use this criteria to collect performance indicators for specific or groups of web applications.

 Users
  a transaction measurement can be executed on behalf of an user. With this criteria a specific user can be monitored.

 Context properties
  a transaction measurement can have several context information associated with. For example within a load balancer the server name can be associated with the transaction measurement and be used as a KPI. Therefore different servers can be monitored simultaneously.

Time interval
 is used to aggregate all single measurements within this interval

Response time thresholds
 are used to calculate a performance index to classify the performance of the transaction measurements. It is possible to define up to three thresholds:

 Threshold 1
  response times below this threshold are expected and judged to be good

 Threshold 2
  response times below this threshold are tolerated

Threshold 3
response times below this threshold are bad and response times above this threshold are unacceptable

Monitor
defines a condition which will trigger the execution of a notification action *(New since 4.0.x.0)*:

Response time condition
defines a threshold for the average of deviation of response times to trigger the action

Index condition
defines a percentage threshold for the response time or status index to trigger the action

Status condition
defines a count threshold for measurements with 'Failed', 'Not good', 'Abort' or 'Unknown' status to trigger the action

Note: Its only available if **runtime notification (RTN)** Add-on is licensed.

To define these KPIs MyARM provides a web application to manage such information. For details read the myarmadmin chapter.

3.3.10.2 Transaction runtime configuration

Transaction runtime configuration is used to manage and control instrumented applications remotely. Appropriate configured myarmdaemon processes are listening for new runtime configuration connections from the myarmadmin web interface to receive new runtime configurations. The myarmdaemon will propagate the runtime configurations to all instrumented applications local or remote by using the TCP datasink.

The following types of runtime configuration are supported:

Transaction management
measuring running transactions can be disabled or enabled again. This can be achieved by disabling or enabling a single transaction definition or a configured group of transactions.

Transaction monitor *(New since 4.0.x.0)*
transactions can be monitored and a notification action can be triggered according to the following conditions:

Context condition
triggers the action if the defined context property contains the specified string

Response time condition
triggers the action if the response time is greater than the specified threshold

Status condition
triggers the action if the status of the transaction equals to the specified status ('Failed', 'Not good', 'Abort' or 'Unknown')

URI condition
triggers the action if the URI contains the specified string

Note: Its only available if **runtime notification (RTN)** Add-on is licensed.

For details how to configure these concepts please read the myarmadmin chapter.

3.3.10.3 Runtime notification

Runtime notifications are new in MyARM 4.0. They are used to notify conditions of the running applications by executing a script which can either send an email/sms or react on the condition by starting a new server for a better load balancing of the application. Runtime notification is a so-called Add-on and can be used with any MyARM edition.

Currently Real time statistics and single transaction measurements can be monitored with a notification condition. For a detailed description please read the "Manage notification actions" chapter.

3.4 Handler

The following section describes all available handler. The MyARM agent implementation or the myarmdaemon are using handler to filter, monitor, manipulate or calculate derived data from measured ARM data.

3.4.1 Apache handler

The apache handler is used in conjunction with the `mod_arm4` module of the Apache HTTP Server. It manipulates the measured ARM data in such a way that virtual hosts are treated as physical hosts within MyARM. There the virtual host server name is used in MyARM to build the system address (physical host name in ARM) for each HTTP request measurement. To enable the apache handler use the following configuration entries:

`agent.handler.apache` *(New since 4.0.x.0)*
> this boolean property is used to enable (*true*) the apache handler.
>
> Default is *false*.

`agent.handler.apache.appname` *(New since 4.0.x.0)*
> specifies the ARM application name of the Apache HTTP Server as configured in the `mod_arm4` configuration file.
>
> Default is "*httpd*".

`agent.handler.apache.tranname` *(New since 4.0.x.0)*
> specifies the ARM transaction name of the HTTP request transaction of the Apache HTTP Server as configured in the `mod_arm4` configuration file.
>
> Default is "*http*".

```
# agent uses the apache handler
agent.handler.apache = true
# apache handler searches for HTTPd application name
agent.handler.apache.appname = HTTPd
# apache handler searches for HTTPRequest transaction name
agent.handler.apache.appname = HTTPRequest
```

Configuration 3.2: **Apache handler enabled**

3.4.2 Event flow auto response (efar) handler

The event flow auto response handler is used to monitor transaction measurements marked with the `ARM_FLAG_TRACE_REQUEST` flag for event flow auto response messages. If another measurement is marked as an end of flow the event flow auto response will broadcast an auto response to any listening instrumented application by sending an UDP-broadcast message to a defined port.

Any instrumented application listening for UDP messages on this port will receive such a message and will stop the root message of the event flow. With this handler the application does not need to keep track of one-way messages but the root measurement will be stopped in a synchronous manner avoiding response time calculation with possibly not synchronized clocks.

agent.transaction.eventflow.auto_response *(New since 4.0.x.0)*
> this boolean property is used to enable (*true*) the event flow auto response feature.
>
> Default is *false*.

agent.transaction.eventflow.broadcast *(New since 4.0.x.0)*
> defines the broadcast IP address and port (`ipaddr:port`) to use for sending and receiving of event flow auto response messages.
>
> Default is *127.0.0.1:5599*.

agent.transaction.eventflow.max_waiting_time *(New since 4.0.x.0)*
> specifies the maximum number of milliseconds to wait for an auto response message before stopping (or dropping) the transaction measurement waiting for an auto response message.
>
> Default is *5000*.

agent.transaction.eventflow.drop_no_response *(New since 4.0.x.0)*
> this boolean property indicates to drop (*true*) a transaction measurement if no auto response message was received. Otherwise it will be stopped with its asynchronous response time.

agent.transaction.eventflow.send_delay *(New since 4.0.x.0)*
> number of microseconds to delay sending an UDP auto response message to collect multiple auto responses within one UDP packet.
>
> Default is *100*.

```
# boolean to enable (asynchronous) event flow auto response feature
agent.transaction.eventflow.auto_response = true
# UDP broadcast address/port to send and receive event flow
# auto response messages
agent.transaction.eventflow.broadcast = "127.0.0.1:5599"
# maximum milliseconds to wait for an event flow auto
# response message
agent.transaction.eventflow.max_waiting_time = 5000
# drop measurements if no auto response was received within
# max waiting time
agent.transaction.eventflow.drop_no_response = false
# microseconds to delay auto response message to collect multiple
# auto response messages for one UDP packet
agent.transaction.eventflow.send_delay = 100
```

Configuration 3.3: **Event flow auto response feature enabled**

3.4.3 Monitor (mon) handler

The monitor handler is used to monitor recorded transaction measurements and to execute a notify script if predefined conditions occur. To enable the transaction monitoring feature use the following configuration property:

agent.transaction.monitor *(New since 4.0.x.0)*
> this boolean property is used to enable (*true*) the transaction monitor feature.
>
> Default is *false*.

For a detailed description on how to define monitor conditions for ARM transaction definitions please read the section myarmadmin transaction configuration.

3.4.4 Passivation (psv) handler

The passivation handler is used to disable (passivate) recorded transaction measurements on a per transaction definition base. To enable the transaction passivation feature use the following configuration property:

`agent.transaction.passivation` *(New since 4.0.x.0)*

> this boolean property is used to enable (*true*) the transaction passivation feature.
>
> Default is *false*.

For a detailed description on how to disable ARM transaction measurements please read the section myarmadmin transaction configuration.

3.4.5 Real time statistics (rts) handler

The rts handler is used to calculate the real time statistics (RTS) for all recorded transaction measurements. The RTS handler can be either used directly within the agent or within the `myarmdaemon`. To enable the RTS calculation feature use the following configuration properties:

`agent.rts.enable` *(New since 4.0.x.0)*

> this boolean property is used to enable (*true*) the rts handler within the ARM agent.
>
> Default is *false*.

`daemon.rts.enable` *(New since 3.0.x.0)*

> this boolean property is used to enable (*true*) the rts handler within the `myarmdaemon`.
>
> Default is *false*.

For a detailed description on how to define real time statistics please read the section myarmadmin real time statistics.

3.5   Datasinks

The following section describes all available datasinks. The MyARM agent implementation and the myarmdaemon are using datasinks to export measured ARM data to a configured datasink destination.

3.5.1   Null datasink

The null datasink in reality is a no operation datasink. Say you have setup a complete environment with ARM instrumentation enabled, but in some circumstances you want no transactions to be measured. Just configure the use of this null datasink and no data will be recorded at all:

```
# agent uses a datasink named noop
agent.sink.name = noop
# noop defines a null datasink type
noop.type = null
```

Configuration 3.4: **No operation datasink**

3.5.2   Database datasinks

The following database datasinks are supported in this edition:

3.5.2.1   SQLite datasink

The sqlite3 datasink uses the SQLite database as described in SQLite database.

3.5.2.2   MySQL datasink

The mysql datasink uses the MySQL database as described in section MySQL database.

3.5.3   Archive datasink

The archive datasink is used to store measured ARM data into a set of binary files. File names will include a timestamp based on ARM data.

In addition to the standard datasink and the basic archive properties the archive datasink uses the following properties:

<name>.file
>   set up sqlite3 archive database used within the archive datasink.
>
>   Default is *${MYARM_VARLIB_DIR}/myarm_archive.db*.

<name>.workdir
>   set up working directory for the archive datasink. This directory is used to create files and popu-late them with appropriate timed ARM data during execution. Each file is moved to the directory configured with the <name>.archivedir property if one of the following conditions are met:
>
>   1. The file size exceeds the size configured with the <name>.max_size property
>
>   2. The number of seconds since file creation exceeds the configured <name>.move_interval value

Default is *${MYARM_VARLIB_DIR}/archive/work/*.

<name>.archivedir

Base archive directory to move closed files to.

Default is *${MYARM_VARLIB_DIR}/archive/final/*.

<name>.max_size

Maximum number of bytes for an archive file. If this limit is reached the archive file is closed and a new file is created.

Default is *1MiB* (1 megabyte), minimum is *128KiB* (128 kilobyte), maximum is *2GiB* (2 gigabyte).

<name>.max_open_files

specifies the maximum number of open files. If more files are needed the least recently used file is closed.

Default is *60*, minimum is *30*, maximum is *300*.

<name>.move_interval

time interval in seconds used to move archive files from workdir to the <name>.archivedir.

Default is *1m* (1 minute), minimum is *30s* (30 seconds), maximum is *5m* (5 minutes).

<name>.zstandard

compress archive files using zstandard library. The suffix .zst will be appended to the file name.

Default is *false*.

3.5.3.1   Configuration example

```
daemon.sink.name = sink_archive
# set up sink type
sink_archive.type = "archive"
# set up sqlite3 archive database used within archive datasink
sink_archive.file = "$MYARM_VARLIB_DIR/myarm_archive.db"
# set up work directory for archive datasink
sink_archive.workdir = "$MYARM_VARLIB_DIR/archive/work/"
# base archive directory to move closed files to
sink_archive.archivedir = "$MYARM_VARLIB_DIR/archive/final/"
# maximum number of bytes for an archive file. If this limit
# is reached the archive file is closed and a new file is created
sink_archive.max_size = 1MiB
# maxmum number of open files; if more are needed the least
# recently used file is closed
sink_archive.max_open_files = 60
# time interval used to move archive files to the archivedir
sink_archive.move_interval = 1m
# compress archive files using zstandard library
sink_archive.zstandard = true
```

Configuration 3.5: **Archive sink**

3.5.4   File datasink

Using the `file` datasink all ARM data are stored in a flat file. This file can be read by the `myarmdaemon` to forward the ARM data to another destination like a TCP connection or a real database. The main purpose of this datasink in conjunction with the `myarmdaemon` is to decouple writing ARM data to the database from the instrumented application using simple file IO.

In addition to the standard datasink properties the file datasink uses the following properties:

`<name>.workfile`
>   specifies the complete file name (including directory names) for the work file. An unique ID generated by the current process will be appended to make the file unique. The ARM data is written to this file and when its closed it is moved to the configured directory.
>
>   Default is */tmp/myarmfile.data*.

`<name>.rolling.seconds`
>   specifies the number of seconds after which a new workfile will be used. The old workfile will be moved to the configured directory.
>
>   Default is *1m* (1 minute), minimum is *5s* (5 seconds) and maximum is *1h* (1 hour).

`<name>.rolling.size`
>   specifies the maximal size in bytes of the workfile. If the workfile gets bigger a new workfile is opened and the old workfile is moved to the configured directory.
>
>   Default is *128* KB. Minimum is *32* KB and the maximum is *128* MB.

`<name>.diskusage.max_used`
>   specifies the maximal size in bytes of all ARM data files in the `basic.filestorage.reader.directory` directory. If this limit is reached any new ARM data files will be dropped and an error will be reported.
>
>   Default is *100* MB. Minimum is *128* KB and no maximum limit.

`<name>.diskusage.min_free`
>   specifies the minimal free size in bytes of the file system of the `basic.filestorage.reader.directory` directory. If this limit is reached any new ARM data files will be dropped and an error will be reported.
>
>   Default is *200* MB. Minimum is *100* MB and no maximum limit.

`basic.filestorage.reader.directory`
>   specifies the directory to move closed files to. See basic file storage configuration section.

A sample file datasink configuration can look like:

```
# specify datasink name for the agent
agent.sink.name = sink_file
# datasink file type
sink_file.type = file
# set up work file for file datasink
sink_file.workfile = /opt/myarm/var/myarmfile.data
# time interval in seconds to use a new work file and move
# the old to the myarmdaemon directory.
sink_file.rolling.seconds = 1m
# number of maximal bytes for the work file. If this limit is
# reached the current work file is closed, moved to the
# myarmdaemon directory and a new work file is created.
sink_file.rolling.size = 128KB
# only use up to 100 MB of ARM data files, if this limit
# is reached new ARM data files are dropped.
sink_file.diskusage.max_used = 100MB
# at least 200 MB of disk space should be left free
sink_file.diskusage.min_free = 200MB
```

Configuration 3.6: **File sink**

3.5.5   TCP datasink

The `tcp` datasink is used in conjunction with the myarmdaemon tool. It connects to the myarmdaemon program and sends all ARM data to the myarmdaemon process through a TCP socket connection. Note that the `tcp` datasink retries to connect to the myarmdaemon periodically as long as there is ARM data available to send to the myarmdaemon. If there is no ARM data available to send for at least `<name>.connection.idle` time, the connection is closed.

In addition to the standard datasink properties, the `tcp` datasink uses the following properties:

`<name>.host`
> name of the host to connect to the myarmdaemon process.
>
> Default is *localhost*.

`<name>.port`
> port number on which the myarmdaemon process is listening and accepts connections.
>
> Default is *5557*.

`<name>.connection.idle`
> specified as an interval (with a suffix `m` for minutes, `s` for seconds or `h` for hours) to wait before closing the connection to the myarmdaemon if no ARM data needs to be sent.
>
> Default is *5m* (5 minutes), minimum is *30s* (30 seconds) and the maximum of *1h* (1 hour) for the connection idle time.

`<name>.connection.keepalive`
> specified as an interval (with a suffix `m` for minutes or `s` for seconds) to send a keep alive message to the myarmdaemon if there is no ARM data to be sent.
>
> Default is *1m* (1 minute), minimum is *30s* (30 seconds) and the maximum is 1/2 of the value for the `<name>.connection.idle` `<name>.connection.idle` property.

`<name>.connection.reconnect`
> specified as an interval (with a suffix m for minutes or s for seconds) to wait before MyARM retries to connect to the myarmdaemon process again.
>
> Default is *30s* (30 seconds), the minimum is *10s* (10 seconds) and the maximum is *5m* (5 minutes).

`<name>.readwrite.timeout`
> timeout value (with a suffix ms for milliseconds or s for seconds) to wait for reads or writes of ARM data through the connection.
>
> Default is *500ms* (500 milliseconds). Minimum is *250ms* (250 milliseconds) and the maximum is *2s* (2 seconds).

`<name>.reply.timeout`
> timeout value (with a suffix ms for milliseconds or s for seconds) to wait for a reply from the myarmdaemon.
>
> Default is *2s* (2 seconds), minimum is *500ms* (500 milliseconds) and the maximum is *4s* (4 seconds).

3.5.5.1 Configuration example

```
# agent uses a datasink named sink_tcp
agent.sink.name = sink_tcp
# set up sink type
sink_tcp.type = tcp
# set up host myarmdaemon is running on
sink_tcp.host  = localhost
# set up port myarmdaemon is listening for incoming connections
sink_tcp.port = 5557
# read/write time out interval
sink_tcp.readwrite.timeout = 1s
# the interval for reconnecting to the daemon in seconds
sink_tcp.connection.reconnect = 1m
# send a keep alive message after 1 minute if no data
sink_tcp.connection.keepalive = 1m
# close the connection to myarmdaemon after 5 minutes if no data
sink_tcp.connection.idle = 5m
```

Configuration 3.7: **TCP sink**

3.6   myarmdaemon – ARM data collection daemon

The agent daemon is running in the background and supports the MyARM agent in delivering the measured ARM data to its final database or archive. It allows the central administration component of MyARM to configure the running MyARM system using the so-called runtime configuration. Also it can be controlled by the `myarmdaemoncmd` command to stop it, query some information.

3.6.1   Command line options

Usage:

```
myarmdaemon [options]
```

The `myarmdaemon` command supports standard options described in the appendix "Standard options".

-tp *port*, **--tcp-port** *port* specifies the port number to wait for incoming TCP data connections if tcp mode is enabled. This overwrites the configuration property.

3.6.2   Collection mode

The `myarmdaemon` can be executed in different data collection modes which can be configured using the `daemon.collection.mode` configuration property:

none
> This defines that the `myarmdaemon` should not collect any ARM data. This is usefull if the `myarmdaemon` is configured to provide only management interfaces (e.g. runtime configuration, access to archive files, Apache Spark support, etc).

tcp
> The `myarmdaemon` opens a TCP socket to listen for incoming ARM data connections from TCP datasinks.

file
> The `myarmdaemon` scans the `basic.filestorage.reader.directory` for new ARM data files written by file datasinks.

tcp,file
> The `myarmdaemon` opens a TCP socket to listen for incoming ARM data connections from TCP datasinks and scans the `basic.filestorage.reader.directory` for new ARM data files written by file datasinks.

tcp->file
> The `myarmdaemon` opens a TCP socket to listen for incoming ARM data connections from TCP datasinks and writes all received ARM data to the `basic.filestorage.reader.directory`. Thus any ARM data received through a TCP socket is directly written to hard disk and later read in and written to its destination datasink as defined by the `daemon.sink.name`.

3.6.3   Main configuration

daemon.sink.name
> specifies the datasink the received ARM data is written to if `daemon.collection.mode` configuration property is not set to `none` (see appendix "Configuring datasink component" for more details).

daemon.group
> defines the group under which the myarmdaemon should be executed. If the myarmdaemon is started as super-user (root) it can change its group identity to the configured group name.
>
> For example a *myarm* group.

daemon.user
> defines the user under which the myarmdaemon should be executed. If the myarmdaemon is started as super-user (root) it can change its user identity to the user name.
>
> For example a *myarm* user.

daemon.log.*
> defines the logging message options and destination for the myarmdaemon program (See appendix "Configuring Log facility" for more details).

daemon.resource.watchdog.*
> defines the resource watchdog logging for the myarmdaemon program (See appendix "Configuring resource watchdog" for more details).

daemon.pidfile
> defines the file where the myarmdaemon writes its process id to.
>
> Default is *${MYARM_VARRUN_DIR}/daemon.pid*.

3.6.4   TCP server component

If the daemon.collection.mode configuration property was set to use the TCP server within the myarmdaemon, a TCP server listening socket will be created for listening for incoming connection requests from ARM instrumented applications using the tcp datasink. The following configuration properties can be used to control the TCP server component.

3.6.4.1   Configuration

daemon.tcp.host
> defines the host/interface where the daemon listens for incoming data connections if the TCP part is enabled by the daemon.collection.mode.
>
> Default host is *localhost*.

daemon.tcp.port
> defines the port number where incoming connections are accepted if the TCP part is enabled by the daemon.collection.mode.
>
> Default port is *5557*.

daemon.tcp.max_clients
> defines the maximum number of TCP clients if the TCP part is enabled by the daemon.collection.mode.
>
> **Note:** This number should always be less than the number of allowed open files of the process. If not, the myarmdaemon tries to increase the number of open files of the process and if this fails, it terminates directly!
>
> Default is *100*, minimum is *1*, maximum is *32768*.

daemon.tcp.flowcontrol.threshold
> if more than this percentage of ARM data buffers are in use, flow control will be enabled with the tcp datasink. If less than daemoon.tcp.flowcontrol.threshold ARM data buffers are currently used a new incoming ARM data buffer is acknowledged directly after reception. If more

than `daemon.tcp.flowcontrol.threshold` ARM data buffers are in use, the acknowledgement of the reception is postponed until the ARM data buffer was processed and written to the `myarmdaemon` configured datasink.

Default is *10%*, minimum is *5%*, maximum is *50%*.

3.6.5 Apache Spark component

The Apache Spark component is used in conjunction with the `archive` datasink. When the `myarmdaemon` is configured to store ARM data in the archive, it is possible to export ARM transaction measurement data to an Apache Spark instance using a TCP socket. The Apache Spark TCP client connects to the `myarmdaemon` Apache Spark TCP server and requests data in CSV format for a specified time interval.

3.6.5.1 Configuration

`daemon.spark.enable` *(New since 4.1.x.0)*

    boolean which indicates if the daemon accepts Apache Spark client connections.

    Default is *false*.

`daemon.spark.host` *(New since 4.1.x.0)*

    defines the host/interface where the daemon listens for incoming Apache Spark connections.

    Default host is *localhost*.

`daemon.spark.port` *(New since 4.1.x.0)*

    specifies the Apache Spark TCP channel port to listen for Apache Spark connections.

    Default is *5587*.

`daemon.spark.client.host` *(New since 4.1.x.0)*

    specifies the hostname or IP address from the Apache Spark client. If not empty, only connections from this host/interface are accepted

    Default is *localhost*.

`daemon.spark.csvformat` *(New since 4.1.x.0)*

    specifies the format string to format the CSV line for each transaction measurement sent to Apache Spark. See formatting a transaction.

    Default is *%h,%n,%S,%x,%E\n*.

`daemon.spark.archive.movefiles` *(New since 4.1.x.0)*

    specifies that a file should be moved from the source directory to the destination directory after reading in a MyARM data file.

    Default is *true*.

`daemon.spark.archive.sourcedir` *(New since 4.1.x.0)*

    specifies the source directory to read MyARM data files from.

    Default is *${MYARM_VARLIB_DIR}/archive/current/*.

`daemon.spark.archive.destinationdir` *(New since 4.1.x.0)*

    specifies the destination directory to move files to when read in.

    Default is *${MYARM_VARLIB_DIR}/archive/final/*.

3.6.6   Archive reader component

The archive reader component provides a TCP socket interface for accessing files stored in the Archive. It is used by the `myarmbrowser` user interface and the `myarmarchive` command line tool to retrieve transaction data from the archive.

3.6.6.1   Configuration

`daemon.armdata.enable` *(New since 4.1.x.0)*
> boolean which indicates if the daemon accepts archive reader client connections (true) or not (false).
>
> Default is *false*.

`daemon.armdata.host` *(New since 4.1.x.0)*
> defines the host/interface where the daemon listens for incoming archive reader connections.
>
> Default host is *localhost*.

`daemon.armdata.port` *(New since 4.1.x.0)*
> specifies the archive reader TCP channel port to listen for new connections.
>
> Default is *5597*.

`daemon.armdata.format` *(New since 4.1.x.0)*
> specifies the format string to format the output for each transaction measurement returned by the archive reader. See formatting a transaction.
>
> Default is *%n\t%r\t%s\t%q %p %A %U %c\n*.

3.6.7   Threshold reader component

The threshold reader component periodically checks the configured threshold directory for new threshold files, reads these files and stores the contents into the configured database.

3.6.7.1   Configuration

`daemon.threshold.reader.enable`
> Boolean which indicates if the `myarmdaemon` should monitor the archive threshold directory for new threshold files (true) and to process such new files.
>
> Default is *true*.
>
> The `myarmdaemon` reads threshold files, stores contained data into the configured database and deletes the files afterwards. See `basic.archive.database.url` and `basic.archive.database.threshold.name` configuration properties for details.

`daemon.threshold.reader.directory`
> Specifies the directory which should by periodically checked for new threshold files.
>
> Default is *${MYARM_VARLIB_DIR}/archive/final/thresholds/*.

`daemon.threshold.reader.transaction.count.max`
> Specifies the maximum count per transaction definition of imported transactions within the defined transaction count interval (see next item). If zero is specified the limiting is disabled and all threshold transactions are imported into the database.
>
> Default is *0* (zero), minimum is *0* (zero), maximum is *100000* (one hundred thousand).

`daemon.threshold.reader.transaction.count.interval`
> Specifies the interval to be used for the limit to import threshold transaction into the threshold database.

> Default is *1h* (1 hour), minimum is *1m* (1 minute), maximum is *24h* (1 day).

`daemon.threshold.reader.interval`
> Specifies the interval to check for new threshold files within the archive.

> Default is *1m* (1 minute), minimum is *1m* (1 minute), maximum is *1h* (1 hour).

`daemon.threshold.reader.cleanup.interval`
> Specifies the interval to periodically clean up old threshold data within the threshold database.

> Default is *1m* (1 minute), minimum is *1m* (1 minute), maximum is *1h* (1 hour).

> Old threshold data is the data which is older than current time minus the period specified in the `basic.archive.cleanup.keep_data` configuration property.

3.6.8   Command component

The `myarmdaemon` can be controlled using a TCP command channel (if enabled). This allows an administrator to adopt the behaviour of the `myarmdaemon` at runtime without stopping and restarting the `myarmdaemon`.

3.6.8.1   Commands

The following commands are currently supported:

`help`
> queries all available commands for this daemon.

`info`
> queries general version, build information and enabled features about the running daemon.

`stop`
> signals the daemon to terminate.

`status`
> queries current status information from the `myarmdaemon` process. It supports the following options:

> **clients**
>> shows detailed information about TCP client threads.

> **speed**
>> shows incoming data speed (bytes per second).

> **all**
>> shows all available information.

> **long**
>> shows more detailed information.

> **noheader**
>> avoid printing a header. This is useful if `myarmdaemoncmd` is used to execute the status command in a loop.

3.6.8.2 Configuration

`daemon.cmd.enable`
> enables or disables the command TCP channel for the `myarmdaemon`.
>
> Default is *true*.

`daemon.cmd.host`
> defines the host/interface where the daemon listens for incoming command connections.
>
> Default host is *localhost*.

`daemon.cmd.port`
> specifies the command TCP channel port to use.

3.6.9 Runtime configuration component

The runtime configuration component is used to accept new runtime configurations from the central `myarmadmin` web application using a TCP channel.

3.6.9.1 Configuration

`daemon.runtime.config.enable`
> boolean which indicates if the daemon accepts new runtime configurations (true) from the runtime administration web frontend or not (false).
>
> Default is *false*.

`daemon.runtime.config.localhost` *(New since 4.0.x.0)*
> boolean which indicates if the daemon should listen on localhost (true) for accepting runtime configuration connections or on the default interface of the host (false).
>
> Default is *true*.

`daemon.rts.enable`
> boolean which indicates if the daemon should calculate real time statistics (RTS) (true) or not (false). Note this only works if runtime configuration is enabled and appropriate RTS definitions are available.
>
> Default is *false*.

3.6.10 Web application component

Starting with version 4.1 of MyARM the `myarmdaemon` can start the standalone web applications provided by MyARM. This can be enabled by specifying the interface/host and port on which the HTTP server of the web application should listen for incoming HTTP connections.

`daemon.web.admin.http`
> defines the interface/host and port (delimited by a colon) on which the `myarmadmin` should listen for incoming HTTP connections. If an empty string is specified this feature will be disabled. For example `localhost:8081` is commonly used for local HTTP server instances.
>
> Default is " " (empty string).

`daemon.web.browser.http`
> defines the interface/host and port (delimited by a colon) on which the `myarmbrowser` should listen for incoming HTTP connections. If an empty string is specified this feature will be disabled. For example `localhost:8080` is commonly used for local HTTP server instances.
>
> Default is " " (empty string).

daemon.web.rtsbrowser.http

defines the interface/host and port (delimited by a colon) on which the `myarmrtsbrowser` should listen for incoming HTTP connections. If an empty string is specified this feature will be disabled. For example `localhost:8082` is commonly used for local HTTP server instances.

Default is "" (empty string).

daemon.web.rtsmonitor.http

defines the interface/host and port (delimited by a colon) on which the `myarmrtsmonitor` should listen for incoming HTTP connections. If an empty string is specified this feature will be disabled. For example `localhost:8083` is commonly used for local HTTP server instances.

Default is "" (empty string).

3.6.11   Configuration example

```
# myarmdaemon processing mode: receive data via TCP and
# store received data into a MySQL/MariaDB database
daemon.collection.mode = tcp
# set up host/interface and port where myarmdaemon is listening
# for incoming data connections, limit to 100 concurrent clients
daemon.tcp.host = localhost
daemon.tcp.port = 5557
daemon.tcp.max_clients = 100
# myarmdaemon TCP flow control threshold
daemon.tcp.flowcontrol.threshold = 10
# use MySQL datasink database
daemon.sink.name = db_mysql
# myarmdaemon process user/group name and pid file
daemon.user = myarm
daemon.group = myarm
daemon.pidfile = /opt/myarm/var/daemon.pid
# set up host/port where myarmdaemon is listening for command
# connections
daemon.cmd.host = localhost
daemon.cmd.port = 5554
```

Configuration 3.8: **myarmdaemon**

3.6.12   See Also

myarmoptions, myarmconfig, myarmdaemoncmd

3.7   Typical deployment

This sections outlines some typical deployment environments for an ARM instrumented distributed system. Here we assume that the distributed system consists out of

- 2 client hosts running a client application

- 1 server host with a web and an application server

- 2 database hosts each running a database server

The clients requests services from the application server through the web server. The application server will process the requests and will use the two database servers according to the type of request. Each box in the following diagram represents a host with the host name in parenthesis within our local domain "`local`":
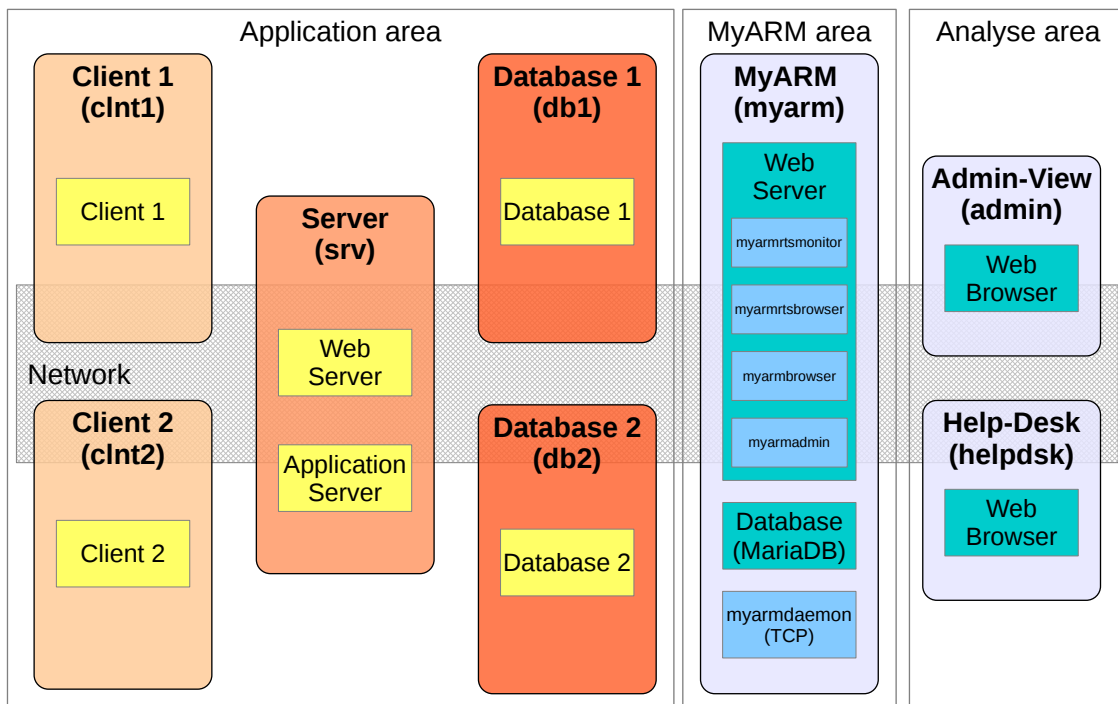
Figure 3.6: Deployment overview

Within such an environment we recommend to setup a separate MyARM server which hosts the following applications:

- a MySQL (or MariaDB) database for storing the ARM measurements. Note: Within the Community Edition just use the SQLite database.

- MyARM running a `myarmdaemon` process with an enabled TCP server to receive measured ARM data through an TCP socket.

- a web server with FastCGI (MyARM Web Edition only) support to deploy the MyARM web applications (e.g. `myarmbrowser`, `myarmrtsbrowser` , `myarmrtsmonitor` , `myarmadmin`).

Administration or the help desk can analyse performance trends or a specific performance problem using the MyARM web front ends.

3.7.1   Using TCP datasink

Once MyARM is installed on each host the configuration needs to be adopted to work properly. The following diagram gives an overview of our deployment example with black arrows representing writing data flow and gray arrows representing a reading data flow:
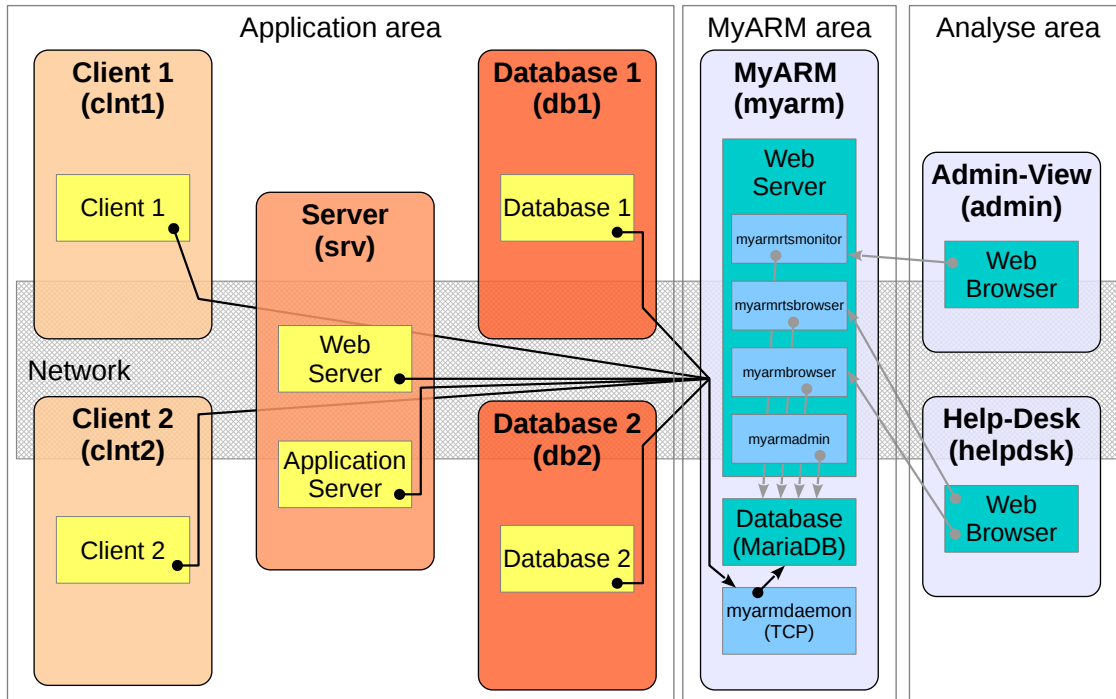
Figure 3.7: Deployment with TCP configuration

As shown in the diagram we can divide the diagram into three different areas:

1. The "**Application area**" which needs the MyARM configuration how to write the measured ARM data into the destination database.

2. The "**MyARM area**" where the `myarmdaemon` with the TCP server, the MySQL SQL database and the web server for providing the MyARM web front ends are running. Note: `myarmadmin` is used to administrate the MyARM infrastructure and is not taken into account here but shown for the sake of completeness.

3. The "**Analyse area**" where some web browsers are used to analyse the measured performance and response time data.

We start to configure the MyARM installation within the "**MyARM area**":

- To configure MyARM properly template configuration files exists in the directory `${MYARM_ROOT}/conf/templates`. Since we want to use the `myarmdaemon` with a TCP server and a MySQL database we use the "`tcp_mysql.conf`" by creating a symbolic link (or by copying):

```
ln -s ${MYARM_ROOT}/conf/templates/tcp_mysql.conf \
      ${MYARM_ROOT}/conf/myarm.conf
```

To adopt this configuration template a configuration file named "`user.conf`" is used within the configuration `${MYARM_ROOT}/conf` directory. In this file we will enter our changes which will override the MyARM default configuration properties.

- The `myarmdaemon` writes any received data into the "`MySQL`" database. Therefore we need to setup the database user "`myarm`" and password. The default configuration is to use "`localhost`" for connecting to the database thus that's all:

```
1: db_mysql.user = myarm
2: db_mysql.password = "dbpasswd"
```

Configuration 3.9: **Daemon database configuration**

- On the "`myarm.local`" host the `myarmdaemon` is running a TCP server thus we need to configure the TCP part first. We define the host and port where the daemon is listening for new data connections (line one and two) and finally we limit the number of concurrent client connections to ten (line three):

```
1: daemon.tcp.host = myarm.local
2: daemon.tcp.port = 5557
3: daemon.tcp.max_clients = 10
```

Configuration 3.10: **Daemon TCP configuration**

- For configuring the MyARM web front ends two different options exist. First configuring a standalone HTTP server and FastCGI integration option to embed the MyARM web front ends into an existing web server infrastructure. Please read the web deployment section for details.

So finally our "`user.conf`" for the host in the "**MyARM area**" looks as follows:

```
daemon.tcp.host = myarm.local
daemon.tcp.port = 5557
daemon.tcp.max_clients = 10
db_mysql.user = myarm
db_mysql.password = "dbpasswd"
```

Configuration 3.11: **myarm.local daemon user.conf configuration**

Now we need to configure the MyARM installations within the "**Application area**":

- To configure MyARM properly we use another template configuration file named "`tcp.conf`" by creating a symbolic link:

```
ln -s ${MYARM_ROOT}/conf/templates/tcp.conf \
      ${MYARM_ROOT}/conf/myarm.conf
```

- Next we need to tell MyARM where the `myarmdaemon` with the TCP server is listening for new connections by creating the new file "`user.conf`" within the configuration `${MYARM_ROOT}/conf` directory:

```
1: sink_tcp.host = myarm.local
2: sink_tcp.port = 5557
```

Configuration 3.12: **Client TCP configuration**

For further configuration properties read the TCP datasink section.

```
sink_tcp.host = myarm.local
sink_tcp.port = 5557
```

Configuration 3.13: **Application hosts user.conf configuration**

For the "**Analyse area**" no additional configuration is needed. Depending on the web deployment you used the different MyARM web application can be reached with the following URLs:

- Standalone HTTP server

    - `myarmbrowser`: http://myarm.local:8080

    - `myarmadmin`: http://myarm.local:8081

    - `myarmrtsbrowser`: http://myarm.local:8082

    - `myarmrtsmonitor`: http://myarm.local:8083

- FastCGI integration

    - `myarmbrowser`: http://myarm.local/fcgi-bin/myarmbrowser.fcgi

    - `myarmadmin`: http://myarm.local/fcgi-bin/myarmadmin.fcgi

    - `myarmrtsbrowser`: http://myarm.local/fcgi-bin/myarmrtsbrowser.fcgi

    - `myarmrtsmonitor`: http://myarm.local/fcgi-bin/myarmrtsmonitor.fcgi

3.7.2   Using file and TCP datasink

In the previous section we configured the MyARM agent to use directly a TCP connection to the `myarmdaemon` running on the "`myarm.local`". In some circumstances it is not wanted that each ARM instrumented application connects to a TCP server. For example if different command line tools are instrumented and are executed within a batch job its not a good idea to connect to the TCP server each time a batch command has some ARM measurements.

In such environments MyARM provides the `file` datasink to write ARM measurements to local hard disk instead of sending the data through a TCP connection. The `myarmdaemon` is used to collect these files on the "`localhost`" and sends the data to the `myarmdaemon` running on "`myarm.local`" using only one TCP connection:
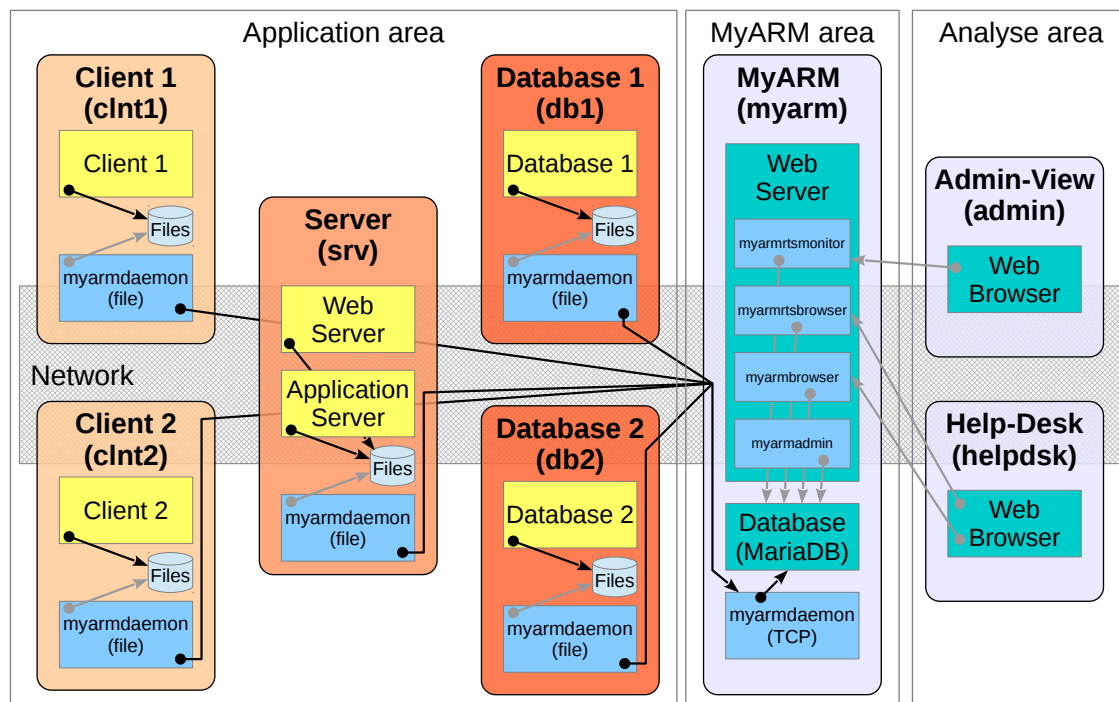
Figure 3.8: Deployment with file and TCP configuration

- To configure MyARM in such a way we use the "`file_tcp.conf`" by creating a symbolic link:

```
ln -s ${MYARM_ROOT}/conf/templates/file_tcp.conf \
    ${MYARM_ROOT}/conf/myarm.conf
```

- Now you have to make sure that the `myarmdaemon` is running on each application host otherwise the data is only stored onto local hard drive but not forwarded to the `myarmdaemon` running on "`myarm.local`".

  Under Windows you can install the `myarmdaemon` as a Windows service and under Linux the `myarmdaemon.sh` script can be used in the appropriate init systems.

  Since we already configured the TCP part we can just use the "`user.conf`" from the previous section.

- The `file datasink` has several configuration options to ensure that the local hard drive will have enough space left and to fine tune the forwarding of measured data.

Chapter 4

Databases

MyARM supports several databases for storage and retrival of ARM data. This section describes the basic setup and configuration for each database.

4.1 SQLite database

SQLite is a simple and easy to use database. It provides a complete SQL database by using just a single data file. The file used for storing all ARM data can be configured using MyARM configuration properties as described below.

`<name>.type`

database type which has to be `sqlite3`.

`<name>.file`

specifies the path and name for SQLite database file.

Default is *${MYARM_VARLIB_DIR}/myarm_sqlite.db*

```
# agent uses a datasink named db_sqlite
agent.sink.name   = db_sqlite
# analysis tools uses also the sqlite3 database
tools.source.name = db_sqlite

# define the type of the database
db_sqlite.type  = sqlite3
# define the SQLite database file
db_sqlite.file  = /opt/myarm/var/sqlite.db
```

Configuration 4.1: **SQLite database**

4.2 MySQL database

For production use a MySQL database can be used to store all ARM data. The database has to be setup correctly. To setup correct databases and tables within MySQL, use the `myarminitdb`.

Note that the configured MySQL database user needs the appropriate privileges to create the MyARM databases. Please use the following SQL statement to grant the privileges:

```
someuser@localhost: mysql -u root
GRANT ALL PRIVILEGES ON *.* TO 'someuser'@'localhost';
```

The MySQL database configuration is split into six database properties defining specific databases for definition data, application instance data, transaction instance and derived runtime data.

These database configuration properties should be used in production environments with a high number of transaction measurements. Therefore the tentative static data like ARM definitions can be separated from the highly dynamic data like transaction measurements, context and metrics values.

For a detailed description of MySQL insert performance please read the appendix Using MySQL.

By default the MyARM MySQL configuration uses the *myarm* user with no password. It is recommended that you create such a MySQL user and change the password according to your security standards. All changes should be written into the `user.conf` file, which will override any standard configuration property.

To configure the MySQL database within MyARM the following properties are used:

`<name>.user`
> database user name.
>
> Default is an empty string (*""*).

`<name>.password`
> password for the database user.
>
> Default is an empty string (*""*).

`<name>.host`
> host where the database is running.
>
> Default is *localhost*.

`<name>.port`
> port number on which the MySQL server listens for incoming connections.
>
> Default is *3306*.

`<name>.connections`
> number of concurrent connections to the MySQL database server.
>
> Default and minimum is *1*, maximum is *16*.
>
> Note: The configuration is only used if **Multiple database connections (MDB)** Add-on is licensed.

`<name>.reconnect`
> defines an interval value in seconds to wait before MyARM retries to connect to the MySQL database process again.
>
> Default is *30s* (30 seconds), minimum is *10s* (10 seconds), maximum is *5m* (5 minutes).

`<name>.database.config`
> Database for the storage of MyARM web 2.0 configuration data.
>
> Default is *MyARM_Config*.

`<name>.database.definition`
> Database for the storage of ARM definition data such as application and transaction definitions which are also called meta data.
>
> Default is *MyARM_Def*.

`<name>.database.application`
> Database for the storage of ARM application instance data such as application instances and application context values.
>
> Default is *MyARM_App*.

`<name>.database.transaction`
>   Database for the storage of ARM transaction instance data such as transaction measurements, transaction properties and metrics as well as transaction correlation information.
>
>   Default is *MyARM_Tran*.
>
>   The MySQL datasink supports on-the-fly generation of transaction databases according to the transaction timestamp. For this purpose the transaction database name can contain '`%`' patterns which are replaced by appropriate values from the transaction to store. The following patterns are supported:

`'%d'` – is replaced by the two digit day of the month ('`DD`') the transaction started.

`'%m'` – is replaced by the two digit month of the year ('`MM`') the transaction started.

`'%y'` – is replaced by the two digit year ('`YY`') the transaction started.

`'%Y'` – is replaced by the four digit year ('`YYYY`') the transaction started.

```
# define the database name where to store MyARM web config data
db_mysql.database.config = MyARM_Date_Config
# define the database name where to store the definition data
db_mysql.database.definition = MyARM_Date_Def
# define the database name where to store the application data
db_mysql.database.application = MyARM_Date_App
# define the database name where to store the transaction data
db_mysql.database.transaction = MyARM_Date_%Y%m%d_Tran
```

Configuration 4.2: **MySQL transaction database pattern**

`<name>.database.rts`
>   Database for the storage of real time statistics data.
>
>   Default is *MyARM_RTS*.

`<name>.database.rtevent` *(New since 4.0.x.0)*
>   Database for the storage of runtime events used to trigger notification actions.
>
>   Default is *MyARM_RTEvent*.

`<name>.engine`
>   MySQL storage engine to use for the database.
>
>   Default is *INNODB*.

The following example configures a MySQL datasink where the database runs on host *dbhost* and the database name is *MyARM*. The MySQL user is set to the user *myarm*:

```
# agent uses a datasink named db_mysql
agent.sink.name   = db_mysql
# define the type of the datasink
db_mysql.type     = mysql
# define the host where the MySQL database runs
db_mysql.host     = dbhost
# number of concurrent connections to MySQL database
db_mysql.connections = 4
# the reconnect interval for reconnecting to the database in seconds
db_mysql.reconnect = 30s
# define the user to log into the database
db_mysql.user     = myarm
# define the database name where to store the MyARM web config data
db_mysql.database.definition = MyARM_Config
# define the database name where to store the ARM definition data
db_mysql.database.definition = MyARM_Def
# define the database name where to store the ARM application data
db_mysql.database.application = MyARM_App
# define the database name where to store the ARM transaction data
db_mysql.database.transaction = MyARM_Tran
# define the database name where to store the RTS data
db_mysql.database.rts = MyARM_RTS
# define the database name where to store the runtime events
db_mysql.database.rtevent = MyARM_RTEvent
# set the default storage engine for creating tables
db_mysql.engine   = INNODB
```

Configuration 4.3: **MySQL sink**

MyARM also provides its own compiled mysql client library. If you have problems connecting to the MySQL database it is possible that your MySQL installation uses connecting socket different from the library provided (by the MyARM distribution). By default, MySQL searches for a connection socket named /tmp/mysql.sock in Unix environments. If your server uses a different path please setup the MYSQL_UNIX_PORT variable correctly.

4.3   MariaDB database

Since MariaDB is a binary drop in replacement of the same MySQL version, we support MariaDB 5.5 to be used instead of a MySQL database. Just use the appropriate MySQL configuration files and properties as described above.

Chapter 5

Command line tools

This section describes all command line tools provided by MyARM. Here is a brief overview of all command line tools.

myarminfo
> displays current settings and configuration values.

myarminitdb
> initializes the configured database.

myarmdefinition
> displays registered ARM definitions.

myarmquery
> displays application and transaction instances.

myarmcorrelate
> creates transaction correlation information for parent/child transaction analysis.

myarmdelete
> deletes application and/or transaction data.

myarmarchive
> queries transaction data from a MyARM archive managed by a `myarmdaemon`.

myarmstat
> computes statistical metrics of transactions.

myarmchain
> computes statistical metrics of transactions chains.

myarmexport
> exports ARM data to a database or a XML file.

myarmimport
> imports ARM data from a database or XML file.

myarmlog
> displays log messages stored in the database.

myarmmodify
> modifies data sets within the database.

myarmdaemoncmd
> sends commands to MyARM `myarmdaemon` processes.

5.1 myarminfo – displays information about MyARM

The `myarminfo` program displays current settings of the MyARM installation. By default it prints any used environment variable with its contents and shows the available and licensed components such as datasinks, language bindings and supported databases. If the following options are used, different aspects of the MyARM installation are shown.

5.1.1 Command line options

Usage:

```
myarminfo [options]
```

-c, --config
  shows the current configuration properties.

-d, --database
  displays an overview of the contents of the currently configured database.

-n, --db-names
  displays database names within the SQL database (only supported by MySQL or MariaDB).

-pw *password*, **--password** *password*
  generates a password hash of the passed password used by the myarmbrowser to enable password login for a pre-defined user.

-p *property*, **--property** *property*
  tries to read the given property path and prints its configured value.

5.1.2 Example

```
ruppert@myarm /opt/myarm $ myarminfo
MyARM Enterprise Edition Version 4.0.5644.0 for Linux.
Copyright 2005-2015 MyARM GmbH
Web:   http://myarm.com/
eMail: info@myarm.com


Environment variables:
MYARM_ROOT=/opt/myarm
MYARM_CONFIG_URL=file:////opt/myarm/conf/sqlite3.conf
MYARM_VARRUN_DIR=/opt/myarm/var
MYARM_VARLOG_DIR=/opt/myarm/var
MYARM_VARLIB_DIR=/opt/myarm/var
MYARM_LICENSE_KEY=82689db90de80bdcbd56f41eeec9ddca35b486d1


Used configuration file is : /opt/myarm/conf/sqlite3.conf


License expiry date: unlimited
Licensed datasinks: tcp, mysql, xml, sqlite3, file, myarm
Licensed datasources: mysql, xml, sqlite3, myarm
Licensed language bindings: ARM 4.0 C, ARM 4.0 Java, ARM 4.0 C#, ARM 2.0 C
Licensed language frameworks: ARM 4.0 C++, ARM 4.0 C++ QArm4
Licensed analysis tools: Command line tools, Manager (GUI), Browser (WUI)
```

Example output 5.1 : **myarminfo output which shows the overall configuration of MyARM**

5.1.3   Database contents overview

```
ruppert@myarm /opt/myarm $ myarminfo -d
Overview of SQLite database:
Configuration:
--------------
File        : /opt/myarm/var/myarm_sqlite.db

Contents:
---------
Applications: Definitions 3 (Properties 0), Instances 16 (Properties 0)
Transactions: Definitions 6 (Properties 20), Instances 65 (Properties 27)
Metrics     : Definitions 0, Instances 0
Systems     : 1
Users       : 0
Logs        : 9
```

Example output 5.2 : **myarminfo-db output which shows the database contents**

5.1.4   See Also

myarmoptions, myarmconfig

5.2   myarminitdb – initializes the configured database

The `myarminitdb` program is used to initialize the currently configured database. In particular it creates the database tables needed by MyARM.

5.2.1   Command line options

Usage:

```
myarminitdb [options]
```

The `myarminitdb` command supports the standard options described in appendix "Standard options" and the following:

-c, --create
>   creates the configured database and all needed tables.

-d, --drop
>   deletes the configured database and all its tables.

--cleanup
>   drops and again creates the configured database and all needed tables.

-e *name***, --engine** *name*
>   selects the storage engine to use for the created database. Currently, this is only supported for the MySQL database.

-q, --quiet
>   suppresses any information output.

5.2.2   Example

Invoking the `myarminitdb` command without an option it prints the current database layout version:

```
ruppert@myarm /opt/myarm $ myarminitdb
myarminitdb: MyARM database tables version=1 found and is up to date
```

Example output 5.3 : **myarminitdb without an option it shows the current database layout version**

5.2.3   See Also

myarmoptions, myarmconfig

5.3   myarmdefinition – displays registered ARM definitions

The `myarmdefinition` command line tool is used to view and list registered ARM definitions.

5.3.1   Command line options

Usage:

```
myarmdefinition [options] [name1] [name2]
```

name1*, *name2
>   specifies the name of the definition to process. This can be either an application and/or transaction name, a system address, an user name or a metric which depends on the entity selection option as described below.
>
>   For details about specifying application and/or transaction names read the section "Application and transaction names".

The `myarmdefinition` command supports standard options described in appendix "Standard options".

5.3.1.1   Entity selection options

With one of the following entity selection options, a definition entity is selected to be printed. If a name is omitted, all definitions of the appropriate selection are printed out. Otherwise, only the definitions with the name(s) provided are printed out. If a name could not be found, an exit code greater than zero is returned.

-a, --apps
>   selects application definition output.

-t, --trans
>   selects transaction definition output. For transaction definitions the optional names have to be specified in the form `<appName>:<tranName>`. If you want to select a specific transaction name for all application definitions just provide an empty application name `:<tranName>`.

-m, --metrics
>   selects metric definition output.

-u, --users
>   selects users output.

-s, --systems
>   selects system address output.

5.3.1.2   Other options

--all
>   if specified, print all data of the definition. Otherwise only the name of the definition is printed.

-r, --recursive
>   if specified, also recursively print out any associated ARM definition. For example if application definitions are selected, print out any transactions and metrics associated with the application definition.

-i, --ids
>   if specified also print out any associated ARM ID.

-b64, --base64
>   prints ARM IDs in base64 encoding instead of hex encoding.

5.3.2   Example

```
ruppert@myarm:~$  myarmdefinition -t --all httpd
  1. Tran: httpd:HTTP
     Context property names:
        1. HostInfo
        2. RemoteAddress
        3. RemoteUser
        4. Scheme
        5. QueryString
        6. Protocol
  2. Tran: httpd:HTTP
     Context property names:
        1. ServerName
        2. ServerPort
        3. RemoteAddress
        4. RemoteUser
        5. Scheme
        6. QueryString
        7. Protocol
```

Example output 5.4 : **myarmdefinition showing all definitions for the application httpd**

5.3.3   See Also

myarmoptions, myarminfo, myarmquery, myarmdelete, myarmconfig

5.4   myarmquery – displays application and transaction instances

This section describes the `myarmquery` command line tool to submit queries to a MyARM datasource. This tool allows queries to any MyARM datasource (database), selecting instances ofapplications and transactions measured by the MyARM agent. It offers various selection criteria such as application and/or transaction names, start and stop time of an instance and some other attributes of these instances. It is possible to limit the query to a maximum number of instances. The following example shows the output of 10 HTTP transaction measurements starting at index 120.

```
$ myarmquery -ri 120 -rm 10 httpd
121. HTTP  40.828 ms GOOD    01.10.12 09:37:20.036 myarm.info /query
122. HTTP  43.812 ms GOOD    01.10.12 09:37:21.465 myarm.info /query
123. HTTP  41.042 ms GOOD    01.10.12 09:37:22.881 myarm.info /ticket/10
124. HTTP  94.771 ms GOOD    01.10.12 09:37:24.292 myarm.info /wiki/WikiRes...
125. HTTP  49.893 ms FAILED  01.10.12 09:37:25.900 myarm.info /wiki/about
126. HTTP 761.434 ms GOOD    01.10.12 09:55:50.362 myarm.info /wiki/TracQuery
127. HTTP 120.569 ms GOOD    01.10.12 10:03:18.445 myarm.info /changeset/1/...
128. HTTP  46.487 ms FAILED  01.10.12 10:03:26.271 myarm.info /wiki/newticket
129. HTTP  46.598 ms FAILED  01.10.12 10:03:26.399 myarm.info /newticket
130. HTTP  44.826 ms FAILED  01.10.12 10:03:26.529 myarm.info /ticket/newti...
```

Example output 5.5 : **myarmquery example showing transactions measured with the apache httpd server**

However the program allows to format the output of each instance by specifying a `printf()` like format string. The following sections describe the supported command line options, optional configuration file properties and the supported format specifiers for formatting a transaction

and formatting an application

output.

5.4.1   Command line options

Usage:

`myarmquery [options] [appName][:tranName] [appName2][:tranName2] ...`

appName:tranName*, *appName2:tranName2
　　specifies the application and/or transaction name(s) to be queried. For details about specifying application and/or transaction names read the section "Application and transaction names".

In addition to the options listed below, the `myarmquery` command supports standard options described in appendix "Standard options", the constraint options in appendix "Constraints options" and the sorting options in appendix "Sorting options".

5.4.1.1   Selection criteria options

`-t, --trans`
　　selects query of transaction instances, which is the default behaviour.

`-a, --apps`
　　selects query of application instances.

–D, --dropped
> selects query of dropped transaction instances

-r, --rts
> selects query of real time statistics data

-e, --rte
> selects query of runtime events

5.4.1.2   Display options

-c, --children
> indicates that each child (sub) transaction should be displayed as well. Note that this only works if you have generated the correlation information within the database by using the myarmcorrelate command.

-rm *num***, --result-max** *num*
> specifies a maximum number of application or transaction instances to display.

-ri *num***, --result-index** *num*
> specifies the number of application or transaction instances to skip and start printing with the instance at index number *num*.

-rp *num***, --result-page** *num*
> specifies the number of application or transaction instances to display per page. When the number of transactions is reached, the program will prompt you to hit the return key.

5.4.1.3   Output formatting options

In addition to the standard formatting options myarmquery also supports the following options:

-dtf *string* **, --date-fmt** *string*
> specifies the date format in *string* for formatting and parsing dates. See "Configuring date formats" for details.
>
> The default is specified by the basic.time.format configuration property.

-tmf *string* **, --time-fmt** *string*
> specifies the time format in *string* for formatting and parsing times. See "Configuring time formats" for details.
>
> The default is specified by the basic.time.date.format configuration property.

-rf *string* **, --rt-fmt** *string*
> specifies the response time format in *string* for printing and parsing response times. See "Configuring response time formats" for details.
>
> The default is specified by the tools.rtformat configuration property.

-li *string***, --level-indent** *string*
> Uses *string* instead of a normal space character for indentation.
>
> Default is " " (two spaces).

-tf *string***, --tran-format** *string*
> Uses *string* as the transaction formatting template (See below).
>
> The default is specified by the tools.query.tranformat configuration property.

-af *string***, --app-format** *string*
> Uses *string* as the application formatting template (See below).
>
> The default is specified by the tools.query.appformat configuration property.

5.4.1.4 RTS data options

The following options are used to specify additional constraints querying RTS data:

−i *ival*, **−−interval** *ival*
> defines the interval (in minutes) for retrieving (aggregating) RTS data. Possible values are 1, 5, 15, 30, 60, hour, day, week, month and year.

5.4.2 Transaction instance formatting

Formatting a transaction instance output can be modified by a so-called format string. This string specifies plain text which is directly copied one by one to the resulting output and so-called format specifiers which insert transaction specific information into the output. These are the characters which introduce a format specifier:

% introduces transaction specific insertion

&, \ introduces layout specific insertion

Also the formatting supports to the insertion if the contents of environment variables. This can be done by specifying $(var) or ${var}, where var stands for the environment variable name.

Here is the list of all supported transaction specifiers:

%a prints the arrival time of the transaction

%A prints the hostname (system address) of the transaction

%b prints the blocked time of the transaction

%B prints the detailed blocking times of the transaction

%c prints the context values associated with this transaction

%d prints the diagnostic detail string associated with the transaction

%[X]e prints the plain net response time in nanoseconds without any unit string (similar to "%r"). If X is specified, it specifies that the time is to be divided by 10 to the power X (e.g. %3e results in microsecond resolution).

%[X]E prints the plain raw response time in nanoseconds without any unit string (similar to "%R"). If X is specified, it specifies that the time is to be divided by 10 to the power X (e.g. %3E results in microsecond resolution).

%g within parent/child transaction chains it will print the percentage of the current transaction in relation to the complete chain

%G prints the application group the transaction belongs to

%h prints the hostname (system address) of the transaction

%H prints the application instance ID of the transaction (in hexadecimal encoding)

%i prints the ID of the transaction definition

%I prints the ID of the associated application definition

%J prints the application instance the transaction was executed on

%k prints the transaction instance ID, also known as the correlation ID

%K prints the parent transaction instance ID also known as the parent correlation ID

%Xm prints the X metric of the transaction, where X is a number between 1 and 7

%n prints the transaction name of the transaction

%N prints the application and transaction name delimited by a colon (':')

%p prints the start time of the transaction

%P prints the stop time of the transaction

%q prints the start date of the transaction

%Q prints the stop date of the transaction

%r prints the net response time of the transaction, that is the response time minus the blocking plus the arrival time

%R prints the raw response time without any adjustments (blocking, arrival time)

%s prints the status of the transaction as descriptive text

%S prints the one letter status of the transaction

%t prints the thread ID the transaction was executed on (if available)

%T prints the thread name the transaction was executed on (if available)

%u prints the URI of the transaction

%U prints the associated user name of the transaction, if any

%x prints the start time of the transaction as milliseconds since 1.1.1970

%X prints the stop time of the transaction as milliseconds since 1.1.1970

%z prints the normalized response time if available. Normalized response time is calculated by dividing the response time by the metric value marked as the transaction size metric.

And here the layout specifiers:

&l indent sub-transactions

&n number of current transaction

\t tabulator

\n new line

5.4.3 Application instance formatting

Formatting an application instance output can be modified by a so-called format string. This string specifies plain text which is directly copied one by one to the resulting output and so-called format specifiers which insert application specific information into the output. There are the following characters which introduce a format specifier:

% introduces application specific insertion

&, introduces layout specific insertion

Also the formatting supports to insert contents of environment variables. This can be done by specifying `$(var)` or `${var}` where `var` stands for the environment variable name.

Here is the list of all support application specifiers:

%A prints the system address of the transaction.

%c prints the context values associated with this application.

%g prints the associated group name.

%i prints the associated instance name.

%I prints the ID of the application.

%k prints the database key of the application.

%n prints the name of the application.

%p prints the start time of the application.

%P prints the stop time of the application.

%q prints the start date of the application.

%Q prints the stop date of the application.

%r prints the running time of the application.

%x prints the start time in milliseconds since 1.1.1970.

%X prints the stop time in milliseconds since 1.1.1970.

5.4.4   Configuration properties

The following properties can be used to configure the `myarmquery` tool within the standard MyARM configuration file.

`tools.rtformat`
>   Specifies the response time display entity. (See appendix "Configuring response time formats")

`tools.query.max`
>   Maximum number of results.

`tools.query.child.indent`
>   Specifies the string to use for indentation of child transactions.

`tools.query.tranformat`
>   Specifies the formatting string for transactions.
>
>   Default is "&l&n. %n\t%r\t%s\t%q %p %A %U %c %0m %1m\n".

`tools.query.appformat`
>   Specifies the formatting string for transactions.
>
>   Default is "&n. %n\t%q %p\t%r\t%A\t%g\t%i %c\n".

5.4.5 Threshold statistics

Selecting threshold statistics will output a summary of activated measurement thresholds. For example executing the following myarmquery command on our `myarm.info` measurement database will print out the following lines:

```
$ myarmquery --utc --thresholds -sf "30.10.12" -su "31.10.12"
1351616640,PyCDDB:DB-Connect,30ms,136ms,1,1
1351555860,httpd:HTTP,1000ms,8395ms,15,637
1351616640,PyCDDB:CDDB-Query,100ms,247ms,1,1
1351616640,PyCDDB:Fetch-Tracks,5ms,36ms,4,20
1351616640,PyCDDB:Generate-Output,5ms,12ms,1,1
1351616640,PyCDDB:Generate-Output,myarm.info,5ms,12ms,1,1
1351616640,PyCDDB:DB-Connect,myarm.info,30ms,136ms,1,1
1351616640,PyCDDB:CDDB-Query,myarm.info,100ms,247ms,1,1
1351555860,httpd:HTTP,myarm.info,1000ms,8395ms,15,637
1351616640,PyCDDB:Fetch-Tracks,myarm.info,5ms,36ms,4,20
```

Example output 5.6 : **myarmquery thresholds example showing threshold statistics of our myarm.info server**

The result consists of lines with columns separated with a comma charater. There are either lines with 6 or 7 columns. Lines with 6 columns represents threshold statistics for all systems and lines with 7 columns is the result for the specific system as specified in the third column (here `myarm.info`). The following list describes the columns briefly:

Lines with 6 columns:

1. UTC timestamp in seconds since epoch

2. Application and transaction name delimited by a colon

3. Threshold in milliseconds

4. Maximum response time in milliseconds of the requested interval

5. Number of measurements exceeded their threshold

6. Total number of measurements of the requested interval

Lines with 7 columns:

1. UTC timestamp in seconds since epoch

2. Application and transaction name delimited by a colon

3. System the measurements were recorded

4. Threshold in milliseconds

5. Maximum response time in milliseconds of the requested interval

6. Number of measurements exceeded their threshold

7. Total number of measurements of the requested interval

5.4.6   See Also

myarmoptions, myarmdefinition, myarmcorrelate, myarmdelete, myarmconfig

5.5   myarmcorrelate – create transaction correlation information

This program creates the transaction correlation information needed to process parent/child relationships.

5.5.1   Command line options

Usage:

```
myarmcorrelate [options]
```

In addition to the options listed below, the `myarmcorrelate` command supports standard options described in appendix "Standard options".

-sf *time*, **--start-from** *time*
>   specifies the start time from which to correlate transactions.

-su *time*, **--start-until** *time*
>   specifies the start time until which to correlate transactions.

-q, --quiet
>   set quiet output mode, e.g. do not print diagnostic information.

5.5.2   See Also

myarmoptions, myarmquery, myarmconfig

5.6   myarmdelete – delete application and/or transaction data

The `myarmdelete` command line tool deletes application and/or transaction instances as well as registered ARM definitions.

5.6.1   Command line options

Usage:

```
myarmdelete [options] [name][:part] [name2][:part2] ...
```

name:part**, **name2:part2
>   specifies the name(s) to be deleted. If a transaction entity is selected, names have to be specified in the form `<appName>:<tranName>`. For details about specifying application and/or transaction names read the section "Application and transaction names".

In addition to the listed options below, the `myarmdelete` command supports standard options described in appendix "Standard options" and the constraint options in appendix "Constraints options".

5.6.1.1   Entity selection options

With one of the following entity selection options, an entity is selected for deleting. If a name is omitted, all instances of the appropriate entity are deleted. Otherwise only the instances with the name(s) provided are deleted.

–a, --apps
>   selects application instances to be deleted.

–t, --trans
>   selects transaction instances to be deleted.

–u, --users
>   selects users to be deleted.

–s, --systems
>   selects system addresses to be deleted.

–r, --rts
>   selects real time statistics (RTS) to be deleted.

5.6.1.2   Other options

–v, --verbose   set verbose output mode, prints more diagnostic information

--all
>   selects all instances of the selected entities to be deleted.

–d, --defs
>   deletes also the definition data of the selected entities.

5.6.2   Examples

The `myarmdelete` command can be used in many ways. If you want to delete all query transaction measurements of our CDDB example Server, use the following command:

```
myarmdelete -t "PyCDDB:CDDB-Query"
```

and the following command for the applications

```
myarmdelete -a PyCDDB
```

If you want to clean up your whole database use

```
myarmdelete --defs --all
```

deleting all (`--all`) instances as well as all definitions (`--defs`).

5.6.3   See Also

myarmoptions, myarmdefinition, myarmquery, myarmconfig

5.7 myarmarchive – queries transaction data from a MyARM archive

The `myarmarchive` command line tool is used to query transaction instance data from a MyARM archive managed by a running `myarmdaemon`. It uses a TCP socket protocol to query the transaction data from the `myarmdaemon`.

5.7.1 Command line options

Usage:

```
myarmarchive [options]
```

In addition to the listed options below, the `myarmarchive` command supports standard options described in appendix "Standard options".

–rd data, ––reqdata data
> specifies the data type to query from the archive. Currently the following data types are supported:

> **query**
>> queries formatted transaction data instances from the archive

> **list**
>> lists file names stored in the archive

–tf *string*, **––tran-format** *string*
> Uses *string* as the transaction formatting template (See myarmquery transaction formatting).

––threads *number*
> Specifies the *number* of threads used to process the query within the `myarmdaemon`.

–sf *time*, **––start-from** *time*
> specifies the start time from which to select transactions.

–su *time*, **––start-until** *time*
> specifies the start time until transactions should match.

–ts *tran-status*, **––tran-status** *tran-status*
> specifies the ARM transaction status to be matched.

––rt-min *min*
> specifies the minimum response time to match a transaction.

––rt-max *max*
> specifies the maximum response time to match a transaction.

5.7.2 See Also

myarmoptions, myarmdefinition, myarmquery, myarmconfig

5.8   myarmstat – computes statistical metrics of transactions

The `myarmstat` command line tool is used to get statistical metrics of transaction instances. It calculates statistics for each specified transaction separately unless the `--all` is specified.

5.8.1   Command line options

Usage:

```
myarmstat [options] [appName][:tranName] [appName2][:tranName2] ...
```

appName:tranName*, *appName2:tranName2
> If specified only calculates statistics for instances of ARM transactions which matches this application and/or transaction name. For details about specifying application and/or transaction names read the section "Application and transaction names".

In addition to the listed options below, the `myarmstat` command supports standard options described in appendix "Standard options" and the constraint options in appendix "Constraints options".

5.8.1.1   Other Options

--all
> process all specified application or transaction names in only one single statistic instead of a single statistic for each specified name.

--verbose
> show also status statistics even the count is zero

5.8.2   Example

```
ruppert@myarm /opt/myarm $ myarmstat httpd:HTTP
Statistics for httpd:HTTP transaction instances (ms)
Status      Count       Mean        Min        Max      Median    Deviation
Good        17009    269.502      0.132 347430.529      32.795    3721.576
Aborted        85   2986.504     13.392  47517.968     228.046    9756.655
Failed        813    744.059      0.173   2115.616     146.203     904.046
All         17907    303.945      0.132 347430.529      43.494    3699.812
```

Example output 5.7 : **myarmstat showing overall statistic for all HTTP transactions of an web-server**

5.8.3   See Also

myarmoptions, myarmquery, myarmchain, myarmconfig

5.9 myarmchain – computes statistical metrics of transactions chains

This program analyse transaction chains and computes statistical metrics. Therefore it scans the current configured database for the (given) parent transactions and analysis the structure of any found transaction chain. For each identical transaction chain (with exact the same structure) it measures the mean, median, min, max, deviation and relative start time for each transaction within the chain.

Note this program only works if the correlation information was generated before by calling the `myarmcorrelate` program.

5.9.1 Command line options

Usage:

```
myarmchain [options] [appName][:tranName] [appName2][:tranName2]
```

appName*:*tranName*, *appName2*:*tranName2
    specifies the application and/or transaction name to be the parent transaction for any transaction chain to analyse. If only an application name is specified all transactions of this applications are interpreted as the parent transaction for any chain to analyse. For details about specifying application and/or transaction names read the section "Application and transaction names".

In addition to the options listed below, the `myarmchain` command supports standard options described in appendix "Standard options" and the constraint options in appendix "Constraints options".

−rm *num*, **−−result−max** *num*
    specifies a maximum number of parent transactions for analysing their chains.

−ri *num*, **−−result−index** *num*
    specifies the number of parent transaction to skip, starting analysing from parent transaction number *num*

5.9.2 See Also

myarmoptions, myarmcorrelate, myarmdefinition, myarmquery, myarmconfig

5.10   myarmexport – exports ARM data to a database or file

The `myarmexport` command line tool is used to export any ARM data from one database type to another, to a XML or a MyARM data file.

5.10.1   Command line options

Usage:

```
myarmexport [options] export-url [appName][:tranName] [appName2][:tranName2]
```

export-url
>   URL-like database specification where to export the data to. See appendix "Database URL notation".

appName:tranName*, *appName2:tranName2
>   If specified, only export instances and/or definitions of ARM applications which match this application and/or transaction name. For details about specifying application and/or transaction names read the section "Application and transaction names".

In addition to the options listed below, the `myarmexport` command supports standard options described in appendix "Standard options" and the constraint options in appendix "Constraints options".

5.10.1.1   Entity selection options

With one or more of the following entity selection options entities are selected for exporting. If a name is omitted all instances of the appropriate entity are exported. Otherwise only the instances with the name(s) provided are exported.

-a, --apps
>   selects application instances to be exported.

-t, --trans
>   selects transactions instances to be exported.

-m, --metrics
>   selects system addresses to be exported.

-u, --users
>   selects users to be exported.

-s, --systems
>   selects system addresses to be exported.

-r, --rts
>   selects real time statistics (RTS) data to be exported.

-e, --rte
>   selects real time events to be exported.

-l, --logs
>   selects log messages to be exported.

-c, --cfg
>   selects runtime configuration to be exported.

5.10.1.2 Other options

--all
> selects all ARM data to be exported.

-d, --defs
> selects only ARM definitions for export. Without this option both definitions and instances are exported.

-e *name***, --engine** *name*
> selects the storage engine to use for the created database. Currently this is only supported for the MySQL database.

5.10.1.3 Example

```
ruppert@myarm /opt/myarm/data $ myarmexport --all myarm:///data.myarm
Exported entities:
Applications: Definitions 1 (Properties 0), Instances 4 (Properties 0)
Transactions: Definitions 3 (Properties 0), Instances 40 (Properties 0)
Metrics     : Definitions 0, Instances 0
Systems     : 1
Users       : 0
Logs        : 236
ruppert@myarm /opt/myarm/data $ ls -l data.myarm
-rw-r--r-- 1 ruppert ruppert 33541 May 12 13:51 data.myarm
```

Example output 5.8 : **myarmexport export ARM data into a MyARM data file**

5.10.2 See Also

myarmoptions, myarmimport, myarmdefinition, myarmquery, myarmconfig, myarmmisc

5.11   myarmimport – imports ARM data from a database or XML file

The `myarmimport` command line tool is used to import any ARM data from a database or a XML file into the currently configured database.

5.11.1   Command line options

Usage:

```
myarmimport [options] import-url [appName][:tranName] \
                                 [appName2][:tranName2]
```

import-url
> URL-like database specification from where to import. See "Database URL notation".

appName:tranName, ***appName2:tranName2***
> If specified, only import instances and/or definitions of ARM applications which match this application and/or transaction name. For details about specifying application and/or transaction names read the section "Application and transaction names".

In addition to the options listed below, the `myarmimport` command supports standard options described in appendix "Standard options" and the constraint options in appendix "Constraints options".

5.11.1.1   Entity selection options

With one or more of the following entity selection options entities are selected for importing. If a name is omitted, all instances of the appropriate entity are imported. Otherwise only the instances with the name(s) provided are imported.

-a, --apps
> selects application instances to be imported.

-t, --trans
> selects transactions instances to be imported.

-m, --metrics
> selects system addresses to be imported.

-u, --users
> selects users to be imported.

-s, --systems
> selects system addresses to be imported.

-r, --rts
> selects real time statistics (RTS) data to be imported.

-e, --rte
> selects real time events to be imported.

-l, --logs
> selects log messages to be imported.

-c, --cfg
> selects runtime configuration to be imported.

5.11.1.2   Other options

–all
> selects all ARM data to be imported.

–d, –defs
> selects only ARM definition data to be imported. Without this option both definition and instance
> data are imported.

5.11.2   Usage example

The following example shows the output of an import of some transaction data from a XML file called
`chain.xml` in the current directory.

```
ruppert@debian:/opt/myarm$ myarmimport -t xml:///chain.xml
Imported entities:
Applications: Definitions 1 (Properties 0), Instances 0 (Properties 0)
Transactions: Definitions 3 (Properties 0), Instances 160 (Properties 0)
Metrics     : Definitions 0, Instances 0
Systems     : 0
Users       : 0
```

Example output 5.9 : **myarmimport example showing the import of transaction data.**

The output of the `myarmimport` program gives an overview which and how many entities (definitions
and instances) were imported.

5.11.3   See Also

myarmoptions, myarmexport, myarmdefinition, myarmquery, myarmconfig, myarmmisc

5.12   myarmlog – displays log messages stored in the database

The `myarmlog` command line tool is used to retrieve logging messages from the configured database. MyARM can be configured to write log messages also to the database besides the standard log message destination. For example the standard MyARM configuration will write any error log message to the database. These messages can be displayed with this command.

5.12.1   Command line options

Usage:

```
myarmlog [options] [pattern]
```

pattern

     specifies a pattern to filter the log messages. For example "`myhost`" will only display log messages from the host `myhost`.

In addition to the options listed below, the `myarmlog` command supports standard options described in appendix "Standard options"

−l *level*, **−−level** *level*

     specifies the log level of messages to retrieve or delete. See Configuring Log facility for details.

−d, −−delete

     deletes log messages instead of retrieving messages

−f "*date time*", **−−from** "*date time*"

     selects log messages starting from "*date time*". See Configuring date and time formats.

−u "*date time*", **−−until** "*date time*"

     selects log messages until "*date time*". See Configuring date and time formats.

−rm *num*, **−−result−max** *num*

     specifies a maximum number of log messages to display.

−ri *num*, **−−result−index** *num*

     specifies the number of log messages to skip and start printing with the log messages at index number *num*.

−desc

     retrieve and print log message in descending order.

5.12.2   See Also

myarmoptions, myarmconfig

5.13   myarmdaemoncmd – sends commands to a daemon process

The `myarmdaemoncmd` command line tool is used to send commands to a myarmdaemon process. With these commands it is possible to get information about the current status of a `myarmdaemon`, to stop it remotely or to set thresholds

5.13.1   Command line options

Usage:

```
myarmdaemoncmd [options] command
```

The `myarmdaemoncmd` command supports the standard options described in appendix "Standard options" and the following options:

−l *num*, **−−loop** *num*
> executes the specified command within a loop *num* times

−r *num*, **−−retry** *num*
> If execution fails, retry *num* times before terminating

−w *secs*, **−−wait** *secs*
> Wait secs seconds between a retry or a loop iteration

The *command* argument specifies the command to be executed as described in the "myarmdaemon commands" section.

5.13.2   See Also

myarmoptions, myarmconfig, myarmdaemon

Chapter 6

Manager

6.1 Introduction

The MyARM-Manager gives a MyARM user the ability to get browse through measured transaction data, change administration configuration and monitor and browse so-called real time statistics (RTS). It provides a standalone user interface of the appropriate MyARM web applications as described in "Web user interface" section.
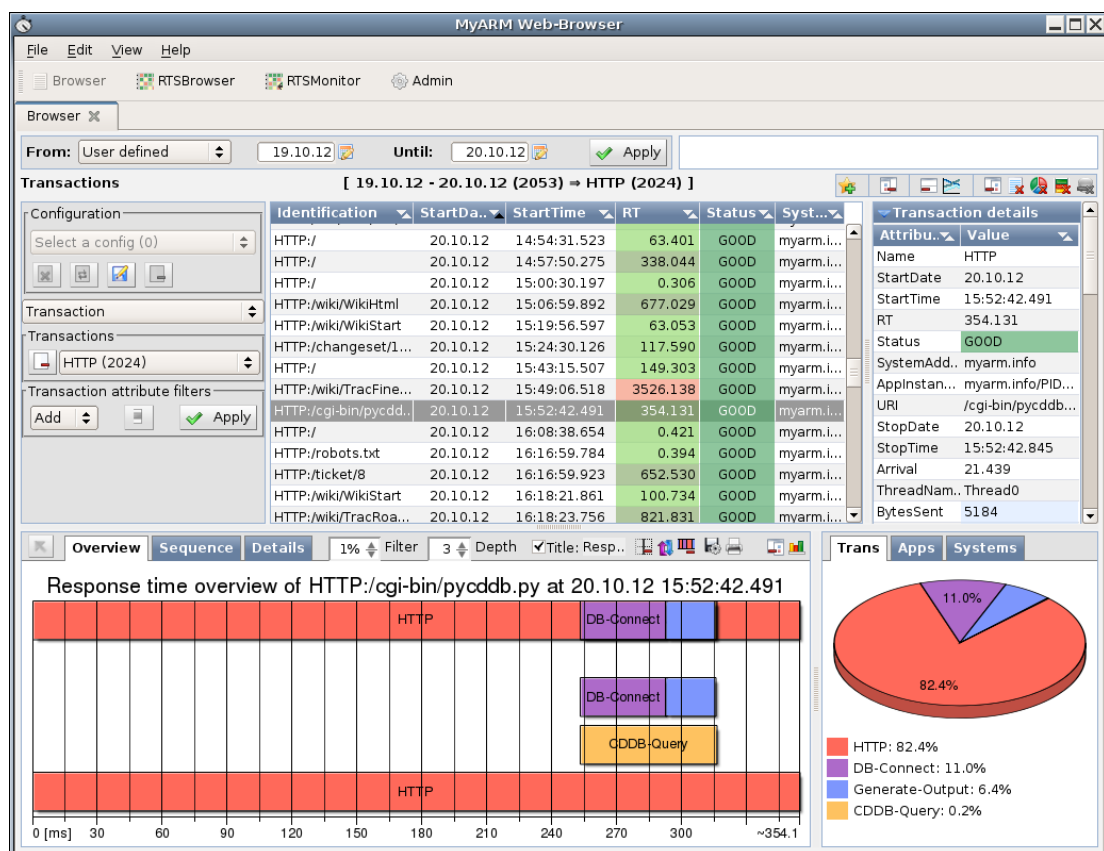
Figure 6.1: The MyARM-Manager with browser tab opened

In Figure 6.1 the "Browser" web application is opened as a tab in the MyARM-Manager.

6.2 Command line options

The `myarmmanager` can be invoked with different command line options to change it start behaviour.
Usage:

```
myarmmanager [options]
```

6.2.1 Options

The `myarmmanager` command supports the standard options described in appendix "Standard options" and the following manager specific options:

-at, --admin-tab
> Opens the Administration tab initially

-bt, --browser-tab
> Opens the Browser tab initially (default)

-rbt, --rtsbrowser-tab
> Opens the RTS-Browser tab initially

-rmt, --rtsmonitor-tab
> Opens the RTS-Monitor tab initially

-nt, --no-tab
> Do not open a tab initially

--base-url *url*
> Specifies a base *url* where the MyARM web-applications are deployed. For example: *http://myarm.info/fcgi-bin/* can be used to access our myarm.info demo web-applications

-mi, --myarm-info
> Runs the myarmmanager to browse and monitor the myarm.info demo web-site

--admin-url *url*
> URL where the MyARM Administration web application is deployed

--browser-url *url*
> URL where the MyARM Browser web application is deployed

--rtsbrowser-url *url*
> URL where the MyARM RTS-Browser web application is deployed

--rtsmonitor-url *url*
> URL where the MyARM RTS-Monitor web application is deployed

--browser-config *config*, **--bcfg** *config*
> Pass the provided configuration name to the MyARM Browser web application to initially load this configuration

6.2.2 See Also

myarmoptions, myarmconfig

6.3   Menu

6.3.1   The 'File' Menu
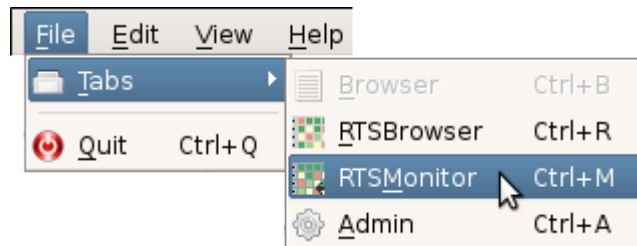
Figure 6.2: The 'File' Menu

6.3.1.1   Tabs

A sub-menu used to open a new tab. "Tabs Toolbar" for details.

6.3.1.2   Quit

Quits the running MyARM-Manager.

6.3.2   The 'Edit' Menu
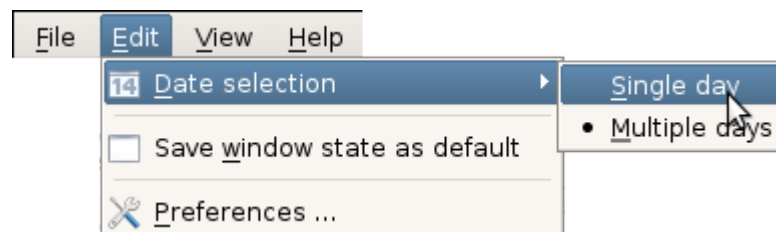
Figure 6.3: The 'Edit' Menu

6.3.2.1   Date selection

The "Edit / Date selection" is used to switch between single and multiple day selection within the "Browser" tab.

6.3.2.2   Save window state as default

Saves the current window state (position, dimension, etc) as the default. This state will be used later on to initialize the MyARM-Manager.

6.3.2.3   Preferences ...

The "Edit / Preferences ..." menu item will open the preferences dialog of the web application in the current tab.

6.3.3 The 'View' Menu
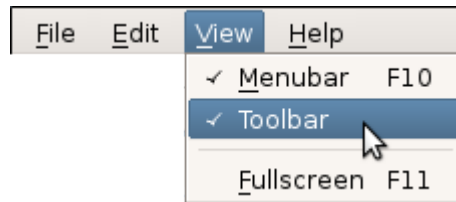
Figure 6.4: The 'View' Menu

6.3.3.1 Menubar

The "View / Menubar" toggles the visibility of the menubar.

6.3.3.2 Toolbar

The "View / Toolbar" toggles the visibility of the toolbar.

6.3.3.3 Fullscreen

The "View / Fullscreen" toggles the full screen mode.

6.3.4 The 'Help' Menu
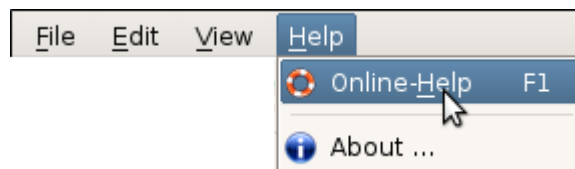
Figure 6.5: The 'Help' Menu

6.3.4.1 Online-Help

The "Help / Online-Help" opens the online help with the default system web-browser.

6.3.4.2 About

The "Help / About" opens the about dialog.

6.4 Tabs Toolbar
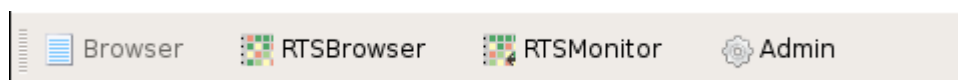
Figure 6.6: The 'Tabs' Toolbar

The following web applications can be opened (if available) in a tab by clicking on the appropriate toolbar button:

"Browser"

"RTS Browser"

"RTS Monitor"

"Admin"

It starts the appropriate web application on the local host and connects to it.

Since the MyARM-Manager is started on behalf of the current user its user name is used to log into the MyARM web application. The web application hides the menu and user name.

If only the "Browser" web application is available (Community Edition) the tabs toolbar can not be used and it is not shown at all.

6.5   RTS-Browser

This section will show you how to use the `myarmrtsbrowser` interface in order to do ARM-related statistic analysis of your applications. Point your browser to the URL identifying the MyARM RTS-Browser web interface. The exact URL you have to use must include the host name of the server running the FastCGI process. If your web browser executes on the node where the web browsing interface has been installed, you can use the URL http://localhost/fcgi-bin/myarmrtsbrowser.fcgi.

If you use the standalone version of the MyARM web browser you can use the URL http://localhost:8082/.

Figure 6.7: The MyARM RTS-Browser

The next section will give you an overview regarding the major parts making up the `myarmrtsbrowser` interface including most important input controls. We will then follow up with the description of a typical user session, showing you what you would see in your web browser. The last section will provide a reference of all concepts of the `myarmrtsbrowser`.

6.5.1   Layout

The user interface is really easy to use, the following paragraphs will give you an overview over the different parts in the GUI and this is how a MyARM RTS-Browser web interface looks like:
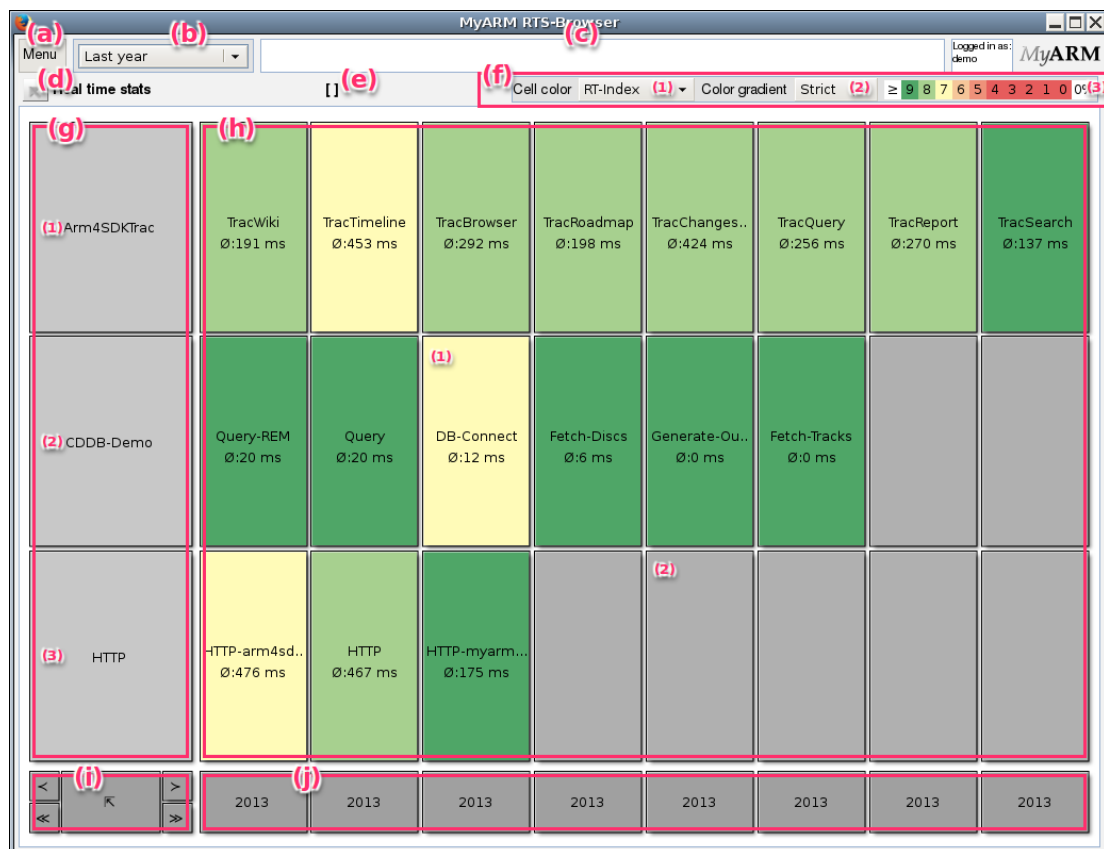
Figure 6.8: The MyARM RTS-Browser layout

(a) The "`Menu`" is used to change settings that affect the whole application such as changing your user name and preferences for displaying data. Those preferences can be saved and loaded from the database.

(b) Within the top-view of the `myarmrtsbrowser` a time period can be selected to start browsing the real time statistics. Here the "`Last year`" ("`2013`") is selected. See the "Time period" drop-down-box for details.

(c) The "`Status-Line`". Error messages related to processing your request and also informational messages (loading data etc.) are displayed here.

(d) The ⬏ button is used to go a level up if current view was drilled down.

(e) Within the brackets the complete drilled down navigation path is displayed.

(f) The "**Cell color area**" is used to select the coloring of the transaction data cells:

    (1) The cell color drop-down-box to select the used statistic index for coloring the transaction statistic cells ("h"). Here the "`RT-Index`" response time index is used.

    (2) The cell color gradient drop-down-box is used to select the strength of the chosen coloring. Here the "`Normal`" color gradient is used.

    (3) The color gradient bar displays the current coloring distribution from left greater than 0% to right greater than 90%.

(g) The "**Row title area**". Depending on the current level general information is displayed in each cell. Within the top group level the cells displays the names of RTS groups:

(1) "`Arm4SDKTrac`" groups any real time statistic regarding HTTP measurements of the Trac instal-lation at https://arm4sdk.org/.

(2) "`CDDB-Demo`" groups any real time statistic regarding measurements of our CDDB-Demo run-ning at https://myarm.info/.

(3) "`HTTP`" groups HTTP measurements of all virtual hosts running at https://arm4sdk.org/.

(h) The "**Transaction statistic area**". Depending on the current level statistics are shown for a specific transaction within a defined time period as shown in ("i").

(1) "`DB-Connect`" shows statistic information about the database connection measurement within the our CDDB-Demo.

(2) Empty cells where no data exists are rendered in a gray color.

(i) The "**Time period navigation area**". Within this area buttons are provided to navigate through the time line of measured statistics. Cells rendered in dark gray color are disabled and can not be clicked since there is no data. See the "Time period navigation" buttons for details.

(j) The "**Time period area**". The cells in this area are used to drill down and select the displayed time period.

6.5.2   Getting started

This section describes how to use and find possible performance bottlenecks within your complete infras-tructure. The following section describes step by step how to dig into the statistic data. Lets start with the top level view of the `myarmrtsbrowser` which displays on overview of different transaction statistics grouped into different application categories:

1. "`Arm4SDKTrac`" groups any real time statistic regarding HTTP measurements of the Trac installa-tion at https://arm4sdk.org/.

2. "`CDDB-Demo`" groups any real time statistic regarding measurements of our CDDB-Demo running at https://myarm.info/.

3. "`HTTP`" groups HTTP measurements of all virtual hosts running at https://arm4sdk.org/.

Now we will focus on our CDDB-Demo measurements and here especially the "`DB-Connect`" statistic since the cell is rendered in yellow indicating not optimal performance as shown in Figure 6.8.

To get a better impression whats going on here we click on the "`CDDB-Demo`" cell within the "**Row title area**". This will change the view to display all real time statistics from "`CDDB-Demo`" group. The real time statistic names are now rendered in the "**Row title area**" (Y-Axis) and in the "**Time period area**" (X-Axis) each month of the year "`2013`" is displayed:

	Jan/2013	Feb/2013	Mar/2013	Apr/2013	May/2013	Jun/2013	Jul/2013	Aug/2013	Sep/2013	Oct/2013	Nov/2013	Dec/2013	Total
Count [#]	862	2894	8503	7381	8258	8834	8013	9565	9120	5841	7820	13278	90369
Query-REM	6	18	31	34	40	45	36	41	49	23	34	70	427
Query	42	166	444	394	441	455	408	491	468	311	406	692	4718
DB-Connect	42	166	444	394	441	455	408	491	468	311	406	692	4718
Fetch-Discs	41	162	437	388	435	445	406	486	467	311	405	691	4674
Generate-Output	41	162	437	388	435	445	406	486	467	311	405	691	4674
Fetch-Tracks	690	2220	6710	5783	6466	6989	6349	7570	7201	4574	6164	10442	71158

< 2013 > (Jan/2013 … Dec/2013, Total)

Figure 6.9: CDDB-Demo overview of year 2013

As the color indicates here the "`DB-Connect`" measurements are not in the range we expected. Therefore we click on the "`DB-Connect`" cell to dig into these measurements:

DB-Connect

Tran: DB-Connect

	Jan/2013	Feb/2013	Mar/2013	Apr/2013	May/2013	Jun/2013	Jul/2013	Aug/2013	Sep/2013	Oct/2013	Nov/2013	Dec/2013	Total
Ø [ms]:	15 ms	11 ms	14 ms	16 ms	10 ms	10 ms	10 ms	10 ms	10 ms	11 ms	12 ms	12 ms	12 ms
σ [ms]:	14 ms	10 ms	14 ms	31 ms	4 ms	2 ms	2 ms	4 ms	5 ms	8 ms	1 ms	1 ms	11 ms
Min [ms]:	9 ms	9 ms	9 ms	9 ms	9 ms	9 ms	9 ms	9 ms	9 ms	9 ms	12 ms	12 ms	9 ms
Max [ms]:	75 ms	147 ms	149 ms	502 ms	106 ms	58 ms	46 ms	98 ms	71 ms	139 ms	19 ms	25 ms	502 ms
Status													
Good [#]:	42	166	444	394	441	455	408	491	468	311	406	692	4718
Failed [#]:	0	0	0	0	0	0	0	0	0	0	0	0	0
Aborted [#]:	0	0	0	0	0	0	0	0	0	0	0	0	0
Unknown [#]:	0	0	0	0	0	0	0	0	0	0	0	0	0
Not stopped [#]:	0	0	0	0	0	0	0	0	0	0	0	0	0
Total [#]:	42	166	444	394	441	455	408	491	468	311	406	692	4718
Index [%]:	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
RT thresholds													
RT ≤ 10 ms [#]:	18	107	167	221	309	388	323	419	382	201	0	0	2535
RT ≤ 50 ms [#]:	21	58	262	154	131	66	85	71	83	106	406	692	2135
RT ≤ 100 ms [#]:	3	0	13	13	0	1	0	1	3	3	0	0	37
RT > 100 ms [#]:	0	1	2	6	1	0	0	0	0	1	0	0	11
Total [#]:	42	166	444	394	441	455	408	491	468	311	406	692	4718
Index [%]:	69.6	81.9	67.8	76.5	84.9	92.6	89.6	92.6	90.7	81.9	50.0	50.0	76.6

< 2013 > (Jan/2013 … Dec/2013, Total)

Figure 6.10: CDDB-Demo DB-Connect details of year 2013

As we can see here the last two month are rendered in deep red indicating that something changed. So we will dig into the November 2013 data:

Figure 6.11: CDDB-Demo DB-Connect details of November 2013

The response time threshold colors shows that the problem exists since beginning of November thus we need to browse back to October 2013:

Figure 6.12: CDDB-Demo DB-Connect details of October 2013

Now we finally see that between 23th and 24th of October something changed here. Maybe a new database version was installed or the average load of the system increased since 24th of October resulting in a slightly higher "DB-Connect" response time.

Checking the MySQL database server we can see that on 24th of October 2013 something changed in the MySQL configuration:

```
myarm:~# ls -lrt /etc/mysql/
insgesamt 24
-rw------- 1 root root  312 Jan 05 2009 15:59 debian.cnf
-rwxr-xr-x 1 root root 1198 Aug 27 2009 12:32 debian-start
-rw-r--r-- 1 root root 3972 Okt 10 2012 15:42 my.cnf.dpkg-old
-rw-r--r-- 1 root root 3972 Okt 24 2013 08:48 my.cnf.old
drwxr-xr-x 2 root root 4096 Okt 24 2013 08:48 conf.d
-rw-r--r-- 1 root root 3612 Okt 24 2013 08:50 my.cnf
```

Configuration 6.1: **myarm.info MySQL configuration change**

After further analysis the mysql-common and python-mysqldb debian packages were upgraded which seems to cause the degradation of response times!

6.5.3 Drill down

The `myarmrtsbrowser` main concept is to drill down from a top level view into details about various transaction statistics. To do so the `myarmrtsbrowser` provides a drill down for defined time periods.

6.5.3.1 The group view

The starting point is the top level view also called "RTS Group view". The time period for this view can be specified using the "Time period" drop-down-box. Within our example we used "Last year" ("2013"):
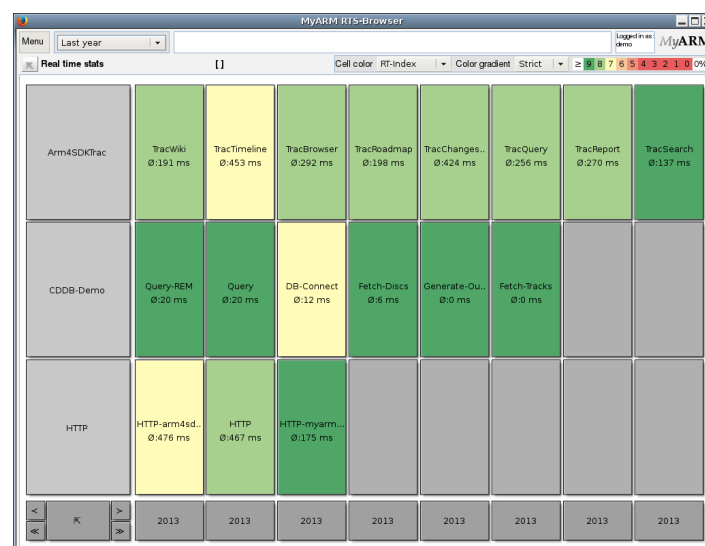
Figure 6.13: The group view

Depending on the selected "Time period" the next drill down level will show years or months.

6.5.3.2 The years view

The following example shows the years "2012" until "2014":
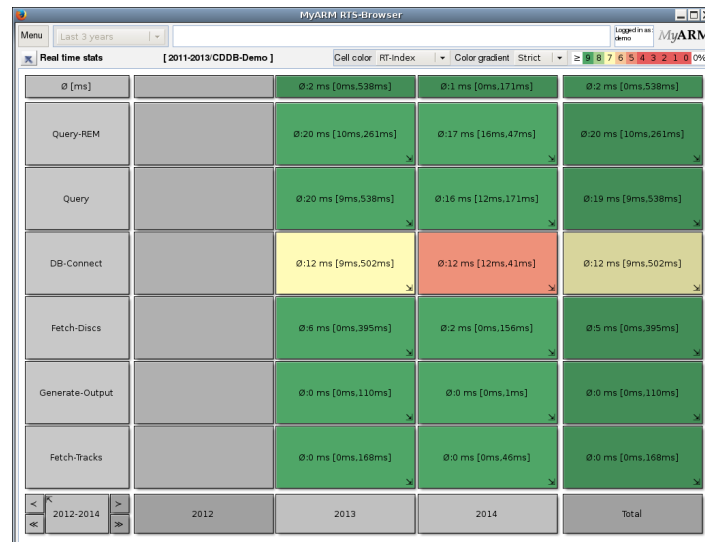
Figure 6.14: The years view

The year "2012" has no data therefore the cells are rendered in dark gray. For the years "2013" and "2014" the average and in brackets the minimum and maximum response time is presented.

6.5.3.3 The months view

To further drill down we clicked on the "2013" cell. The following figure shows data from "December 2012" until "November 2013" since we moved the view by one month into the past be clicking the "<" in right bottom corner:
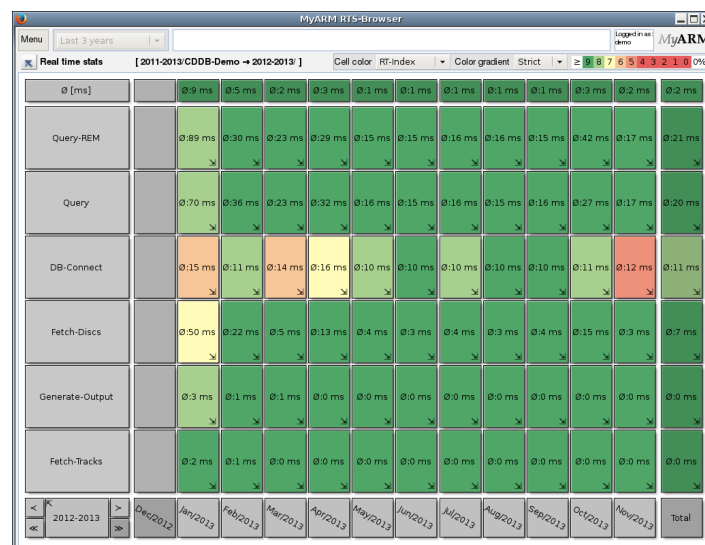
Figure 6.15: The months view

We know that there is no data within "`2012`" the cells for "`December 2012`" are rendered in dark gray.

6.5.3.4   The days view

Now we pick up "`April 2013`" to step deeper into the data:

Figure 6.16: The days view

Within the days view there are two days rendered with slightly more space between the other days. These two days are "`Saturday`" and "`Sunday`". Therefore its possible to see the difference between working days and the weekend at a glance.

6.5.3.5   The hours view
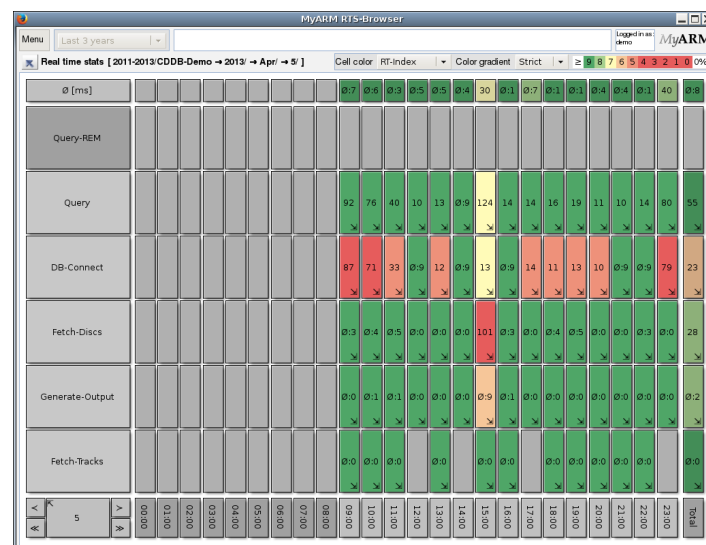
Figure 6.17: The hours view

We drilled down into "5th April" and we can see here measurements are recorded after 9am until 12pm and this days no one searched for REM within our CDDB-Demo.

6.5.3.6 The minutes view

The last time period view is the minutes view and here we selected "15:00" (3pm):
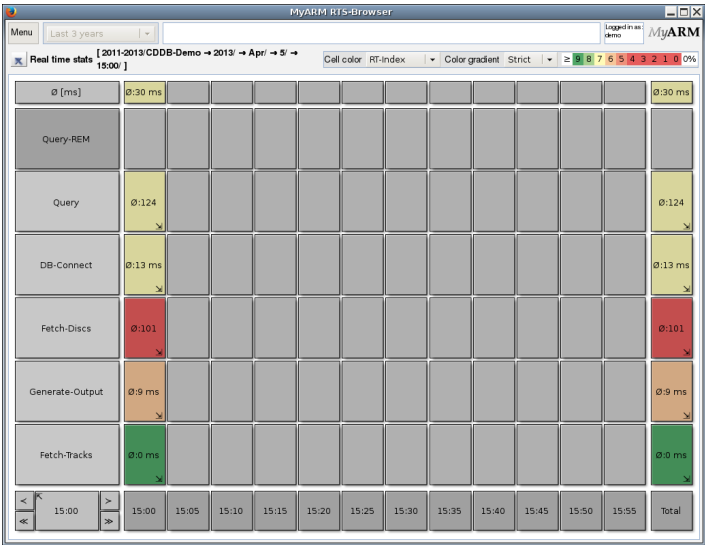
Figure 6.18: The minutes view

6.5.3.7 The details view

Within any time period view it is possible to drill down to the details view by clicking on the "DB-Connect" in the "**Row title area**". The following example shows the details of "2013":

Figure 6.19: The details view of DB-Connect for 2013

6.5.4 The Menu
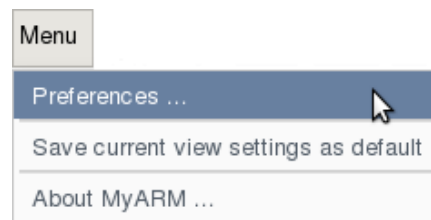
Figure 6.20: The menu

6.5.4.1 Preferences

The menu item labelled "`Menu / Preferences...`" opens a dialog for changing your preferences. The preference dialog is described in the Preferences section of the `myarmadmin` application.

6.5.4.2 Save current view settings as default

Saves the current view settings (e.g. time period, cell coloring, etc.) of the `myarmrtsbrowser`.

6.5.4.3 About

Selecting the menu item "`Menu / About MyARM ...`" will open a window displaying an informational text about the MyARM (e.g. version, edition, contact address, etc) (see the browser "About" menu).

6.5.5 Control buttons

6.5.5.1 Time period

The top level view of the `myarmrtsbrowser` is the starting point for browsing through transaction statistics data. To define the starting time period the following time period drop-down-box is provided:
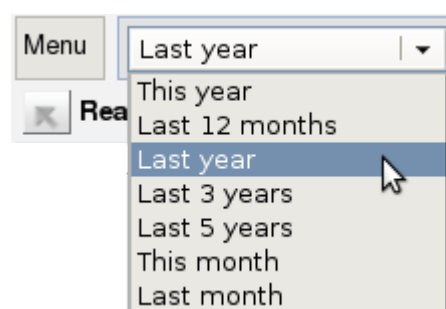
Figure 6.21: The time period drop-down-box

"This year"
> will aggregate all available data of this year. The next drill-down level will be all month of this year.

"Last 12 months"
> will aggregate all available data of the last 12 month. The next drill-down level will be the last 12 month.

"Last year"
> will aggregate all available data of the last year. The next drill-down level will be the 12 month of last year.

"Last 3 years"
> will aggregate all available data of the last 3 years. The next drill-down level will be the last three years.

"Last 5 years"
> will aggregate all available data of the last 5 years. The next drill-down level will be the last five years.

"This month"
> will aggregate all available data of the this month. The next drill-down level will be all days of this month.

"Last month"
> will aggregate all available data of the last month. The next drill-down level will be all days of last month.

6.5.5.2   Cell coloring

To support identifying problematic statistics at a glance different colors are used to represent the overall statistic of the different transaction measurements. Figure 6.22 shows all controls to adopt cell coloring:
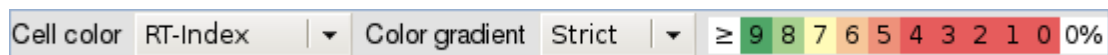
Figure 6.22: The cell coloring bar

The following different cell coloring's based on transaction statistics are supported:

"Neutral"
> no cell coloring

"RT-Index"
> cell coloring according to the response time index calculated by the number of measurements within the defined response thresholds.

"Status-Index"
> cell coloring according to the status index calculated by the number of measurements within the transaction status ("GOOD", "FAILED", etc).

With the following controls the user can slightly change the distribution of the colors:

"Scrict"
> Interprets the selected color index real strict. Values between 0% and 60% are displayed in red, values between 60% and 70% are displayed in orange, values over 70% in yellow and values above 80% are displayed in light and dark green:
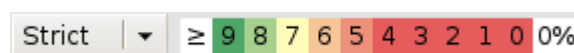
Figure 6.23: Strict cell coloring

"Normal"

Values between 0% and 40% are displayed in red, values between 40% and 60% are displayed in orange, values over 60% in yellow and values above 70% are displayed in green gradients:
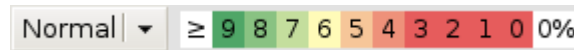
Figure 6.24: Normal cell coloring

"Tolerant"

Values between 0% and 30% are displayed in red, values between 30% and 50% are displayed in orange, values over 50% in yellow and values above 60% are displayed in green gradients:
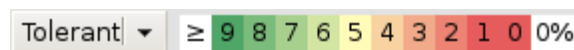
Figure 6.25: Tolerant cell coloring

"Linear"

Values between 0% and 20% are displayed in red, values between 20% and 40% are displayed in orange, values over 40% in yellow and values above 50% are displayed in green gradients:
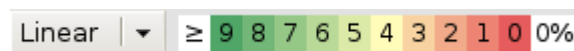
Figure 6.26: Linear cell coloring

6.5.5.3   Time period navigation

The time period navigation buttons in the left bottom corner of the `myarmrtsbrowser` is used to navigate through the time line in the past and future direction. Depending on the current time period it moves the view by minutes, hours, days, months or years.

Figure 6.27: The time period navigation buttons

(1) By clicking this button the time view is moved by one unit into the past. For example in the hours view it moves the view by one hour into the past.

(2) By clicking this button the time view is moved by one unit into the future. For example in the days view it moves the view by one day into the future.

(3) By clicking this button the time view is moved by one complete view into the past. For example in the days view it moves the view by one month into the past.

(4) By clicking this button the time view is moved by one complete view into the future. For example in the months view it moves the view by one year into the future.

(5) Shows the currently viewed time period. By clicking this button the view is moved one level up.

6.5.5.4 Dig into raw data

Within each cell in the right bottom corner there is an arrow pointing east south. Clicking this arrow will open a new browser tab or windows and load the raw data using the `myarmbrowser` application as shown in the following figure.

Note that you need to configure the correct base URL for the `myarmbrowser` as described in section Configuring web user interfaces.
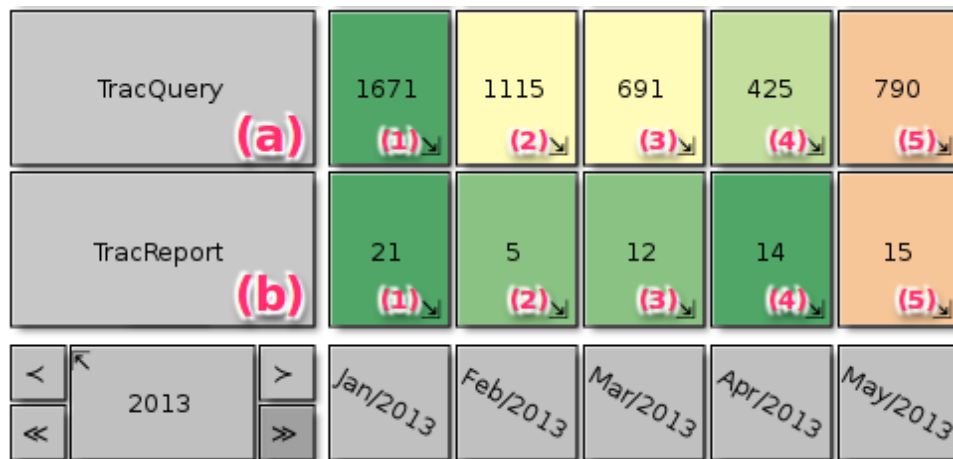
Figure 6.28: The dig into raw data arrows

(a) Represents all measurements for the HTTP request of the Trac query interface:

  (1) By clicking this button all raw `1671` measurements recorded in January 2013 of the "`TracQuery`" are loaded into a new browser tab.

  (2) By clicking this button all raw `1671` measurements recorded in February 2013 of the "`TracQuery`" are loaded into a new browser tab.

  (3) By clicking this button all raw `691` measurements recorded in march 2013 of the "`TracQuery`" are loaded into a new browser tab.

  (4) By clicking this button all raw `425` measurements recorded in April 2013 of the "`TracQuery`" are loaded into a new browser tab.

  (5) By clicking this button all raw `790` measurements recorded in may 2013 of the "`TracQuery`" are loaded into a new browser tab.

(b) Represents all measurements for the HTTP request of the Trac report interface:

  (1) By clicking this button all raw `21` measurements recorded in January 2013 of the "`TracReport`" are loaded into a new browser tab.

  (2) By clicking this button all raw `5` measurements recorded in February 2013 of the "`TracReport`" are loaded into a new browser tab.

  (3) By clicking this button all raw `12` measurements recorded in march 2013 of the "`TracReport`" are loaded into a new browser tab.

  (4) By clicking this button all raw `14` measurements recorded in April 2013 of the "`TracReport`" are loaded into a new browser tab.

  (5) By clicking this button all raw `15` measurements recorded in may 2013 of the "`TracReport`" are loaded into a new browser tab.

6.5.6   URL parameter

For managing URL bookmarks for the `myarmrtsbrowser` the following URL parameter are supported:

"`tp=<value>`"
  is used to pass an initial "Time period". The following values can be used:

  "`ty`" – "`This year`"

  "`12m`" – "`Last 12 months`"

  "`ly`" – "`Last year`"

  "`3y`" – "`Last 3 years`"

  "`5y`" – "`Last 5 years`"

  "`tm`" – "`This month`"

  "`lm`" – "`Last month`"

"`grp=<group-name>`"
  is used to pass an initial group name

6.6   RTS-Monitor

This section will show you how to use the `myarmrtsmonitor` interface in order to monitor current real time statistics of your applications. Point your browser to the URL identifying the MyARM RTS-Monitor web interface. The exact URL you have to use must include the host name of the server running the FastCGI process. If your web browser executes on the node where the web monitoring interface has been installed, you can use the URL http://localhost/fcgi-bin/myarmrtsmonitor.fcgi.

If you use the standalone version of the MyARM RTS-Monitor you can use the URL http://localhost:8083/.
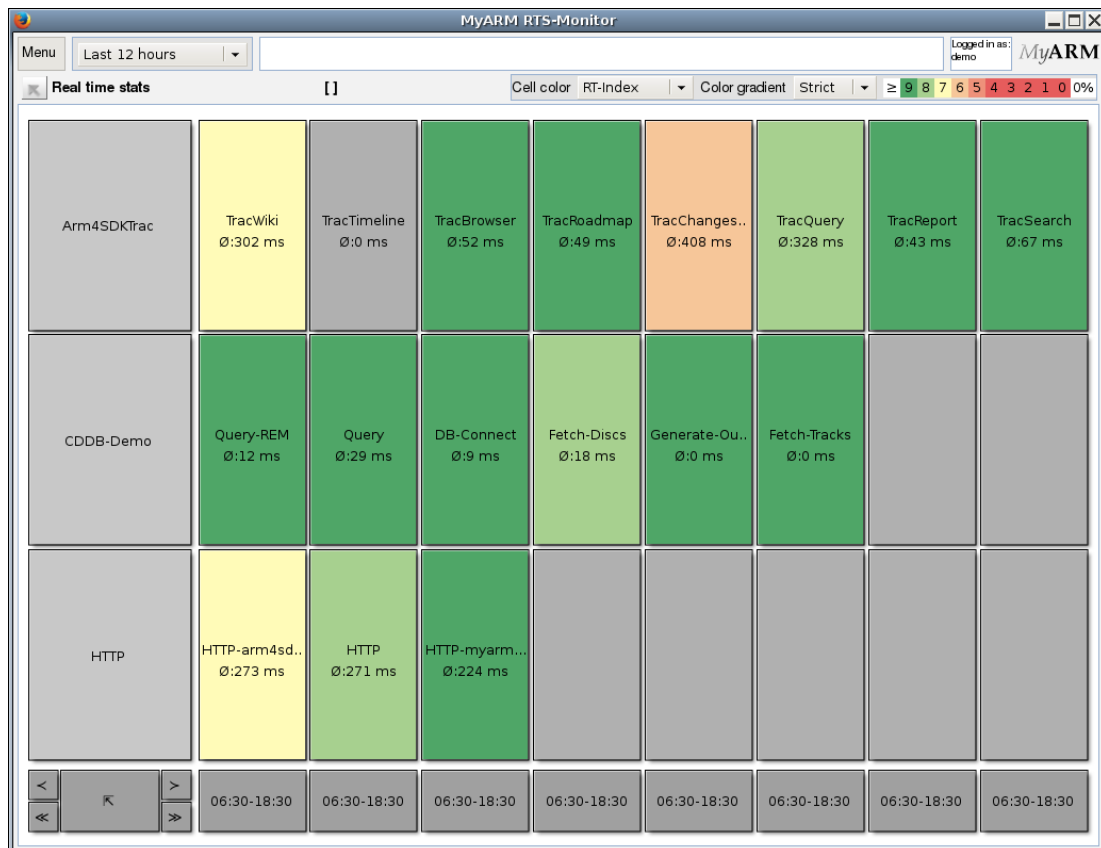
Figure 6.29: The MyARM RTS-Monitor

As you can see the major parts of the `myarmrtsmonitor` are equal to the `myarmrtsbrowser`. The only difference is that the `myarmrtsbrowser` is used to browse through the recorded RTS data and the `myarmrtsmonitor` is used to monitor current (todays) RTS data. Therefore the RTS-Monitor updates its current view every minute automatically. If response times are getting worse you will notice this by color changes of the appropriate transaction measurements.

6.6.1   Monitoring time period

The top level view of the `myarmrtsmonitor` is the starting point for monitoring current transaction statistics data. To define the starting time period the following time period drop-down-box is provided:
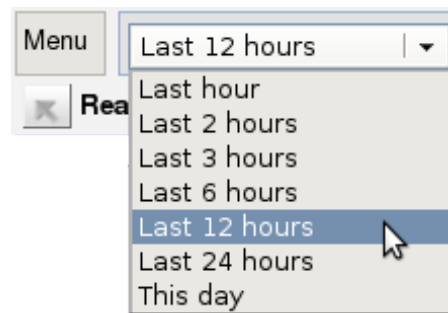
Figure 6.30: The time period drop-down-box

"Last hour"
 will aggregate all available data of the last hour. The next drill-down level will be a five minute view of the last hour.

"Last 2 hours"
 will aggregate all available data of the last 2 hours. The next drill-down level will be a hour view of the last 2 hours.

"Last 3 hours"
 will aggregate all available data of the last 3 hours. The next drill-down level will be a hour view of the last 3 hours.

"Last 6 hours"
 will aggregate all available data of the last 6 hours. The next drill-down level will be a hour view of the last 6 hours.

"Last 12 hours"
 will aggregate all available data of the last 12 hours. The next drill-down level will be a hour view of the last 12 hours.

"Last 24 hours"
 will aggregate all available data of the last 24 hours. The next drill-down level will a hour view of the last 24 hours.

"This day"
 will aggregate all available data of today. The next drill-down level will be a hour view of today.

All other controls are identical to the `myarmrtsbrowser` thus please read the appropriate sections there:

- "Cell coloring"

- "Time period navigation"

- "Dig into raw data"

6.6.2 URL parameter

For managing URL bookmarks for the `myarmrtsmonitor` the following URL parameter are supported:

"tp=<value>"
 is used to pass an initial "Time period". The following values can be used:

 "1" – "Last hour"

"2" – "Last 2 hours"

"3" – "Last 3 hours"

"6" – "Last 6 hours"

"12" – "Last 12 hours"

"24" – "Last 24 hours month"

"d" – "This day"

"grp=<group-name>"
   is used to pass an initial group name

6.7   Admin

This section will show you how to use the `myarmadmin` interface in order to manage so-called MyARM runtime configurations. Point your browser to the URL identifying the MyARM web administration interface. The exact URL you have to use must include the host name of the server running the FastCGI process. If your web browser executes on the node where the web browsing interface has been installed, you can use the URL http://localhost/fcgi-bin/myarmadmin.fcgi.

If you use the standalone version of the MyARM web browser you can use the URL http://localhost:8081/.
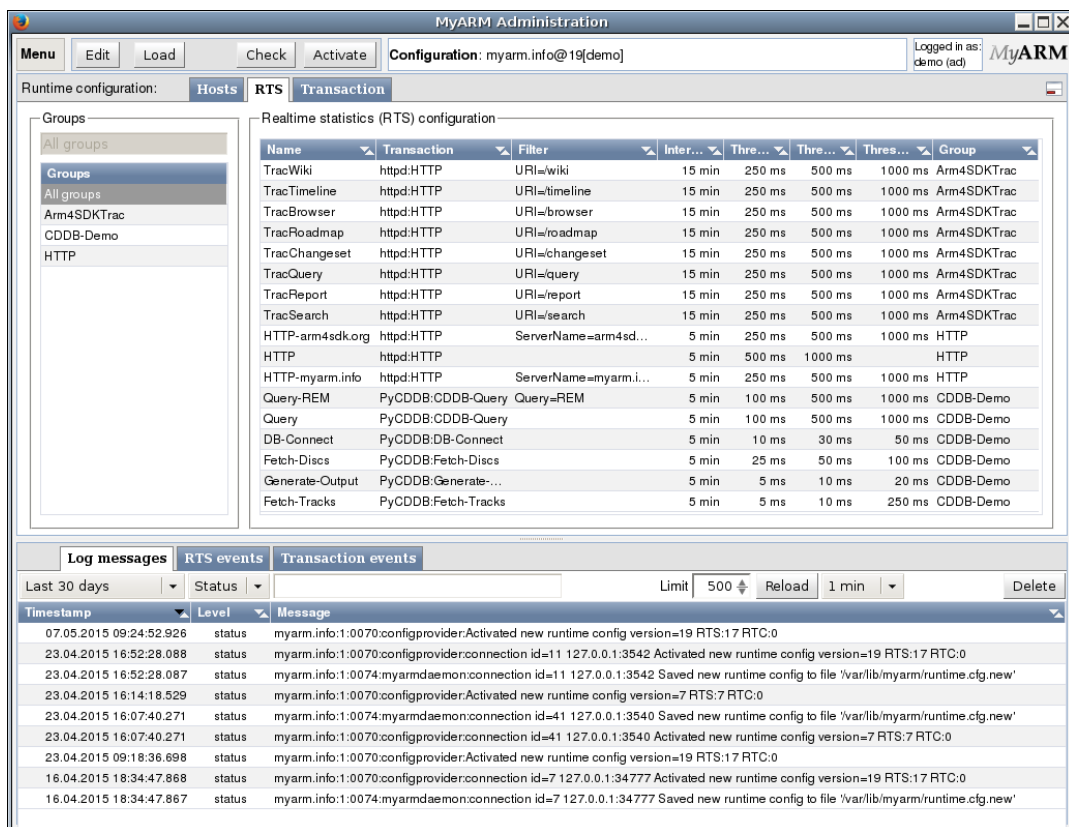
Figure 6.31: The MyARM web admin interface

The next section will give you a short overview regarding the major parts making up the `myarmadmin` interface. We will then follow up with the description of a typical start how to setup runtime configurations with the `myarmadmin` interface.

The following sections will provide detailed information on how to change or manage the runtime configurations within the `myarmadmin` interface.

6.7.1   Layout

This is how an (initially) empty MyARM web admin interface looks like:
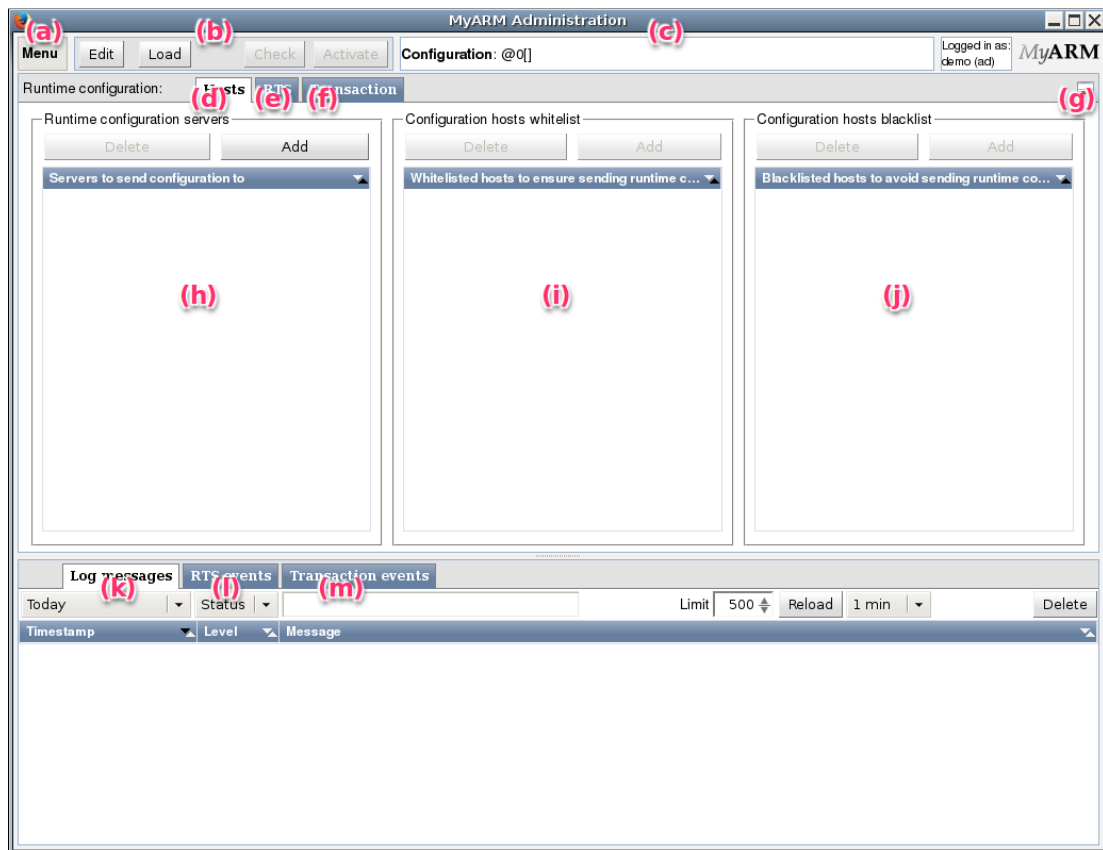
Figure 6.32: The main layout of the MyARM web admin

(a) The "`Menu`" is used to change settings that affect the whole application such as changing your user name and preferences for displaying data. Those preferences can be saved and loaded from the database.

(b) The "**Control area**" buttons are provided to edit, load or activate a runtime configuration set.

(c) The "`Configuration-Line`". Displays current loaded configuration name, version and user which created or changed the configuration.

(d) The runtime configuration "`Hosts`" tab. This tab is used to configure all hosts where to send runtime configurations to and to include or exclude some hosts using white- and blacklists.

(e) The runtime configuration "`RTS`" tab. This tab is used to manage all "`Real Time Statistics (RTS)`" configuration data

(see RTS concept)

(f) The runtime configuration "`Transaction`" tab. This tab is used to manage all "`Transaction runtime configurations (RTC)`" data

(see Transaction runtime configuration concept)

(g) The icon can be used to show and hide the "**Bottom area**"

(h) A list of hosts where `myarmdaemon` processes are running and accept runtime configurations

(i) A list of host patterns to include (whitelist) in runtime configuration distribution through the TCP datasink

(j) A list of host patterns to exclude (blacklist) from runtime configuration distribution through the TCP datasink

(k) The "`Log messages`" tab. This tab provides access to all log messages of the whole MyARM system written into the database

(l) The "`RTS events`" tab. This tab provides access to all RTS events triggered by configured RTS notification actions.

(m) The "`Transaction events`" tab. This tab provides access to all runtime transaction events triggered by configured transaction notification actions.

6.7.2 Manage runtime configurations

This section describes how to create and manage runtime configurations. The following section describes step by step how to create and activate a new runtime configuration and in a second part how to load and activate an older configuration.

1. By default the `myarmadmin` application provides a read-only view of runtime configuration data. Therefore to create or change runtime configurations we need to enter the edit mode by hitting the "`Edit`" button of the "**Control area**":
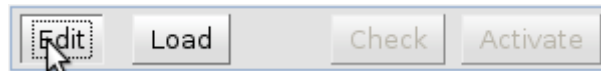
Figure 6.33: The "`Edit`" button within "**Control area**"

2. The "**Control area**" changes and the "`Cancel`" and a text field are displayed. Enter in the text field a name of the configuration. This name is to identify the configuration and is presented in the loading configuration dialog later.
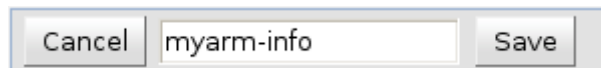
Figure 6.34: The "`Cancel`" button and configuration name text field within "**Control area**"

3. Add a host where a `myarmdaemon` is running and accepts runtime configurations:

| (a) Click "Add" button | (b) Enter a host name | (c) Host added |

Figure 6.35: Adding a new host for runtime configuration distribution

4. Real time statistics are always arranged in groups. Therefore we need to enter a group name to associate real time statistics as depicted in Figure 6.36a.

(a) Entering a RTS group name   (b) RTS group "Arm4SDKTrac" added
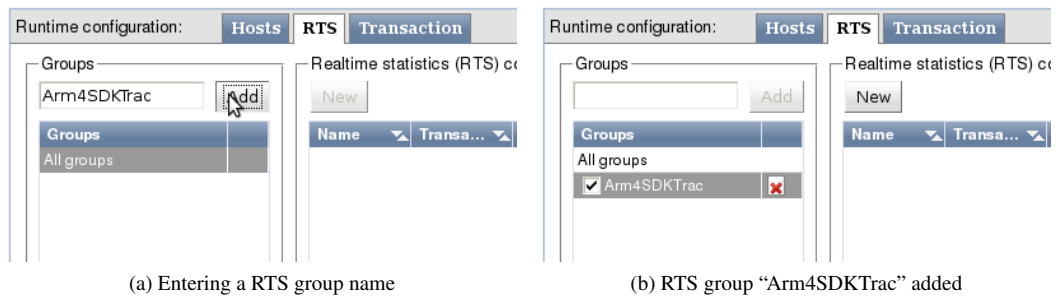
Figure 6.36: Adding a real time statistics group

In Figure 6.36b the group is added to the list of RTS groups and automatically selected. Now the "New" is enabled within the real time definition view. Clicking this button will open the following dialog to enter a new real time statistic definition.

5. The RTS definition dialog is used to create a new definition or to change an existing one. Here we create a new RTS definition. To do so we need first to select the application name for which an real time statistic should be generated:
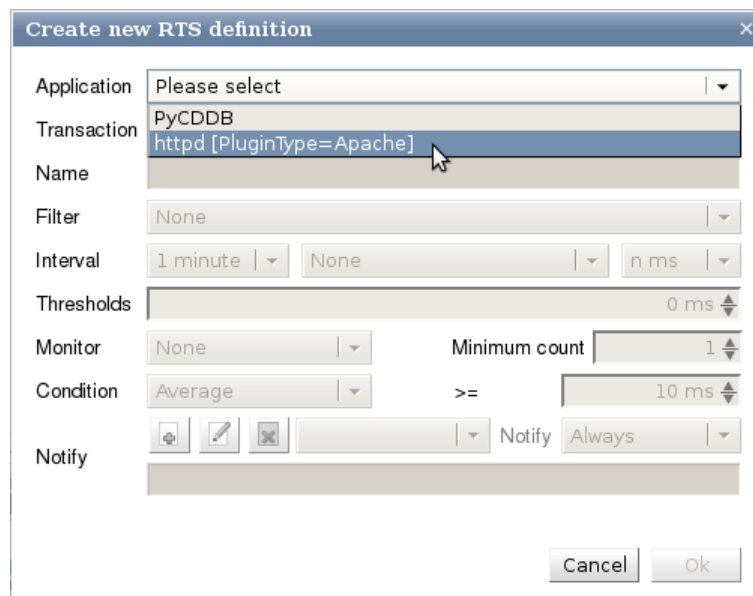
Figure 6.37: Selecting the application for a RTS definition

In our case we select the application "httpd" for the Apache-HTTPD-Server. In brackets the associated identity properties of the ARM application definition is shown.

6. Next we select the appropriate transaction definition name here "HTTP":

Figure 6.38: Selecting the transaction "`HTTP`" for a RTS definition

In brackets the associated identity properties and context property names of the ARM transaction definition is shown.

If the "`Drop`" check box is checked only RTS data is calculated and any single transaction measurement is dropped. If you have measurements with high frequency you can get the RTS statistic without storing any single measurement. This saves a lot of disk space.

7. After selecting the application and transaction definition names we need to enter a name for our new RTS definition:

Figure 6.39: The name and URI filter of the RTS definition

The real time statistic is named "TracWiki" and it generates statistics for "HTTP" measurements with an URI containing "/wiki" string. With these settings the real time statistic is calculated for each request to a Trac wiki page.

Note that now all needed data is entered the "Ok" button is enabled thus it is possible to apply our inputs.

8. However there are some more data we can define for the RTS definition:

Figure 6.40: RTS definition interval and thresholds

(a) the aggregation interval in minutes for the real time statistic to "15 minutes"

(b) the threshold unit from a drop-down-box selecting

   "n ms"
   milliseconds

   "n.nn ms"
   milliseconds with fraction

   "n s"
   seconds

   "n.nn s"
   seconds with fraction

(c) three thresholds (by selecting from a drop-down-box) to define response time constraints to estimate the response time in four categories:
   - below threshold 1 is a expected response time (good)
   - below threshold 2 is a tolerated response time (tolerated) (by a factor of 2 from the first threshold)
   - below threshold 3 is a bad response time (bad) (by a factor of 4 from the first threshold)
   - above threshold 3 is a frustrating response time (not wanted)

(d) Entering a monitor condition and a notify action is described in detail in section "Real time statistic configuration"

Given all this data we can click the "Ok" button and the RTS definition is created.

9. We created a new real time statistic definition called "`TracWiki`". The next step is to store the complete new runtime configuration into the database. This can be achieved by clicking on the "`Save`" button within the "**Control area**":
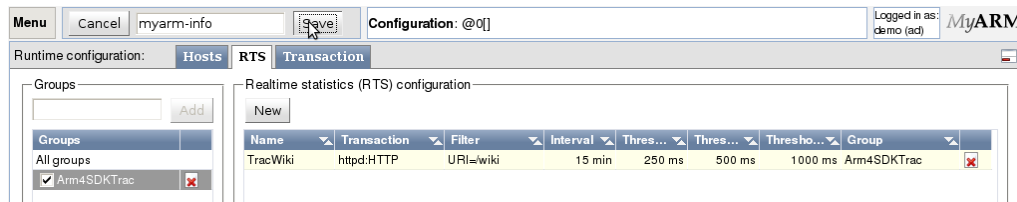
Figure 6.41: Save complete runtime configuration

10. After clicking the "`Save`" button the configuration is stored with a new version number into the database:
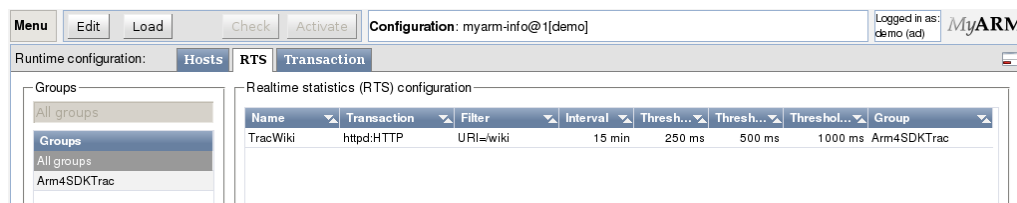
Figure 6.42: Newly created RTS definition stored in database

As depicted in Figure 6.42. The created RTS definition is stored in the database with the configuration name "`myarm-info`" at revision number "`1`" and created by user "`demo`".

11. To activate the newly created runtime configuration we need to inform any configured `myarmdaemon` process about the new runtime configuration. This can be done by clicking the "`Activate`" button in the "**Control area**". This will bring up an activation dialog which allows to select the configured hosts where `myarmdaemon` processes are running:
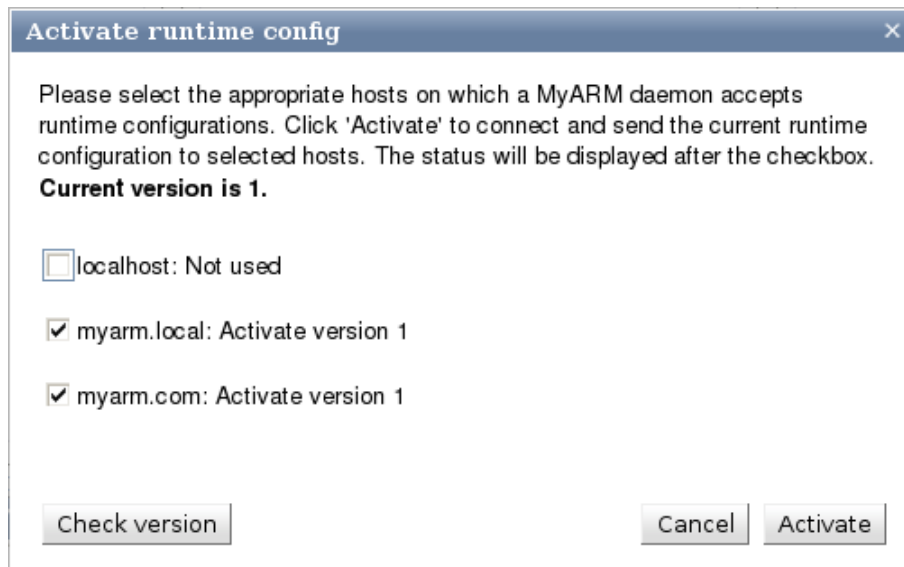
Figure 6.43: Runtime configuration activation dialog

As shown three hosts are configured to accept runtime configurations, here "`localhost`", "`myarm.local`" and "`myarm.com`". The check boxes before each host name can be used to exclude a host from the current activation. It is also possible to check the runtime configuration version at the appropriate hosts by just clicking the "`Check version`" button:
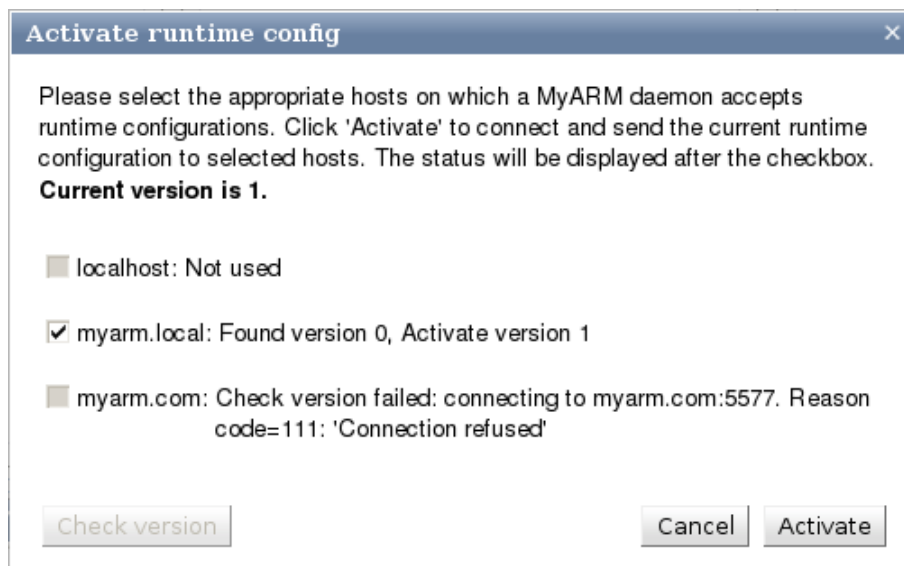
Figure 6.44:   Runtime configuration activation dialog versions checked at "`myarm.local`" and "`myarm.com`"

Since at "`myarm.com`" no `myarmdaemon` is running an error is returned. At "`myarm.local`" the `myarmdaemon` is running but has no configuration (version zero). Now to activate the runtime configuration at "`myarm.local`" just click the "`Activate`" button:
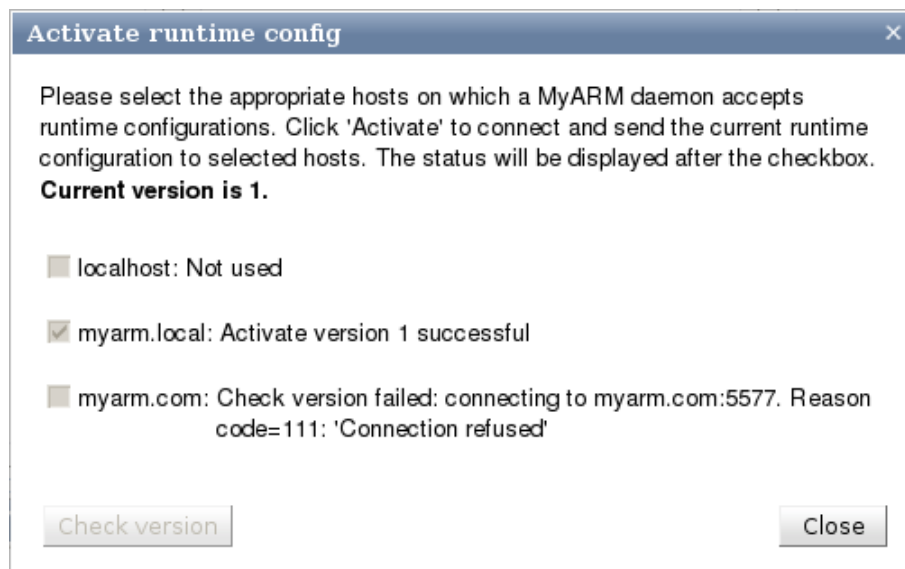
Figure 6.45: Runtime configuration activation dialog result

After clicking the "`Activate`" button and few moments later the activation results are displayed in the dialog:

(a) "`localhost`" was not used

(b) The `myarmdaemon` running on "`myarm.local`" successfully received and activated the new runtime configuration.

(c) "`myarm.com`" was not used since the "`Check version`" operation failed

6.7.3 The Menu
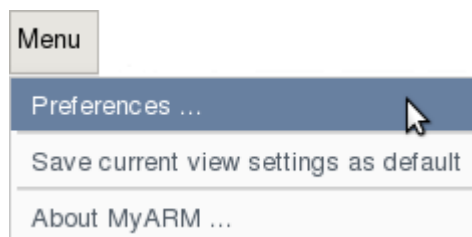
Figure 6.46: The menu

6.7.3.1 Preferences

The menu item labelled "`Menu`/`Preferences...`" opens a dialog for changing your preferences:
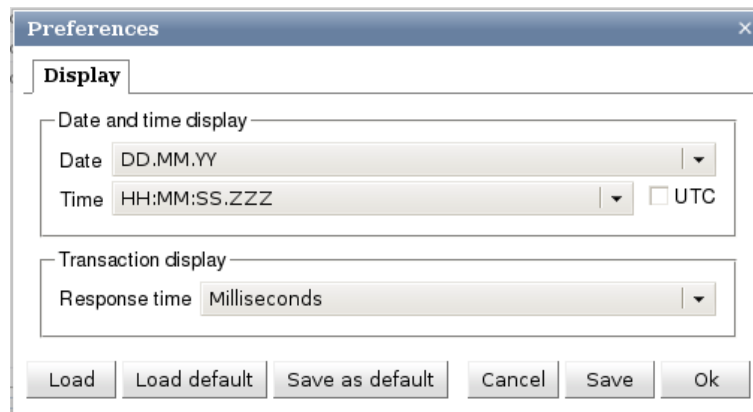
Figure 6.47: Preference dialog

The tab "`Display`" specifies the time and date formatting used when displaying dates and times. The default setting is to use European settings. By clicking the check box "`UTC`" you can force displaying the time in Universal Time Coordinated (UTC).

The drop-down-box labelled "`Response time`" determines which granularity is used for displaying response times. You can specify "`Microseconds`", "`Milliseconds`", "`Seconds`", "`Minutes`", "`Hours`" or "`Days`".

6.7.3.2 Save current view settings as default

Selecting this menu item all current settings are stored into database and if the application is started later on it will use these settings. Mainly the current viewed tabs and the settings of the Log messages, RTS events and Transaction events tabs are stored.

6.7.3.3 About

Selecting the menu item "`Menu / About MyARM ...`" will open a window displaying an informational text about the MyARM (e.g. version, edition, contact address, etc) (see the browser "About" menu).

6.7.4 Control buttons

The control buttons are used to edit, load, save or activate runtime configurations. In the read-only mode the following buttons are shown in the "**Control area**":
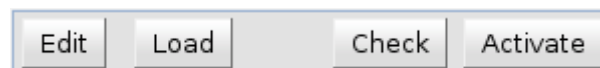
Figure 6.48: The control buttons

"**`Edit`**" button
> if clicked the `myarmadmin` session enters the edit mode

"**`Load`**" button
> if clicked a dialog is opened to select a runtime configuration from the database to load into the `myarmadmin` session

"Check" button
   if clicked a dialog is opened to check current runtime configuration within all configured `myarmdaemon` processes

"Activate" button
   if clicked a dialog is opened to activate current runtime configuration within all configured `myarmdaemon` processes
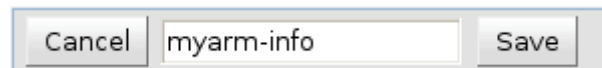
Within the edit mode the following buttons are shown:

Figure 6.49: The control button edit mode

"Cancel" button
   if clicked cancels the edit mode and returns to the read-only view

"Configuration name" text field
   enter here the configuration name to be used for storing the runtime configuration into the database

"Save" button
   if clicked it stores the current runtime configuration into the database and associates the configuration name entered into the "Configuration name" text field. This button is only enabled if some configuration data has changed, is new or old data was deleted

6.7.5 Real Time Statistic configuration

This section describes the "RTS Config" tab within the "**Configuration area**". The Figure 6.50 gives an overview of the real time statistic configuration tab:
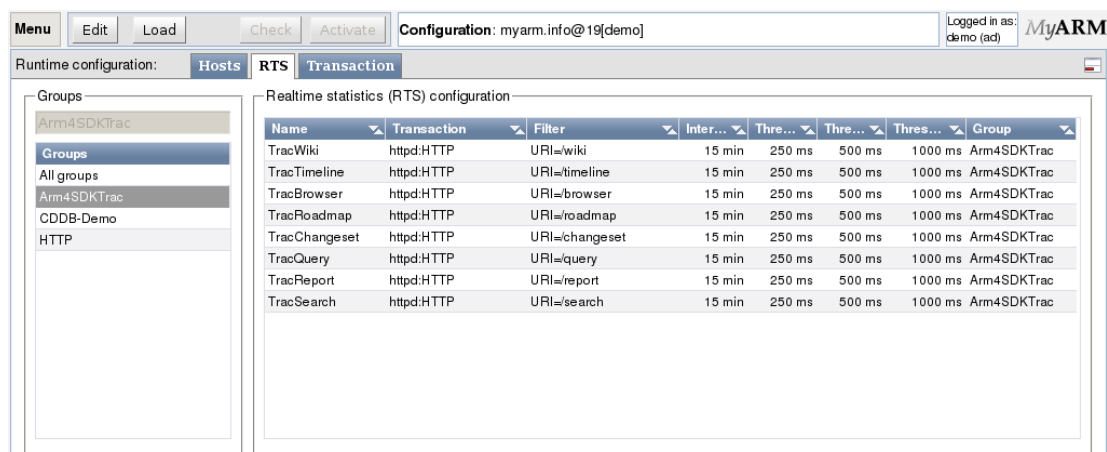
Figure 6.50: RTS configuration overview

Here the "Arm4SDKTrac" group is selected and only the "Trac*" definitions are shown. Entering the edit mode by clicking the "Edit" will present the following picture:
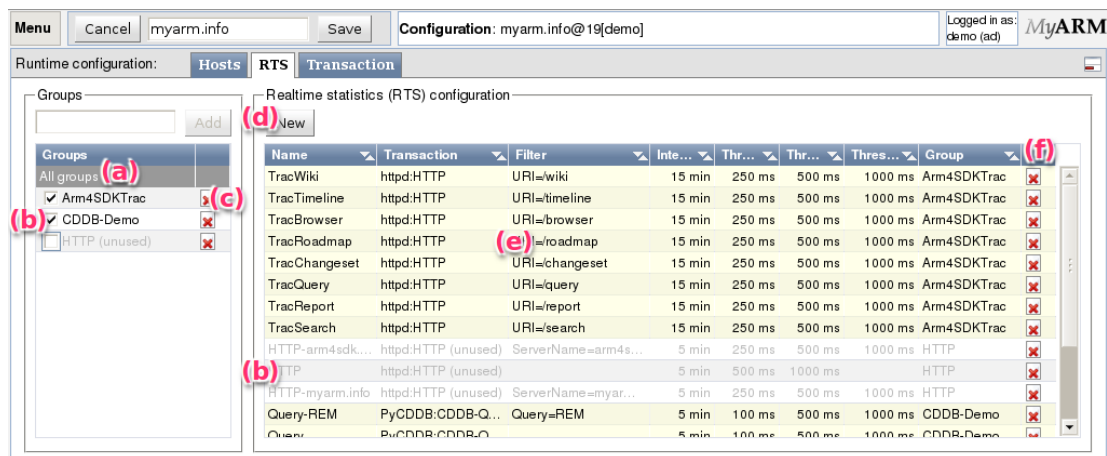
Figure 6.51: RTS configuration edit mode

(a) "`All groups`" entry is selected and all RTS definitions are shown

(b) The checkbox before each group name is used to set all RTS definitions of a group to be unused (not checked) or used (checked). Here the "`HTTP`" is unchecked and therefore unused. This is shown by a grayed out RTS definitions in the configuration view on the right.

(c) The "`Delete`" column presents a delete icon. When clicking this icon the appropriate RTS group name and possibly all associated RTS definitions will be deleted. A dialog will be opened to choose between deleting containing definitions or to associate the definitions to another group:
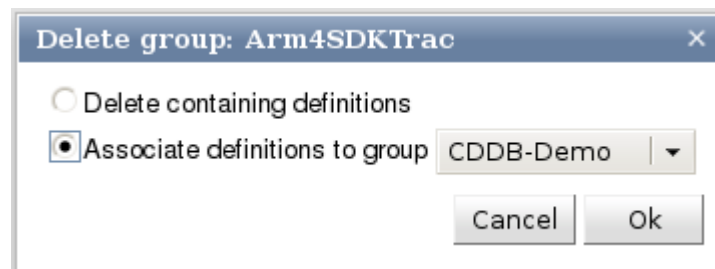
Figure 6.52: RTS configuration delete group dialog

(d) "`New`" button is used to open a dialog for creating a new RTS definition (see "create RTS definition").

(e) Each row rendered with light yellow background is editable and clicking on it a dialog is opened to edit the settings of the appropriate RTS definition (the background of this definition is then rendered in light blue):

Figure 6.53: RTS definition edit dialog

As shown in Figure 6.53 some of the input fields are disabled and this data can not be changed (e.g. transaction name and filter type). All other data can be changed and by hitting the "`Ok`" button the changes will be applied.

Controls up and including to the response time thresholds are already described in section Manage runtime configurations. Thus only the monitor controls are still missing:

The data produced for each RTS definition can be monitored regarding some conditions and if the condition occur a notify action can be triggered. The following conditions are supported:

Response time
> The average and deviation response times within the defined RTS interval can be monitored if they gets bigger then the configured value.

Index
> The response time and status index within the defined RTS interval can be monitored if they gets smaller then the configured percentage value.

Status
> The count of the "`Failed`", "`Not good`", "`Aborted`" or "`Unknown`" status transactions within the defined RTS interval can be monitored if they gets bigger than the configured value.

Note the "`Minimum count`" spin-box is used to define the minimum number of measurement to evaluate and possible trigger the notification action. This is used to avoid notifications for data with too few transaction measurements.

Note that the controls in below the monitor condition within the dialog can be used to select a notification script (here send rts email) is described in detail in the section "Managing notification actions".

(f) The "`Delete`" column is used to delete the appropriate RTS definition by clicking the delete icon. If an entry is deleted by mistake hitting the "`Cancel`" button from the "**Control area**" will reset the current changes.

6.7.6 Transaction configuration

This section describes the runtime configuration "`Transaction`" tab within the "**Configuration area**". The Figure 6.54 gives an overview of the transaction runtime configuration tab:
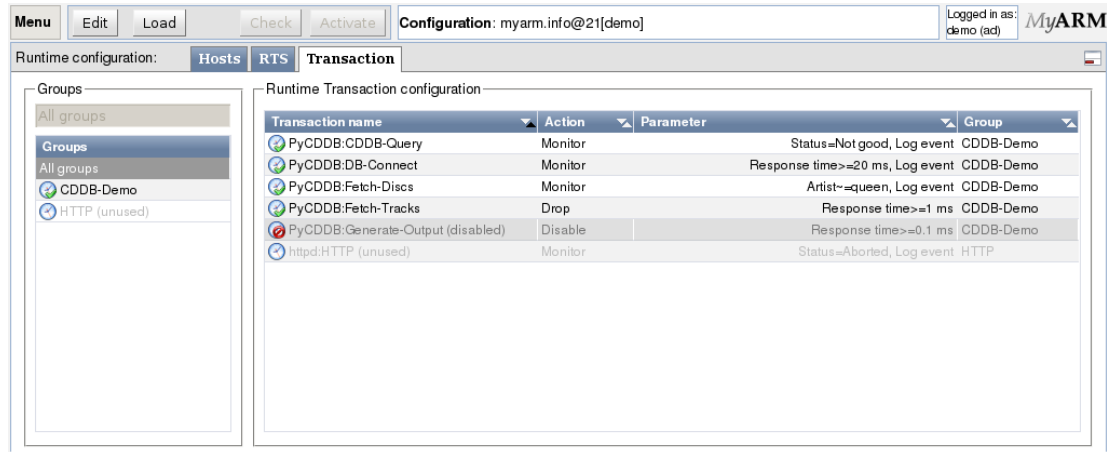

Figure 6.54: Transaction configuration overview

Here all groups and all transaction configuration entries are shown. Entering the edit mode by clicking the "`Edit`" will present the following picture:
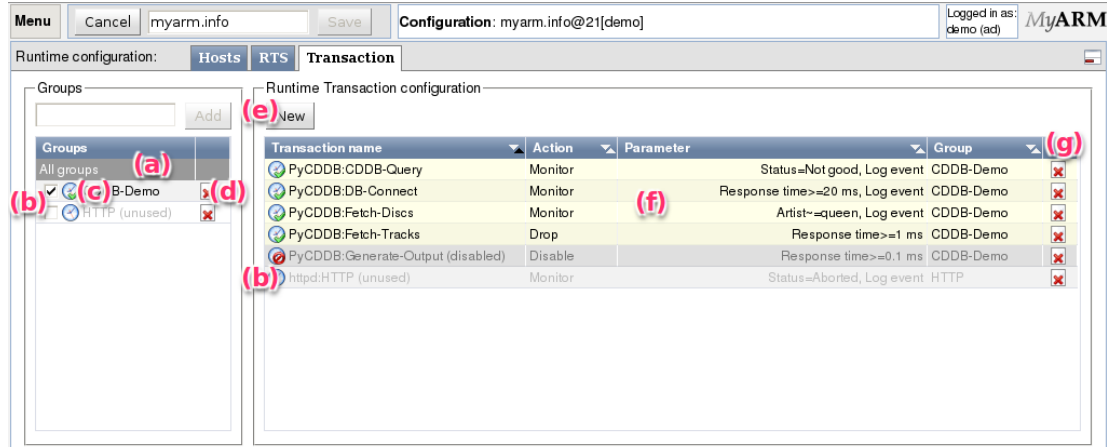

Figure 6.55: Transaction configuration edit mode

(a) "`All groups`" entry is selected and all transaction definitions are shown

(b) The checkbox before each group name is used to set all transaction definitions of a group to be unused (not checked) or used (checked). Here the "`HTTP`" is unchecked and therefore unused. This is shown by a grayed out transaction definition in the configuration view on the right.

(c) The clock icon before each group name is used to indicate if the transaction definitions of the group are disabled (), not used () or used (). Here the "`HTTP`" is unchecked and therefore unused and the is shown.

(d) The "Delete" column presents a delete icon. When clicking this icon the appropriate transaction group name and possibly all associated transaction definitions will be deleted. A dialog will be opened to choose between deleting containing definitions or to associate the definitions to another group.

(e) "New" button is used to open a dialog for creating a new transaction definition. The procedure for creating such a new definition is the same as creating a RTS definition. First the application and transaction is selected from a drop-down-box (see "create RTS definition") then following action types:

"Monitor"
Figure 6.56 depicts the input controls to enter a monitor for a transaction definition. The following attributes of a transaction can be monitored:

Status
Monitoring status "Failed", "Not good", "Aborted" or "Unknown"

Response time
Response time greater than a defined threshold

Context
A specified context property contains a defined string

URI
An URI contains a defined string

Figure 6.56: Transaction configuration monitor action

Below the action drop-down-box you can select a notification action which is executed if the monitoring condition is encountered. This is described in detail in section "Managing notification actions".

(f) Each row rendered with light yellow background is editable and clicking on it a dialog is opened to edit the settings of the appropriate transaction definition (the background of this definition is then rendered in light blue).

(g) The "Delete" column is used to delete the appropriate transaction definition by clicking the delete icon. If an entry is deleted by mistake hitting the "Cancel" button from the "**Control area**" will reset the current changes.

6.7.7 Managing notification actions

With release 4.0 MyARM introduces so-called notification actions. A notification action is a shell script which will be executed by the `myarmdaemon` if it encounter a runtime event created by a RTS or transaction monitoring condition. To manage notification actions the following controls are provided in the RTS definition and Transaction definition dialog:
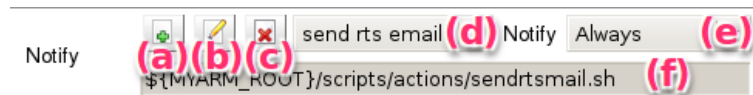

Figure 6.57: Notification action controls

(a) The "`Add`" ( ) is used to add a new notification action. A dialog is opened to enter a name (used in the drop-down-box (d)) and the script to execute (note you can uses environment variables if these are set where the `myarmdaemon` process is running).

(b) The "`Edit`" ( ) is used to edit the currently selected (shown in (d)) notification action (changing either name or notification script).

(c) The "`Delete`" ( ) is used to delete the currently selected (shown in (d)) notification action.

(d) The "`Actions`" is used to select the notification action from a drop-down-box list.

(e) The "`Notify`" drop-down-box is used to set the following three execution types:

Always
    execute script for every single runtime event

Once per hour
    execute script only once per hour even if there are more runtime events within one hour

Confirm
    execute script only once and wait for an user confirmation. If the user does not confirm the notification no further notification actions are executed.

(f) Displays the script which will be executed for a runtime event

6.7.7.1 Environment variables

With the execution of a shell script (or a batch script under Windows) the following environment variables can be used to get detailed information about the runtime event.

The following generic variables can be used within the script be referencing a appropriate environment variable (`${myarm_var}` on Linux and `%myarm_var%` on Windows):

myarm_appname
    name of the ARM application

myarm_details
    descriptive text describing the details of the notification

myarm_name
    shortcut for `${myarm_appname}:${myarm_tranname}`

myarm_tranname
    name of the ARM transaction

myarm_type
    type of the notification (RTS, Transaction, etc)

6.7.7.1.1 RTS specific environment variables

The following variables are specific to the RTS data triggered the runtime event:

myarm_abort
> number of measurements with "ABORT" status

myarm_average
> average response time within the RTS data interval

myarm_count
> number of measurements counted so far

myarm_deviation
> deviation of the response times

myarm_failed
> number of measurements with "FAILED" status

myarm_good
> number of measurements with "GOOD" status

myarm_max
> maximum response time within the RTS data interval

myarm_min
> minimum response time within the RTS data interval

myarm_notgood
> number of measurements with "NOT GOOD" status

myarm_notstopped
> number of measurements with "NOT STOPPED" status

myarm_rtidx
> response time index (percentage) derived from the RTS response time thresholds

myarm_rtsgroup
> name of the RTS group the RTS definition belongs to

myarm_rtsname
> name of the RTS definition

myarm_statusidx
> status index (percentage) derived from the different status counts

myarm_time
> time of RTS data which triggered the runtime event

myarm_unknown
> number of measurements with "UNKNOWN" status

6.7.7.1.2 Transaction specific environment variables

The following variables are specific to the transaction triggered the runtime event:

myarm_context
> the transaction context properties (if any)

myarm_rt
> the response time of the transaction

myarm_start
> the start time of the transaction

myarm_status
> the status of the transaction

myarm_system
> the system the transaction was executed on

myarm_user
> the user (if any) on behalf the transaction was executed

6.7.8   Log message area

This section describes the "**Log messages area**" which shows the log messages which are written to the MyARM database. Typically "`status`" and "`error`" messages are written into the MyARM database according to the configuration of the MyARM agent. The Figure 6.58 gives an overview of the supported filter controls and the log message table view:

Figure 6.58: Log messages

(a) A drop-down-box which selects the time interval for loading log messages from the database. Following time periods are supported: "`Today`", "`Last 24 hours`", "`Last 2 days`", "`Last 3 days`", "`Last 7 days`", "`Last 14 days`", "`Last 21 days`", "`Last 30 days`" and "`All time`"

(b) A drop-down-box which selects the log level for loading log messages from the database. The following levels are supported: "`Any`", "`Error`", "`Warning`", "`Status`", "`Config`", "`Info`" and "`Debug`".

(c) An filtering text field to filter loaded log messages by entering a sub-string of a log message. Here all log message containing the string "`config`" are displayed.

(d) "`Limit`" spin-box to limit the number of loaded log messages.

(e) "`Reload`" button is used to reload the log messages from the database.

(f) A drop-down-box specifying the time period for automatically reloading the log messages from the database Following time periods are supported: "`1 minute`", "`5 minutes`", "`15 minutes`", "`30 minutes`", "`1 hour`" and "`None`".

(g) "`Delete`" button to delete log messages older than the selected time interval (as selected by (a)) A dialog is opened which needs to be confirmed by the user prior deleting the events.

(h) Log messages table view

6.7.9 RTS events

This section describes the "**RTS runtime events area**" which shows RTS runtime events triggered by RTS monitor conditions. The Figure 6.59 gives an overview of the supported filter controls and the RTS event table view:

Figure 6.59: RTS runtime events

(a) A drop-down-box which selects the time interval for loading RTS runtime events from the database. Following time periods are supported: "`Today`", "`Last 24 hours`", "`Last 2 days`", "`Last 3 days`", "`Last 7 days`", "`Last 14 days`", "`Last 21 days`", "`Last 30 days`" and "`All time`"

(b) "`Conform`" button to confirm runtime events. This button is only enabled if a RTS event is selected and need confirmation (rendered with light red background).

(c) "`Limit`" spin-box to limit the number of loaded RTS events.

(d) "`Reload`" button is used to reload the RTS events from the database.

(e) A drop-down-box specifying the time period for automatically reloading the RTS events from the database Following time periods are supported: "`1 minute`", "`5 minutes`", "`15 minutes`", "`30 minutes`", "`1 hour`" and "`None`".

(f) "`Delete`" button to delete RTS events older than the selected time interval (as selected by (a)) A dialog is opened which needs to be confirmed by the user prior deleting the events.

(g) The details view provides details of the currently selected RTS event

(h) RTS event table view. The transaction column provides a link to the `myarmbrowser` to directly analyse the measurement which triggered the event.

6.7.10 Transaction events

This section describes the "**Transaction runtime events area**" which shows transaction runtime events triggered by Transaction monitor conditions. The Figure 6.60 gives an overview of the supported filter controls and the transaction event table view:

Figure 6.60: Transaction runtime events

(a) A drop-down-box which selects the time interval for loading transaction runtime events from the database. Following time periods are supported: "`Today`", "`Last 24 hours`", "`Last 2 days`", "`Last 3 days`", "`Last 7 days`", "`Last 14 days`", "`Last 21 days`", "`Last 30 days`" and "`All time`"

(b) "`Conform`" button to confirm runtime events. This button is only enabled if a transaction event is selected and need confirmation (rendered with light red background).

(c) "`Limit`" spin-box to limit the number of loaded transaction events.

(d) "`Reload`" button is used to reload the transaction events from the database.

(e) A drop-down-box specifying the time period for automatically reloading the transaction events from the database Following time periods are supported: "`1 minute`", "`5 minutes`", "`15 minutes`", "`30 minutes`", "`1 hour`" and "`None`".

(f) "`Delete`" button to delete transaction events older than the selected time interval (as selected by (a)) A dialog is opened which needs to be confirmed by the user prior deleting the events.

(g) The details view provides details of the currently selected transaction event

(h) Transaction event table view. The transaction column provides a link to the `myarmbrowser` to directly analyse the measurement which triggered the event. Rows rendered with light green background were confirmed by an user and rows rendered with light red background needs to be confirmed by the user.

`myarmadmin.wt` – starts the MyARM Administration web front end

> The script accepts command line parameters for overriding the default settings for the HTTP server address and port:
>
> Usage:
>
> ```
> myarmadmin.wt [options] [config]
> ```
>
> **`-p`** *port*, **`-port`** *port* Uses *port* as the HTTP server port. Default is `8081`.
>
> **`-a`** *addr*, **`-addr`** *addr* Uses *addr* as the HTTP server address. Default is `127.0.0.1`.
>
> *config* specifies a MyARM configuration file name to use (e.g. mysql.conf). Default is to use the current MyARM configuration file.

myarmrtsbrowser.wt – starts the MyARM RTS-Browser web front end

> The script accepts command line parameters for overriding the default settings for the HTTP server address and port:
>
> Usage:
>
> ```
> myarmrtsbrowser.wt [options] [config]
> ```
>
> **−p** *port*, **−port** *port* Uses *port* as the HTTP server port. Default is 8082.
>
> **−a** *addr*, **−addr** *addr* Uses *addr* as the HTTP server address. Default is 127.0.0.1.
>
> *config* specifies a MyARM configuration file name to use (e.g. mysql.conf). Default is to use the current MyARM configuration file.
>
> The maximum number of active session is limited to 2.

myarmrtsmonitor.wt – starts the MyARM RTS-Monitor web front end

> The script accepts command line parameters for overriding the default settings for the HTTP server address and port:
>
> Usage:
>
> ```
> myarmrtsmonitor.wt [options] [config]
> ```
>
> **−p** *port*, **−port** *port* Uses *port* as the HTTP server port. Default is 8083.
>
> **−a** *addr*, **−addr** *addr* Uses *addr* as the HTTP server address. Default is 127.0.0.1.
>
> *config* specifies a MyARM configuration file name to use (e.g. mysql.conf). Default is to use the current MyARM configuration file.
>
> The maximum number of active session is limited to 2.

6.7.11 FastCGI integration

The web front end can be integrated into an existing web server infrastructure which supports the FastCGI protocol. The following installation recipe documents the necessary steps to configure the Apache HTTPD server with the mod_fcgi module to deploy a MyARM web front end.

First of all, MyARM provides a shell script which should be used to spawn the FastCGI process. This script is located in the scripts directory of the MyARM installation:

myarmadmin.fcgi – starts the MyARM Administration web front end as a FastCGI process

myarmbrowser.fcgi – starts the MyARM Browser web front end as a FastCGI process

myarmrtsbrowser.fcgi – starts the MyARM RTS-Browser web front end as a FastCGI process

myarmrtsmonitor.fcgi – starts the MyARM RTS-Monitor web front end as a FastCGI process

The following steps are required to run the MyARM web front ends using FastCGI and the Apache HTTPD server:

1. Configure MyARM to use an appropriate database.

2. Enable mod_fcgi support within the Apache HTTPD server configuration.

3. Configure a fcgi directory as follows:

```
ScriptAlias /fcgi-bin/ /usr/lib/fcgi-bin/
<Directory "/usr/lib/fcgi-bin">
    SetHandler fcgid-script
    Options +ExecCGI
    Order allow,deny
    Allow from all
    # For apache 2.4 and above
    Require all granted
</Directory>
```

4. Copy the MyARM FastCGI shell scripts to that directory:

```
cp $MYARM_ROOT/scripts/myarmbrowser.fcgi /usr/lib/fcgi-bin/
```

5. Copy all style sheets and resources needed by the MyARM web front end to your document root:

```
cp -r $MYARM_ROOT/webapp/myarmweb /var/www/mydocroot
```

6. Create a directory to store session information and change the owner to the user running the Apache HTTPD server (apache, www-data, etc.):

```
mkdir /var/run/myarmweb
chown www-data.www-data /var/run/myarmweb
```

Note if you do not want to create such a directory under /var/run you need to change the MyARM web front end deployment configuration file found in $MYARM_ROOT/webapp/etc/myarmweb_fcgi_config.xml.

Once all these steps are done, restart the Apache HTTP server and point your browser to the configured site.

Chapter 7

Instrumentation

MyARM provides support for instrumentation of C/C++, C#, Java and Python applications. The following tools and frameworks are shipping with the MyARM distribution.

armtime – Command line tool which executes and measures another program given at command line arguments.

ARM 4.0 QArm4 framework – The QArm4 framework will easily integrate into existing Qt applications and will provide a simple measurement interface derived from the official ARM 4.0 Java bindings.

ARM 4.0 C++ framework – The ARM 4.0 C++ framework will easily integrate into any C++ application and will provide a simple measurement interface derived from the official ARM 4.0 Java bindings.

7.1 Correlator handover

ARM provides an unique and powerful correlation of transactions within distributed systems. The so-called correlator is used to correlate two transactions with a parent-child relationship. A correlator needs to be passed from one system to another for example from a web-client (browser) to a web-server (httpd). Therefore the correlator must be converted into an ASCII representation because these programs communicate with each other using an ASCII protocol named HTTP.

MyARM provides different string representations of a correlator. For compatibility reasons with the first mod_arm4 Apache HTTP Server module, a correlator string without any type information is treated as a hexadecimal encoded string.

To support more encodings MyARM introduces a string prefix followed by a colon to specify the type of encoding. Currently the following encoding types are supported:

b64 the correlator string following the colon is base 64 encoded.

hex the correlator string following the colon is hexadecimal encoded.

7.1.1 General rule

MyARM suggests to pass a correlator to other components within the application by extending the interfaces whenever its possible. If ARM is used it should be integral part of the application and therefore the ARM correlator is just another parameter which needs to be passed to used components.

7.1.2 Using HTTP

Passing an ARM correlator from a HTTP client to a HTTP server you only need to add the `ARM_CORRELATOR` header line to the HTTP header. For example:

```
GET /index.html HTTP/1.0\r\n
ARM_CORRELATOR: 00196400010000020167ab054d2901b71984e04ef3a53e068c\r\n
\r\n
```

7.1.3 Using an environment variable

If an correlator is passed using an environment variable between two components the environment variable called `ARM_CORRELATOR` is used. This technique can be used to pass a correlator between:

Two programs – The first program sets the `ARM_CORRELATOR` environment variable and starts a second program which reads the environment variable to use its contents as its parent correlator. See `armtime` program.

Program and a library – If a program uses a library which does not support ARM by its interface the environment variable technique can be used to pass a parent correlator to the ARM instrumentation within a library.

For example the following command will print out the MyARM generated correlator as a hexadecimal encoded string:

```
armtime echo "ARM_CORRELATOR=\$ARM_CORRELATOR"
```

will produce the following output:

```
ARM_CORRELATOR=00196400010000020121270979529ab7519b8fa97486216c
```

7.2 Tools

7.2.1 armtime – ARMed time command

The `armtime` tool is used for instrumenting and measuring programs and shell scripts without modifying the source code. Its called with the program and its arguments as arguments to the armtime command itself. It will register application and transaction meta data with the ARM library, start the ARM application and transaction instances using standard API calls and then call the given program. After the program terminates it stops both ARM transaction and application instances and will exit.

One most advantage of this approach is that it can be used for a huge variety of existing applications. Another advantage of using `armtime` is that it can make use of ARM correlator and properties:

Correlation – If a program which was started using `armtime` in turn starts another program using `armtime` these measurements are correlated using the standard ARM correlator mechanism resulting in a measurement chain of started programs.

Properties – Program parameters often influence the behaviour of program execution. To better understand the response time measurement of an `armtime` command the command line arguments can be associated to the measurement using standard ARM properties.

7.2.1.1 Command line options

Usage:

```
armtime [options] command [command arguments]
```

7.2.1.1.1 Options

–h, –help prints help page.

–a *name*, **–app** *name* specifies that *name* should be used as the ARM application name instead of *armtime*.

–f *code*, **–fail** *code* specifies that *code* or higher is the return code to mark a command be failed.

–g *string*, **–group** *string* provide application group used for starting the ARM application.

–i *string*, **–instance** *string* provide application instance used for starting the ARM application.

–l *libname*, **–lib-arm4** *libname* specifies *libname* as the ARM4 library to be loaded and used. Note *libname* can either be the base name of the library or the complete library name including suffix like `.so`.

–p, –prop-arg indicates that `command arguments` should be associated to the transaction measurement as ARM properties.

–P, –prop-args indicates that `command arguments` should be associated to the transaction as multiple context values.

–t *name*, **–tran** *name* specifies that *name* should be used as the ARM transaction name instead of `command name` passed.

–u *user*, **–user** *user* provide an *user* name to be associated with the ARM transaction.

Appendix A

Environment

The MyARM agent library can be configured through configuration files (See Appendix Configuration). MyARM needs to find the configuration files during start up. This is achieved by using environment variables.

The following environment variables can be set to specify the location of the configuration files and to set a valid license key. The easiest way to do this is to use the provided setup.sh script found in the scripts directory of the MyARM installation.

Please note that you have to source this script in order to set the environment variables in the current shell!

A.1 scripts/setup.sh

The setup.sh script is used to setup a MyARM environment in an easy way. The script tests your shell environment and modifies the following system environment variables to enable the full usage of MyARM:

PATH
   is modified to add the path to the MyARM programs so that any MyARM program can be executed.

LD_LIBRARY_PATH
   is modified to enable the runtime linker to find MyARM shared libraries (LD_LIBRARY_PATH on Linux).

MANPATH
   is modified to add the MyARM man page directory to the search path of the man system.

MYSQL_UNIX_PORT
   is set to the MySQL port as used by the current system. If this could not be determined, a warning message is issued.

To work properly, the following MyARM specific environment variables are set:

MYARM_ROOT
   root directory of the MyARM installation.

MYARM_VARRUN_DIR
   directory where to store any runtime data such as pid files or temporarily used files.

MYARM_VARLOG_DIR
   directory where to store any log files.

MYARM_VARLIB_DIR
> directory where to store any files which need to be persistent, such as SQLite databases or MyARM ARM data files.

MYARM_CONFIG_URL
> contains the configuration end point in URL notation. For more information read section MYARM_CONFIG_URL.

MYARM_LICENSE_KEY
> contains the license key for MyARM. This key is extracted from some configuration files. For more information see section "License key".

Usage:

```
$ . scripts/setup.sh [-q] [config]
```

The "`-q`" option suppresses any warning and the basic configuration output. The optional "`config`" parameter specifies a configuration to be used. This configuration must reside in the `$MYARM_ROOT/conf` or `$MYARM_ROOT/conf/templates` directory.

Typical usage of the `setup.sh` script, which configures MyARM in the current shell to use the myarmdaemon to receive data sent by the `tcp` datasink and store the ARM data into MySQL database:

```
user@localhost:/opt/myarm$ . scripts/setup.sh tcp_mysql.conf
```

A.2 MyARM environment variables

A.2.1 MYARM_CONFIG_URL

The `MYARM_CONFIG_URL` environment variable is used to specify the MyARM configuration file. It uses the common URL notation of the form `<protocol>://<host>:<port>/<filepath>`. Currently, the following `<protocol>` types are supported:

file
> specifies a local configuration file.

For example, if you want to specify a global `<filepath>` you have to use two slashes after the protocol indicator, resulting in the following setup:

```
MYARM_CONFIG_URL=file:////opt/myarm/conf/foo.conf
export MYARM_CONFIG_URL

$ myarmquery FooTran FooApp
```

A.2.2 MYARM_LICENSE_KEY

If the `MYARM_LICENSE_KEY` environment variable is set and is not empty, it contains the license key to deploy the MyARM components on this host.

A.2.3 MYARM_VARRUN_DIR

This variable is used within the configuration files to reference the directory where to store runtime data like pid files or temporary data.

A.2.4   MYARM_VARLOG_DIR

This variable is used within the configuration files to reference the directory where to log messages.

A.2.5   MYARM_VARLIB_DIR

This variable is used within the configuration files to reference the directory where to store persistent data such as a SQLite database file or ARM data files.

A.2.6   MYARM_TRACE

With the `MYARM_TRACE` environment variable it is possible to enable diagnostic trace messages during configuration and initialization of the MyARM agent to the standard error channel. Currently, the following options are supported, which can be combined using a comma character as the delimiter:

none
>   no tracing at all.

all
>   enables all the following trace options.

init
>   enables tracing of initialization and cleanup of the MyARM agent.

env
>   enables tracing output of all used environment variables.

config
>   enables tracing output of all configuration properties.

readcfg
>   enables tracing of reading configuration files.

For example:

```
export MYARM_TRACE=init,env
export MYARM_TRACE=all
```

Appendix B

License key

MyARM checks for a valid license using the so-called *license key*. The MyARM product family is available in several editions. A license key is only valid for a given edition and release. You get your license key for MyARM only by email after finishing the ordering process and payment of the license fee.

The license key can be made available to the MyARM components in the following ways:

1. Using the MYARM_LICENSE_KEY environment variable which contains the license key provided to run MyARM components on the current host.

2. Adding the following line to the user.conf configuration file:

   ```
   myarm.key = <license_key_string>
   ```

Appendix C

Configuration

C.1 Introduction

The configuration of the MyARM agent library is done mainly through configuration files which contain so-called named value pairs or just called properties. Each property is specified in one or more lines of the file in the following form:

```
name = value
```

where at least the name and the sign character have to be in the first line. Line concatenation have to be introduced by a backslash character at the end of a line. The name can be structured hierarchically by separating name parts with dots:

```
agent.sink.name = db_mysql
```

Configuration C.1: **Assignment**

This example assigns the value of *db_mysql* to the property name `agent.sink.name` which consists of the three parts `agent`, `sink` and `name`. One use of these hierarchical names is to unify the configuration of one component within the system by it's first name part. In the above example the first name part is `agent` which means its a property of the agent library component of the MyARM system. The second name part `sink` specifies the sub-component and the third part `name` specifies the property of the sub-component. This configuration line configures the datasink name *db_mysql* to be used for the datasink of the MyARM agent.

A value can also refer to an environment variable which has to be enclosed by braces and prefixed by a dollar sign: `${VAR}`. This gives you powerful possibilities in writing configuration files. For example you can write user independent configuration files.

```
agent.log.file = ${TMPDIR}/agent.log
```

Configuration C.2: **User-specific**

To make configuration tasks easier, MyARM has the ability to `include` other configuration files. Just add a line

```
include filename
```

Configuration C.3: **Includes**

in a configuration file and the file with the name *filename* within the configuration directory is included. The template configuration files makes use of the include mechanism.

C.1.1   Configuration files

There are three possibilities to specify a configuration file for a MyARM tool or an ARM instrumented application using MyARM. The first possibility overrides configuration settings in the following ones, etc.

1. specifying a configuration file through command line options: `-cf` *conf_file* or `--conf-file` *conf_file*. This is only available for MyARM tools. The ARM standard defines no mechansim for accessing command line options. Therefore, it is not possible to configure an ARM instrumented application using this technique.

2. specifying a configuration source through the $MYARM_CONFIG_URL environment variable. See section "Environment".

3. providing the `.myarmrc` file in the $HOME directory of the current user.

C.1.1.1   User configuration file

MyARM provides a mechanism for user-specific configuration of MyARM. If the file `user.conf` exists in the current configuration directory of MyARM, it is loaded after all other configuration files. Each configuration property in this file will overwrite a previously loaded configuration property. Therefore, the user can place any configuration property in the file which should be different from the MyARM standard configuration.

C.1.1.2   Provided configuration files

By default MyARM provides ready-to-run configuration files for the all components. Thus all you need to do is to select your favourite configuration file from the `templates` directory and to change the values to your needs by overriding these values within the `user.conf` file.

C.1.1.2.1   Template configuration files

These template configuration files are the anchor to the MyARM configuration system and should be used with the `setup.sh` script or used with the MYARM_CONFIG_URL environment variable. They are located in the `$MYARM_ROOT/conf/templates` directory.

Under Unix-like systems a symbolic link from `$MYARM_ROOT/conf/<conffile>` to the `$MYARM_ROOT/conf/templates/<conffile>` configuration file is created automatically with the `setup.sh` script.

file_mysql.conf

Main configuration file for using the `myarmdaemon` process to collect data from the filestorage directory of instrumented applications and to store the ARM data within a MySQL database.

file_sqlite3.conf

Main configuration file for using the `myarmdaemon` process to collect ARM data from the filestorage directory of instrumented applications and to store the ARM data within a SQLite3 database.

mysql.conf

Main configuration file for using MyARM with a MySQL database without a `myarmdaemon`. Note this configuration will use the database directly from your instrumented application.

sqlite3.conf

Main configuration file for using MyARM with a SQLite3 database without a `myarmdaemon`. Note this configuration will use the database directly from your instrumented application.

tcp.conf

Main configuration file used by instrumented applications to forward collected ARM data to the `myarmdaemon` using the `tcp` datasink.

tcp_archive.conf

Main configuration file used by an instrumented application which uses a `tcp` datasink and the appropriate `myarmdaemon` stores the data into an `archive` datasink.

tcp_daemon_archive.conf

Main configuration file used by the `myarmdaemon` process to collect ARM data from instrumented applications by using the `tcp` datasink to store collected ARM data into MyARM binary files.

tcp_mysql.conf

Main configuration file for using the `myarmdaemon` process to collect ARM data from instrumented applications and to store the ARM data within a MySQL database.

tcp_sqlite3.conf

Main configuration file for using the `myarmdaemon` process to collect ARM data from instrumented applications by using the `tcp` datasink and to store the ARM data within a SQLite3 database.

C.1.1.2.2   Main configuration file

The main configuration file includes all supported configuration properties with their default values. Thus this is the place to look for a configuration property and to copy it to the `user.conf` file and changing its value. That's all.

C.1.2   Configuring components

Currently, the following components can be configured using properties where the base prefix is specified in parentheses:

- Basic (`basic`). See appendix "Configuring basic properties".

- Logging (`log`). See appendix "Configuring log facility".

- Agent (`agent`). See appendix "Configuring agent".

- Resource watchdog (`resource.watchdog`. See appendix "Configuring resource watchdog facility".

- Sink and database (`free`). See appendix "Configuring datasink component".

  `free` in this context means you can choose a name as you prefer (e.g. `db_mysql` for a MySQL database configuration).

- myarmdaemon (`daemon`). See section "myarmdaemon".

- Tools (`tools`). See appendix "Configuring command line tools".

  – myarmquery (`tools.query`). See section "myarmquery".

C.2 Configuring basic attributes

Some properties are subject to all MyARM components and therefore can be configured in the basic section of the MyARM configuration. The following basic properties are available:

basic.usage

> The basic usage property is used to specify the general usage of the MyARM agent. Depending on the following usage value, the appropriate configuration properties set meaningful values (see configuration properties below):
>
> **min**
>> Minimum agent usage can be used for few (1-100) measurements per second. Uses the minimum value of the properties listed below.
>
> **normal**
>> Normal agent usage can be used for about 100 up to 1000 measurements per second. Uses the default value of the properties listed below.
>
> **high**
>> High agent usage can be used for about 1000 to 5000 measurements per second. Uses half of the maximum value of the properties listed below.
>
> **max**
>> Maximum agent usage can be used for about 5000 to 100000 measurements per second. Uses the the maximum value of the properties listed below.
>>
>> Note that the agent can handle such high numbers of measurements per second, but the underlying datasink needs to handle also such a high measurement volume. So this configuration is able to handle very high measurement peaks!
>
> **user**
>> The user is responsible for setting all configuration properties listed below.
>
> Configuration properties directly set by the `basic.usage` property (if not using `user`):
>
> - `basic.armdata.buffer.size`
> - `basic.armdata.buffer.pool.size`
> - `basic.armdata.buffer.pool.max`
> - `basic.filestorage.writer.rolling.size`
> - `agent.transaction.pool.size`
> - `agent.transaction.pool.max`
> - `agent.metric.pool.size`
> - `agent.metric.pool.max`
> - `db_mysql.connections`

basic.time.use_utc

> this boolean property is used to switch display of times between UTC (`true`) and local time (`false`).
>
> Default is *false*.

basic.time.format *(New since 4.0.x.0)*

> specifies the format pattern to use for displaying time values. See "Configuring time formats" section for details.
>
> Default is *HH:MM:SS.ZZZ*.

basic.time.date.format

> specifies the format pattern to use for displaying date values. See "Configuring date formats" section for details.
>
> Default is *DD.MM.YY*.

C.2.1   Archive configuration properties

basic.archive.enable

> Boolean which indicates if MyARM uses the archive concept to store measured ARM data (true) or not (false).
>
> Default is *false*.
>
> This property is normally set by a template configuration file.   For example the `tcp_archive.conf` template file contains the following lines:

```
# set sink for transporting ARM data to the myarmdaemon
agent.sink.name = sink_tcp
# only ARM instrumented applications are allowed in this config
# thus disable daemon
daemon.disabled = true
# no database in the config thus disable tools
tools.disabled = true
# global property to allow any part of MyARM to check if archive
# is enabled or not
basic.archive.enable = true
# MyARM - include main (all) configuration sections            #
include main.conf
```

Configuration C.4: **TCP Archive template example**

basic.archive.directory

> Final root archive directory for the storage of MyARM data files. This directory is used by the myarmarchive tool and the myarmdaemon archive component to read data from.
>
> Default is *${MYARM_VARLIB_DIR}/archive/final/*.

basic.archive.transaction.filepattern

> File pattern (including directories) for selecting the appropriate file for transaction measurements.
>
> Default is *transactions/$m/$s/%Y%m%d_%H%M%S_tran*.
>
> The following format options are supported:
>
> **$m** – is replaced by a string defining two directories with timestamps in seconds since epoch (1.1.1970). The first directory represents the seconds for each day at midnight (beginning of the day) and the second directory represents the beginning of each minute within the day of the first directory. For example the timestamp 4th december 2017, 16:12:00 will create the folling two directories: `1512345600/1512403920`.
>
> **$s** – is replaced by the appropriate system address
>
> **$a** – is replaced by the appropriate application name
>
> **%Y** – is replaced by the year of the start timestamp of the transaction
>
> **%m** – is replaced by the month of the start timestamp of the transaction
>
> **%d** – is replaced by the day of the start timestamp of the transaction
>
> **%H** – is replaced by the hour of the start timestamp of the transaction
>
> **%M** – is replaced by the minute of the start timestamp of the transaction
>
> **%S** – is replaced by the second of the start timestamp of the transaction

${ENV} – is replaced by the contents of the *ENV* environment variable

`basic.archive.definition.filepattern`

File pattern (including directories) for selecting the appropriate file for ARM definition data.

Default is *definitions/$s_defs*.

The following format options are supported:

$s – is replaced by the appropriate system address

$a – is replaced by the appropriate application name

${ENV} – is replaced by the contents of the *ENV* environment variable

`basic.archive.threshold.enable`

Boolean which indicates if MyARM should monitor transactions which exceed configured thresholds (true) or not (false).

Default is *false*.

If this option is turned on, thresholds for certain transaction definitions can be activated within the `myarmdaemon` by using the appropriate `myarmdaemon` **command**.

`basic.archive.threshold.filepattern`

File pattern (including directories) for selecting the appropriate file for threshold data.

Default is *thresholds/thresholds_%Y%m%d%H/%M*.

The following format options are supported:

%Y – is replaced by the year of the start timestamp of the transaction

%m – is replaced by the month of the start timestamp of the transaction

%d – is replaced by the day of the start timestamp of the transaction

%H – is replaced by the hour of the start timestamp of the transaction

%M – is replaced by the minute of the start timestamp of the transaction

%S – is replaced by the second of the start timestamp of the transaction

${ENV} – is replaced by the contents of the *ENV* environment variable

`basic.archive.database.url`

Specifies the database connection parameters using the URL syntax (including host, user and optional password). See Database URL notation for details.

Default is *mysql://myarm@localhost:3306/?connections=4*.

`basic.archive.database.export.basename`

Specifies the base name of database instances used by the `myarmdaemon` archive component to export matching transactions from the archive into the configured database (`basic.archive.database.url`). Each export will use basename as the prefix plus a given name to create a new database instance.

Default is *MyARMIncidients_*.

`basic.archive.database.threshold.name`

Specifies the database instance name to be used for storing threshold data within the database. This property is only used if `basic.archive.threshold.enable` is set to true.

Default is *MyARMThresholds*.

`basic.archive.late_data`

Defines the period in the past to define data as late. Any data which arrives later in the archive will not be forwarded to the Apache Spark connector.

Default is *15m* (15 minutes), minimum is *1m* (1 minute), maximum is *1h* (1 hour).

basic.archive.cleanup.enable
> If set to true old archive data will be deleted periodically.
>
> Default is *true*.

basic.archive.cleanup.interval
> Specifies the period to periodically delete old archive files.
>
> Default is *1h* (1 hour), minimum is *1m* (1 minute), maximum is *24h* (24 hours).

basic.archive.cleanup.keep_data
> Specifies the period to keep archive data within the archive. Files older than current time minus this period will be deleted.
>
> Default is *28d* (28 days), minimum is *1d* (1 day), maximum is *365d* (365 days).

basic.archive.user.context.script
> Specifies a shell script which is used by the myarmbrowser to provide context specific archive queries.
>
> Default is "" (empty string).
>
> The script needs to support at least the `-list-attributes` option to return a list of available options. Each supported option is specified by by a comma separated tuple of `Command-option`, `Name`, a `Regular-Expression` matching a valid input of the attribute and a `Description`. Each entry can be qwouted with double-quotes to include a comma as well. Here is an example:

```
# command-option, name, reg-exp, description
--user=,User,^.+ .+$,"Enter username (forename, surname or fore- and surname)"
--email=,Email address,^.+@.+\..+$,Enter a valid email address
```

C.2.2 ARM data buffer configuration properties

basic.armdata.buffer.size
> various MyARM components are collecting ARM data into a memory buffer before further processing. The buffer size for this purpose can be configured using this property. By default this buffer is *16384* bytes of size which is also the minimum value for this property. The maximum size if *512KB*.

basic.armdata.buffer.pool.size
> number of ARM data buffers to pre-allocate during initialization of MyARM.
>
> Default is *32*, Minimum is *8*, Maximum is defined by the `basic.armdata.buffer.pool.max`

basic.armdata.buffer.pool.max
> maximum number of ARM data buffers MyARM will allocate. If this limit is reached, any data which should be written to an ARM data buffer is dropped and appropriate dropped counters are increased.
>
> Default is *512*, Minimum is *256*, Maximum is *1024*.

C.2.3 File storage configuration properties

The following configuration properties belongs to the file storage reader part:

basic.filestorage.reader.directory
> defines the directory to scan for new ARM data files.
>
> Default directory is *${MYARM_VARLIB_DIR}/myarmfiles*.

`basic.filestorage.reader.directory.scan.period`
> defines the number of seconds to scan the directory for new files.
>
> Default is *5s* (5 seconds), minimum is *1s* (1 second), maximum is *5m* (5 minutes).

`basic.filestorage.reader.directory.scan.keep_corrupt`
> boolean which defines if corrupted ARM data files should be kept or not. If kept the file is renamed by adding the suffix `.corrupt`.
>
> Default is *true*.

The following configuration properties belongs to the file storage writer part:

`basic.filestorage.writer.workfile`
> specifies the complete filename (including directory names) for the work file. An unique suffix will be appended for each running process. The ARM data is written to this file and when its closed it is moved to the configured file storage reader directory (`basic.filestorage.reader.directory`).
>
> Default is *${MYARM_VARLIB_DIR}/myarmfile.data*.

`basic.filestorage.writer.rolling.seconds`
> specifies the number of seconds after which a new work file will be used. The old work file will be moved to the configured directory.
>
> Default is *60s* seconds.

`basic.filestorage.writer.rolling.size`
> specifies the maximal size in bytes of the work file. If the work file gets bigger a new work file is opened and the old work file is moved to the configured directory.
>
> Default is *128KiB* (128 kilobytes).

`basic.filestorage.writer.diskusage.max_used`
> specifies the maximal size in bytes of all ARM data files in the file storage reader directory (`basic.filestorage.reader.directory`). If this limit is reached any new ARM data files will be dropped and an error will be reported.
>
> Default is *100MiB* (100 megabytes).

`basic.filestorage.writer.diskusage.min_free`
> specifies the minimal free size in bytes of the file system the file storage reader directory (`basic.filestorage.reader.directory`) resides. If this limit is reached any new ARM data files will be dropped and an error will be reported.
>
> Default is *200MiB* (200 megabytes).

C.2.4 Runtime configuration properties

`basic.runtime.config` *(New since 4.0.x.0)*
> defines the local absolute filename to be used for storing and reading the runtime configuration.
>
> Default is *${MYARM_VARLIB_DIR}/runtime.cfg*.

`basic.runtime.config.period` *(New since 4.0.x.0)*
> defines the number of seconds to check for a new runtime configuration stored in the `basic.runtime.config` file.
>
> Default is *5m* (5 minutes), minimum is *1m* (1 minute), maximum is *1h* (1 hour).

basic.runtime.config.client.host *(New since 4.0.x.0)*
>   defines the host from which new runtime configuration connections are accepted. This is the host or
>   IP address where the myarmadmin web application server part runs on.
>
>   Default is *localhost*.

basic.runtime.config.port *(New since 4.0.x.0)*
>   defines the port to listen for new runtime configuration connections.
>
>   Default is *5577*.

C.3   Configuring the ARM agent

The behaviour of MyARM can be changed by the following agent properties. Note: not all properties are
used for the different language binding agents. See the binding information for each property.

agent.correlator.add_user
>   can be set to `yes` or `no` and, if enabled the ARM agent adds a passed user identity to the generated
>   correlator. This user identity within the correlator is afterwards used by sub-transactions and starts
>   the sub-transactions on behalf of this user. **Note:** This will override the user identity passed with the
>   standard ARM 4.0 API.
>
>   **Bindings**: arm40c, arm40java, arm40c#.

agent.disabled
>   with this boolean property the complete MyARM agent can be disabled.
>
>   Default is *false*.
>
>   **Bindings**: arm40c, arm40java, arm40c#.

agent.flush.period
>   defines the number of seconds after which all cached ARM data are written to the configured back-
>   end.
>
>   The default is *10s* (10 seconds), minimum is *1s* (1 second), maximum is *5m* (5 minutes).
>
>   **Bindings**: arm40c, arm40java, arm40c#.

agent.api.log.error
>   if this boolean property is set to `true`, any invalid parameters passed through the ARM API are
>   logged.
>
>   Default is *false*.
>
>   **Bindings**: arm40c, arm40java, arm40c#.

agent.api.log.warning
>   if this boolean property is set to `true`, any warnings regarding ARM 4.0 standard compliance are
>   logged.
>
>   Default is *false*.
>
>   **Bindings**: arm40c, arm40java, arm40c#.

agent.sink.name
>   defines the name of the datasink to use. See "MySQL configuration example".
>
>   **Bindings**: arm40c, arm40java, arm40c#.

agent.transaction.pool.size
>   number of pre-allocated transaction instances for the transaction pool. Increase this value if high
>   frequency transaction measurement is expected. When the instrumented application stops or the

resource watchdog concept is enabled, a short message is written to the log file which reports the number of used transaction instances. With this information, it is possible to adjust the initial transaction pool size so that no memory allocation is needed during runtime.

Default is *128*. Minimum is *32*, Maximum is defined by the `agent.transaction.pool.max` property.

Bindings: arm40c, arm40java, arm40c#.

`agent.transaction.pool.max`
maximum number of transaction instances which are allocated by MyARM. If this limit is reached and the instrumented application starts a new transaction measurement, an error is returned and the transaction dropped counter is increased by one.

Default is *1024*. Minimum is *512*, Maximum is *8192*.

Bindings: arm40c, arm40java, arm40c#.

`agent.transaction.waiting`
waiting time in milliseconds to try to process an transaction before dropping it. In environments with high measurement frequencies increasing this waiting time will result in less dropped transactions.

Default is *1* millisecond. Minimum is *0* (disabling waiting), Maximum is *1000*.

Bindings: arm40c, arm40java, arm40c#.

`agent.metric.pool.size`
number of pre-allocated metric instances for the metric pool. Increase this value if high frequency transaction measurement with metrics are expected. When the instrumented application stops or the resource watchdog concept is enabled, a short message is written to the log file which reports the number of used metric instances. With this information, it is possible to adjust the initial metric pool size so that no memory allocation is needed during runtime.

Default is *64*. Minimum is *16*, Maximum is defined by the `agent.metric.pool.max` property.

Bindings: arm40c, arm40java.

`agent.metric.pool.max`
maximum number of metric instances which are allocated by MyARM. If this limit is reached and the instrumented application tries to allocate new metrics a warning is returned and the metric dropped counter is increased by the number of metrics that could not be allocated.

Default is *512*. Minimum is *256*, Maximum is *4096*.

Bindings: arm40c, arm40java.

`agent.log`
configures the logging facility of the agent. See appendix "Configuring log facility" for details.

Bindings: arm40c, arm40java, arm40c#.

`agent.log.datasink.level`
configures the logging level of log messages which also should be written to the datasink and not only to the configured logging destination. With this feature, logs can be centralized within the configured database.

Default is *error*.

Bindings: arm40c, arm40java, arm40c#.

`agent.filestorage`
configures the agent to use (true) or not to use (false) the file storage concept for temporarily storing ARM data if the configured datasink is not available.

Default is *true*.

Bindings: arm40c, arm40java.

C.3.1 Configuring agent binding API

agent.binding.api.return_ok
> boolean which defines whether all API calls should return ok (zero) even if an error was detected.
>
> Default is *false*.

C.4 Configuring the log facility

ARM postulates that in any circumstance the instrumented application has to continue working regardless of errors within the ARM agent or with the current instrumentation. Therefore any error or inconsistency within the ARM agent has to be hidden from the application. But in many situations the instrumentor should know if an error occurred within the ARM agent. Therefore, MyARM supports error and information reporting which can be configured using the following configuration parameters:

<component>.log.type
> Specifies the type of the reporting facility. One of the following types can be used:
>
> **none** No reporting at all
>
> **file** Reports to the file specified via `log.file`
>
> **syslog** Reports to the system log service (Unix)
>
> **stderr, stdout** Reports to standard error/output (file) channel (Unix)

<component>.log.level
> Specifies the level of detail for logging. Currently the following levels are supported:
>
> **none** no logging at all
>
> **error** only log errors
>
> **warning** log warnings and errors
>
> **status** also log status information
>
> **info** some more information
>
> **config** log the read configuration of the different MyARM components
>
> **max** maximum logging information

<component>.log.datasink.level
> Specifies the log level for which log messages should also be reported to the datasink (e.g. will be stored in the centralized database). Default is *error*.

<component>.log.file
> Specifies the file to write any reporting information to.

<component>.log.file.mode
> Specifies the mode to open the file specified via `log.file`. Mode can be either "w" to open the log file and possibly truncate the existing file or "a" to append to the existing file.

<component>.log.file.size
> If `log.file.generation` is enabled, this property specifies the maximum size of the log file in bytes. If this limit is reached, a new log file will be opened.
>
> Default is *1048576* (1MB).

`<component>.log.file.generation`
>   Specify the maximum number of log files which will be generated by the MyARM log component. In conjunction with the `log.file.size` property, MyARM will not use more bytes of hard disk space than the arithmetic product of these two properties. If this property is zero, the log file generation rotation is disabled and only one log file will be used. The latest log file has a suffix of `.0` and the oldest log file generation has a suffix of `.<generation>-1`.
>
>   Default is *5*.

`<component>.log.format`
>   Specifies a template string used to format the log message. The following format specifiers can be used:
>
>   **%A** is replaced with the name of the MyARM application (e.g. myarmdaemon). For the agent log the last registered application definition name is used.
>
>   **%P** is replaced with the current process id.
>
>   **%H** is replaced with the current host name.
>
>   **%T** is replaced with the internal thread id.
>
>   **%C** is replaced with the component name.
>
>   **%N** is replaced with the log message number.
>
>   **%I** is replaced with the log message ID.
>
>   **%M** is replaced with the log message. If this specifier is omitted, the log message is appended at the end of the line.

`<component>.log.show.level`
>   Indicates that the log level should be logged (`true`) or not (`false`).

`<component>.log.show.time`
>   Indicates that the time of the log should be logged (`true`) or not (`false`).

`<component>.log.flush`
>   Specifies, if true, that after each log message a flush should be performed so that it is immediately written to its destination without any buffering. Error log messages will be flushed always.

`<component>.log.syslog.facility`
>   Specifies the facility to use for syslog service. It supports 'local0' to 'local7' and 'user'. Default is 'user'.

C.5   Configuring the resource watchdog

The following configuration properties controls logging of resources:

`<component>.resource.watchdog.period`
>   defines the period to log the resource usage. If the period is zero, the resource usage is only reported on program termination.
>
>   Default is *5m* (5 minutes), minimum is *0s* (0 seconds), maximum is *1h* (1 hour).

`<component>.resource.watchdog.log.entities`
>   boolean which indicates to log (`true`) recorded, processed, stored and read entities or not (`false`).
>
>   Default is *false*.

`<component>.resource.watchdog.log.tranpool`
>   boolean which indicates to log (`true`) transaction pool allocation or not (`false`).
>
>   Default is *true*.

`<component>.resource.watchdog.log.metricpool`
	boolean which indicates to log (`true`) metric pool allocation or not (`false`).

	Default is *false*.

`<component>.resource.watchdog.log.armdatapool`
	boolean which indicates to log (`true`) ARM data buffer pool allocation or not (`false`).

	Default is *true*.

`<component>.resource.watchdog.log.dropped`
	boolean which indicates to log (`true`) dropped entities or not (`false`).

	Default is *true*.

`<component>.resource.watchdog.log.transaction.calls`
	boolean which indicates to log (`true`) number of calls to the appropriate ARM 4.0 C transaction function calls. Especially for *arm_start_transaction()* and *arm_stop_transaction()*.

	Default is *true*.

The `<component>` can be either:

- '`daemon`' for configuring the `myarmdaemon`.

- '`agent`' for configuring the watchdog for any instrumented application.

C.6   Configuring the datasink component

This section describes how to configure a datasink which should be used to store all ARM data. MyARM supports a wide range of different datasinks, therefore the configuration of a datasink depends on its type. The common datasink properties are described below. For specific datasink configuration such as for a MySQL or MariaDB database see the appropriate sections in datasinks.

`<name>.type`
	specifies the type of the datasink (e.g. mysql or sqlite3).

The `name` placeholder can be replaced by any other word. For example if you use a MySQL database you can use the word `db_mysql` to clearly specify which datasink you configure. To actually use this datasink configuration just specify the name as the value for the `<component>.sink.name` property of the appropriate component. For example, to use the MySQL datasink directly from the instrumented application use the following lines in your configuration file:

```
agent.sink.name = db_mysql

db_mysql.type               = mysql
db_mysql.host               = myhost
db_mysql.user               = myarm
db_mysql.database.definition = MyARM_Def
db_mysql.database.application= MyARM_App
db_mysql.database.transaction= MyARM_Tran
```

Configuration C.5: **MySQL example**

C.7   Configuring command line tools

The following configuration properties are used to change the behaviour of all command line tools.

`tools.source.name`
> configures the name of the datasource to be used. This property is normally set by a template configuration file. For example the `tcp_sqlite3.conf` template file contains the following lines:

```
# set sink for transporting ARM data to the myarmdaemon
agent.sink.name = sink_tcp
# set myarmdaemon database
daemon.sink.name = db_sqlite3
daemon.source.name = db_sqlite3
daemon.collection.mode = tcp

# set database name for command line tools
tools.source.name = db_sqlite3
tools.sink.name = db_sqlite3

include main.conf
```

Configuration C.6: **TCP SQLite3 template example**

`tools.sink.name`
> configures the name of the datasink to be used by all tools (e.g. myarmimport). This property is normally set by a template configuration file. See template example above.

`tools.rtformat`
> configures the response time format used by all tools. See "Configuring response time formats" for details.
>
> Default is *ms*.

C.8   Configuring date formats

Date values can be represented in various ways. MyARM supports the following date formatting patterns. Examples are given for the eleventh December 2009.

DD.MM.YYYY – day.month.year; for example `11.12.2009`

YYYY-MM-DD – year-month-day; for example `2009-12-11`

MM/DD/YYYY – month/day/year; for example `12/11/2009`

DD.MM.YY – day.month.year with two digit year; for example `11.12.09`

YY-MM-DD – year-month-day with two digit year; for example `09-12-11`

MM/DD/YY – month/day/year with two digit year; for example `12/11/09`

C.9   Configuring time formats

Time values can be represented in various ways. MyARM supports the following time formatting patterns:

HH:MM:SS – `hour:minute:second`; for example `16:29:37` for nearly half past four in the afternoon.

HH:MM:SS.ZZZ – `hour:minute:second.millisecond`; for example `16:29:37.546` for nearly past four in the afternoon including millisecond precision.

HH:MM:SS AP – `hour:minute:second`; in 12-hour clock notation for example `04:29:37 PM` for nearly half past four in the afternoon.

HH:MM:SS.ZZZ AP – `hour:minute:second.millisecond`; in 12-hour clock notation for example `04:29:37.546 PM` for nearly half past four in the afternoon including millisecond precision.

C.10   Configuring response time formats

Response times can be in a wide range of time intervals according to their corresponding transactions. For long running transactions it is not useful to display the response time in micro- or even nanoseconds. But if you want to measure low level transactions it may be necessary to use such short time units. For this purpose, you can configure how MyARM tools display and parse response times:

rtfmt can be one of the following strings (letters in parentheses can be omitted):

ns response times in nanoseconds

μs or mic(s) response times in microseconds

ms response times in milliseconds, nanoseconds remainder is omitted

s(ec) response times in seconds, nanoseconds remainder is omitted

m(in) response times in minutes, microseconds remainder is omitted

h(our) response times in hours, microseconds remainder is omitted

d(ay) response times in days, milliseconds remainder is omitted

Appendix D

Command line options

D.1 Standard options

The following set of options are supported by all MyARM commands line tools:

-cf *conf_file* **, --conf-file** *conf_file*
> specifies the configuration file to use instead of the default one.

-h, -?, --help
> prints a help page.

--version
> prints the current version number of the program.

--build
> prints the current build number of the program.

--edition
> prints the current edition name the program belongs to.

The following options allow to override the configured datasink or datasource.

-so *source-url* **, --source-url** *source-url*
> The *source-url* is a database location expressed in the common URL notation. A detailed description of supported database URLs is in section "Database URL notation".
>
> For example, an exported XML file can be analysed if all relevant data are exported into that file. The command:

```
myarmquery --source-url xml:///export_data.xml Apache2
```

> will print any transactions of the ARM application Apache2 found in the `export_data.xml` XML file.

-si *sink-url* **, --sink-url** *sink-url*
> The *sink-url* is a database location where ARM data should be written to. It uses the common URL notation as described in the "Database URL notation". section. This option can be used to override the configured datasink with MyARM tools which write ARM data to a database such as `myarmexport`.

D.2 Constraints options

The following set of options can be used by any MyARM command line tool which operates on transaction data such as myarmquery or myarmexport:

-ai *<op><value>*, **--app-instance** *<op><value>*
> specifies the application instance *<value>* to match application or transactions instances. Supported *<op>* operators can be found in section "Constraint value operators".

-ag *<op><value>*, **--app-group** *<op><value>*
> specifies the application group *<value>* to match application or transactions instances. Supported *<op>* operators can be found in section "Constraint value operators".

--context *<name><op><value>*
> with the `--context` option, a context property filter can be specified in the form of a `<key><op><value>` tuple. Supported *<op>* operators can be found in section "Constraint value operators".

-u *user*, **--user** *user*
> specifies the user name for each matching transaction.

--uri *<op><value>*
> specifies the URI *<value>* for matching transactions. Supported *<op>* operators can be found in section "Constraint value operators".

-ts *tran-status*, **--tran-status** *tran-status*
> specifies the ARM transaction status to be matched.

-sn *system-name*, **--system-name** *system-name*
> specifies the system name to match application or transaction instances.

--rt-min *min*
> specifies the minimum response time to match a transaction.

--rt-max *max*
> specifies the maximum response time to match a transaction.

-sf *time*, **--start-from** *time*
> specifies the start time from which to select transactions.

-su *time*, **--start-until** *time*
> specifies the start time until transactions should match.

--utc *time*
> specifies that the from and/or until start time is expressed in UTC.

-oc, **--only-children**
> specifies that only transaction with children should match.

-or, **--only-root**
> specifies that only transactions which are root transactions of a transaction tree should match.

D.2.1 Formatting options

To interpret time and response time arguments correctly, you can set the appropriate format specifications with the following options:

–tmf *string*, **--time-fmt** *string*
>   Use *string* for formatting and scanning times. See appendix "Configuring time formats".

–dtf *string*, **--date-fmt** *string*
>   Use *string* for formatting and scanning dates. See appendix "Configuring date formats".

–rf *string*, **--rt-fmt** *string*
>   Specifies how the response time is displayed or parsed. See appendix "Configuring response time formats".

D.2.2   Sorting options

If the underlying database supports sorting of data you can use the following options to specify sort criteria and order.

–sb *criteria*, **--sort-by** *criteria*
>   specify by which criteria the output of transaction or application data should be sorted. Currently valid sorting criteria are:

>   **None**   no sorting at all
>   **Start**   sorting by start time of the transactions
>   **Stop**   sorting by stop time of the transactions
>   **Duration**   sorting by response time of the transactions
>   **Arrival**   sorting by arrival time of the transactions
>   **Blocked**   sorting by blocked time of the transactions

–sd, --sort-descending
>   specify descending sort order. By default it is ascending.

D.3   Constraint value operators

OPERATOR	DESCRIPTION
~=	the string is interpreted as a pattern as supported by the underlying (SQL) database. The value can include a single '%' character matching any number of characters or '_' matching exactly one character. To use the '%' or '_' characters inside the value string use '\ ' to escape it, e.g. '\%'.
=	matches any string which equals to this string
!=	matches any string which is not equal to this string
>	matches any string which is greater than this string
<	matches any string which is less than this string

Table D.1: **Constraint value operators**

D.4   Application and transaction names

Application and/or transaction names are supported by various MyARM commands and are specified in the form <appName>:<tranName>. The application and transaction name is separated by a colon (':').

If you want to select only a specific transaction name for any application, just provide an empty application name `:<tranName>`.

If the application or transaction name contains a colon (`':'`) it must be escaped with a second colon. For example the application name `WebSphere:APPLICATION_SERVER` must be escaped and specified as follows:

```
myarmquery WebSphere::APPLICATION_SERVER
```

And to select a specific transaction a single colon delimits the application name from the transaction name:

```
myarmquery WebSphere::APPLICATION_SERVER:URI
```

Appendix E

Troubleshooting

This section describes some hints to solve common problems which may occur during deployment and operation of the MyARM agent.

E.1 Tracing initialization and configuration

It is possible to enable a tracing mechanism which helps you to identify the deployment problem you might have. This can be enabled using the `MYARM_TRACE` environment variable. For a detailed description of the options please see Appendix Environment.

E.2 Some transactions are missing

MyARM limits its resource usage (memory) in order to work properly, even if the instrumented application uses the ARM API incorrectly. Therefore it is possible that some transaction measurements are missing due to resource shortage. If more transaction response time measurements are active at a time than the number of configured transaction instances, some response time measurements are dropped. This situation is logged with `ARM DATA DROPPED` log message to notify about the lost data.

The ARM data buffer pool size (`basic.armdata.buffer.pool.size`) and the size of each ARM data buffer (`basic.armdata.buffer.size`) can be configured (See ARM data buffer configuration).

E.3 MySQL database creation failed

To minimize the efforts to setup all necessary databases and tables for the MyARM agent when using a MySQL server to store ARM data, the program `myarminitdb` is provided. If the MySQL server is not setup correctly it is possible that the `myarminitdb` fails to create the needed databases and tables. Be sure that you have the right permissions for creating databases and tables.

```
someuser@localhost: mysql -u root
GRANT ALL PRIVILEGES ON *.* TO 'someuser'@'localhost';
```

Note that you need root privileges to change the privileges of databases.

Appendix F

Miscellaneous

F.1 Scripts

F.1.1 MyARM daemon support script

MyARM agent has daemon processes described in section `myarmdaemon` which is working in background and provide different kind of services. To simplify the starting and stopping of this daemon process the following script is provided:

myarmdaemon.sh starts, stops and provides basic information about the running `myarmdaemon`.

Usage:

```
myarmdaemon.sh [-q] [-v] start|stop|restart|status|info
```

This script can also be used as an `init.d` script used to start the `myarmdaemon` process during boot time of the machine.

The following parameter can be passed to the script:

-q
   enables quiet mode. Nothing will be printed to stdout.

-v
   enables verbose mode. Prints the used MyARM environment if it was sourced.

start
   tries to start the `myarmdaemon`.

stop
   tries to stop the `myarmdaemon`.

restart
   tries to stop and restart the `myarmdaemon`.

status
   checks if the `myarmdaemon` is running or not.

info
   if the `myarmdaemon` is running it prints basic information of the current `myarmdaemon` process.

F.2 Database URL notation

In MyARM a database is normally configured using MyARM configuration properties. If a database configuration should be specified (for example) on the command line these properties does not work. For this purpose MyARM provides an URL-like notation for each supported database type. The type of the database is defined by the scheme (or normally known as protocol prefix) of the URL. Currently the following MyARM database URL types are supported:

sqlite3:///<sqlite_db_file>
> defines a SQLite database where `<sqlite_db_file>` is the file name of the sqlite database file. The name can be an absolute or a relative file name.

xml:///<xml_file>
> defines a XML file where `<xml_file>` is the file name of the XML file. The name can be an absolute or a relative file name.

myarm:///<myarm_file>
> defines a MyARM data file where `<myarm_file>` is the file name of the MyARM data file. The name can be an absolute or a relative file name (e.g. `../myapp.myarm` or `/opt/data/myapp.myarm`).

mysql://<password>:<user>@<host>:<port>/<tranDB>[?parameters]
> defines a MySQL database where `<password>` and `<user>` is the password and user name to log into the MySQL database. The `<host>` specifies the host name where the MySQL database is running on and using `<tranDB>` as the MySQL database instance.
>
> The following optional `parameters` can be specified:
>
> **connections=num**
> > The optional `connections` parameter can be used to specify the number of parallel active connections.
>
> **config=database**
> > The optional `config` parameter can be used to specify the database where MyARM stores configuration data.

Appendix G

Using MySQL

This appendix describes how to configure MyARM for using MySQL database with high transaction measurement numbers. For general information regarding MyARM or MySQL please consult the appropriate user manuals.

G.1 Motivation

MyARM is designed to measure and store (unlike other performance monitoring tools) all measured transactions of an ARM instrumented application into a connected database. This can result in high number of measurements which need to be stored into the database. This document describes how to configure MyARM using the MySQL database to achieve an optimal transaction `INSERT` throughput.

G.2 Scenarios

The ARM standard defines a variety number of optional features which influence the throughput of storing transactions into a database we need to define typical classes of scenarios. The following list defines the most common (items 1 to 5) used and worth case (items 6 and 7) ARM transaction measurements:

1. Simple transaction measurement without any additional data (Simple).

2. Simple transaction measurement with a parent correlator (ParentCorr).

3. Transaction measurement with 3 gauge metrics attached (Metrics).

4. Transaction measurement with 1 context value string with a size of 32 Bytes (Context 1).

5. Transaction measurement with 5 context value string with a size of 32 Bytes (Context 5).

6. Transaction measurement with 10 context value string with a size of 32 Bytes (Context 10).

7. Transaction measurement with 20 context value string with a size of 32 Bytes (Context 20).

Some of these scenarios are likely to be combined. For example it is quite usual that a transaction measurement has correlators as well as metrics and context properties. The tested scenarios should help to get an overview how additional data influence the overall database transaction throughput.

G.3   Test design

The following test design is used to get the transaction storing rate into a MySQL database. A simple C program is used to generate ARM transactions at a maximum rate. MyARM is configured to buffer all transaction measurements and write the buffered transactions into the MySQL database. When all transactions are written into the MySQL database the application terminates.

The elapsed real time divided by the number of transactions stored in the MySQL database gives an good overview of the maximal transaction throughput into the MySQL database.

To increase the maximal transaction throughput MyARM supports a so-called `thread` datasink which uses a defined number of threads to write data to a destination, here MySQL, datasink. Each thread open its own connection to the MySQL database. All test scenarios described above are executed with one, four, eight, ten or twelve MySQL datasink connections.

G.3.1   Hardware

All tests were executed on an Intel® Xeon® CPU E5-2660 v2 (10 cores with hyper threading) running Debian Linux 7.8 (amd64) with 128GB of memory and a 512GB Samsung SSD 840 PRO hard disk.

G.3.2   MySQL configuration

The MySQL version 5.5.43 (x86_64) is used and databases were created using the INNODB storage engine with the following parameters changes according to standard debian installation:

innodb_file_per_table – is set

max_binlog_size – is disabled

For a detailed description of the modified MySQL variables please consult the MySQL user manual.

G.3.3   MyARM configuration

The standard MyARM 4.0.x.0 configuration is used with the following property changes:

db_mysql.connections – is set to `1`, `4`, `8`, `12` or `16` (maximum allowed value).

basic.armdata.buffer.size – is set to `196608`.

basic.armdata.buffer.pool.max – is set to `1024`.

agent.transaction.pool.max – is set to `2048`.

agent.metric.pool.max – is set to `8192`.

These property changes are made to support very high transaction measurement rates of the MyARM ARM 4.0 C agent. For a detailed description of the modified MyARM properties please consult the MyARM user manual.

G.4   Results

The Figure G.1 shows the results of all tests as described above. First of all the number of MySQL database connections influence the overall transaction throughput in any scenario. Thus if high transaction rates are expected configure MyARM to use more MySQL database connections (`db_mysql.connections`).

Within this test setup `12` MySQL database connections performs best. As a rule of thumb the number of MySQL database connections should be around the number of physical CPU cores to get best results. Within our test scenario 10-14 MySQL database connections produced the highest transaction rates per second. Above these numbers the transaction rate per second descreases slightly.

The shown results are the average transaction per second values from the last 10 MyARM builds of the version MyARM 4.0.x.0.
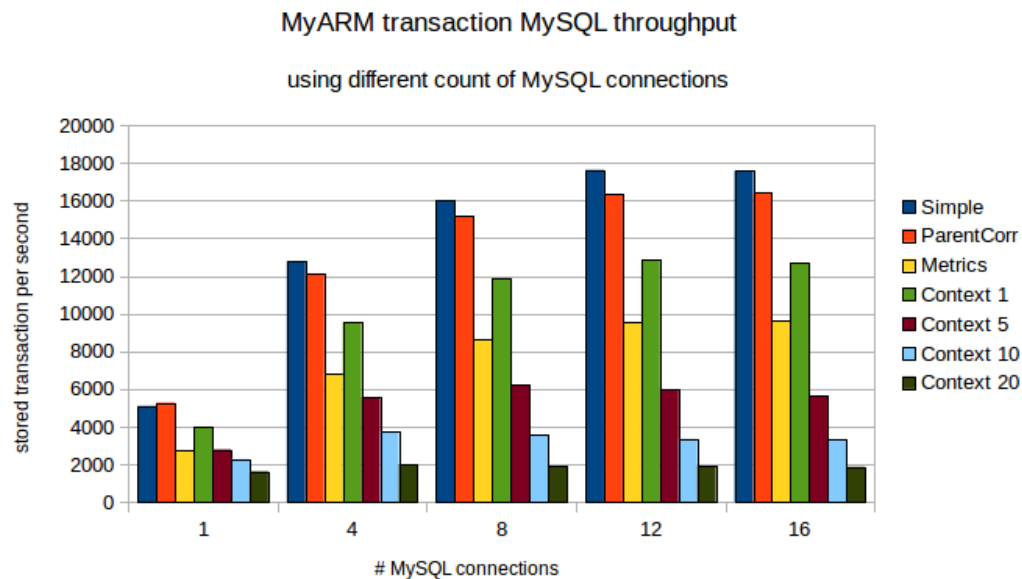
Figure G.1: MyARM transaction MySQL throughput

In real world ARM instrumented application a mixture of the outlined scenarios are present. However the following rules should be taken into account for instrumenting applications with ARM:

- Add only information (metrics, context properties) which are of real interest

- Generate only a correlator if its passed to a sub-transaction.

- Prefer context properties instead of metrics if possible.

- Use diagnostic detail for failed or aborted transactions instead of repeating information in any transaction for diagnosing errors.

The following table lists all average transaction throughput rates in detail:

Threads	Simple	ParentCorr	Metrics	Context 1	Context 5	Context 10	Context 20
1	5102	5275	2722	4010	2794	2283	1629
4	12760	12138	6821	9553	5562	3729	1994
8	16010	15176	8641	11857	6208	3571	1916
12	17624	16330	9525	12841	5963	3352	1925
16	17578	16404	9609	12686	5627	3329	1871

Table G.1: **MyARM transaction MySQL average throughput (transactions per seconds)**

G.4.1   One MySQL database connections result details

Version	Simple	ParentCorr	Metrics	Context 1	Context 5	Context 10	Context 20
4.0.5685.0	4659	5462	2767	3720	2828	2266	1626
4.0.5684.0	5261	5190	2679	3866	2876	2267	1600
4.0.5683.0	5086	5165	2739	4127	2820	2328	1636
4.0.5682.0	5031	5260	2708	4084	2780	2284	1657
4.0.5679.0	5343	5448	2659	4177	2850	2267	1625
4.0.5677.0	5056	5083	2770	4008	2716	2231	1635
4.0.5676.0	5220	5291	2673	3911	2833	2263	1668
4.0.5675.0	4980	5286	2680	4139	2713	2310	1639
4.0.5674.0	5330	5272	2895	3989	2655	2288	1589
4.0.5672.0	5054	5288	2653	4081	2865	2321	1618

Table G.2: **MyARM transaction MySQL throughput with one MySQL database connection (transactions per seconds)**

G.4.2 Four MySQL database connections result details

VERSION	SIMPLE	PARENTCORR	METRICS	CONTEXT 1	CONTEXT 5	CONTEXT 10	CONTEXT 20
4.0.5685.0	12755	12040	6675	9680	5829	3567	1990
4.0.5684.0	12666	11730	6763	9606	5889	3578	1904
4.0.5683.0	12755	12128	6856	9661	5788	4085	2034
4.0.5682.0	12634	12300	6856	9610	5211	3954	2060
4.0.5679.0	12399	12269	6870	9541	5817	3454	2032
4.0.5677.0	12642	12307	6718	9124	4948	3533	2014
4.0.5676.0	12738	12121	6772	9551	5827	3648	1986
4.0.5675.0	12861	11968	6922	9732	5223	3814	1976
4.0.5674.0	13114	12437	6879	9451	5224	3693	1969
4.0.5672.0	13037	12077	6903	9573	5865	3965	1973

Table G.3: **MyARM transaction MySQL throughput with four MySQL database connections (transactions per seconds)**

G.4.3 Eight MySQL database connections result details

VERSION	SIMPLE	PARENTCORR	METRICS	CONTEXT 1	CONTEXT 5	CONTEXT 10	CONTEXT 20
4.0.5685.0	16051	15408	8631	11997	6222	3522	1858
4.0.5684.0	16000	14716	8650	11918	6626	3429	1813
4.0.5683.0	16051	15174	8602	11689	6688	3916	2009
4.0.5682.0	15974	15302	8673	11778	6563	3570	1963
4.0.5679.0	15910	15255	8550	11947	6618	3658	2069
4.0.5677.0	15748	14958	8650	11890	5973	3462	1965
4.0.5676.0	16366	15432	8485	11954	5750	3615	1945
4.0.5675.0	15835	15060	8654	11661	5595	3727	1841
4.0.5674.0	16025	15232	8833	12106	5360	3317	1828
4.0.5672.0	16142	15220	8680	11634	6684	3490	1864

Table G.4: **MyARM transaction MySQL throughput with eight MySQL database connections (transactions per seconds)**

G.4.4   Twelfe MySQL database connections result details

VERSION	SIMPLE	PARENTCORR	METRICS	CONTEXT 1	CONTEXT 5	CONTEXT 10	CONTEXT 20
4.0.5685.0	17746	16501	9505	12936	5524	3245	2048
4.0.5684.0	17761	16488	9350	12845	6321	3231	1983
4.0.5683.0	17559	16326	9620	12936	6731	3482	1938
4.0.5682.0	17889	16194	9610	12820	6711	3594	1974
4.0.5679.0	17543	16207	9429	12787	5810	3321	1878
4.0.5677.0	17256	16420	9564	12953	5370	3390	1939
4.0.5676.0	17873	16406	9601	12944	6049	3349	1881
4.0.5675.0	17376	16142	9478	12706	5488	3431	1880
4.0.5674.0	17574	16460	9451	12812	6025	3290	1878
4.0.5672.0	17667	16155	9638	12666	5605	3191	1853

Table G.5: **MyARM transaction MySQL throughput with twelfe MySQL database connections (transactions per seconds)**

G.4.5   Sixteen MySQL database connections result details

VERSION	SIMPLE	PARENTCORR	METRICS	CONTEXT 1	CONTEXT 5	CONTEXT 10	CONTEXT 20
4.0.5685.0	17746	16366	9689	12787	5586	3179	1925
4.0.5684.0	17182	16339	9671	13089	5701	3334	2030
4.0.5683.0	17889	16597	9661	12795	6361	3424	1809
4.0.5682.0	17497	16406	9510	12730	5465	3287	1933
4.0.5679.0	17636	16064	9578	12995	5585	3350	1904
4.0.5677.0	17346	16420	9478	11363	5768	3245	1817
4.0.5676.0	17809	16542	9573	12903	5534	3343	1820
4.0.5675.0	17452	16339	9633	12453	5360	3460	1803
4.0.5674.0	17652	16583	9666	12928	4961	3366	1742
4.0.5672.0	17574	16380	9629	12812	5947	3301	1929

Table G.6: **MyARM transaction MySQL throughput with ten MySQL database connections (transactions per seconds)**

Appendix H

Agent overhead

This document describes overhead produced by the MyARM ARM implementation and the methods how the overhead was measured.

H.1 Performance impact

Measuring any business or technical transactions needs to execute itself. Therefore getting measurement information is not just for free. In order to decide if the instrumentation does not significantly interfere with the real application tasks the performance impact of the instrumentation is good to know. As of its nature it is highly hardware and platform dependent with higher performing processors the performance impact of the measurement decreases expressed in real time.

One key question about instrumentation for measuring performance is the performance overhead of the measurement itself. The interference with the application to measure should be reduced to a minimum. The MyARM ARM implementation is written with this requirement in mind.

H.2 Overhead measurement

ARM can be used in a variety of use cases therefore we measured the overhead of MyARM as described in the following section.

H.2.1 Simulating a real application

Simulation of a real application is done using busy waiting for a defined number of microseconds (application load, t_{app}). Each such task is measured using normal ARM start and stop calls (instrumentation overhead, t_{arm}). The simulation application runs a defined number of seconds. Therefore the total number of ARM transaction measurements N_{arm} and the total time of the application T_{app} (application load) can be easily calculated using the following formulas:

- $T_{app} = 60 seconds$

- $t_{app} = 1 millisecond$

- $N_{arm} = T_{app}/t_{app} * 1000$

When the ARM instrumentation ideally needs no time the application would execute exactly in 60 seconds. This is for sure not true, therefore we measure the overall execution time (R_{app}) of the running application without initialization and termination of the application. The difference between the measured execution

time and the 60 seconds are the overhead for the ARM instrumentation. This can be broken down to a single transaction measurement overhead by the following formula:

- $t_{arm} = (R_{app} - T_{app})/N_{arm}$

H.2.2   ARM optional features

One major factor influencing the performance impact is how many and which context information is associated with a transaction measurement. The different optional features (correlators, metrics and context values) of ARM influences the time spent by the ARM instrumentation in manifold ways. To reflect these options we measured the overhead of MyARM for the following typical ARM instrumentation patterns:

1. Simple transaction measurement using null call library (e.g. `libarm4null.so` for the ARM 4.0 C binding) (`NullCall`).

2. Simple transaction measurement without any additional data (`Simple`).

3. Simple transaction measurement returning a correlator (`CorrGen`).

4. Transaction measurement with 3 gauge metrics attached (`Metrics`).

5. Transaction measurement with 1 context value string of size 100 Bytes (`Ctxt 1`).

6. Transaction measurement with 5 context value string of size 100 Bytes (`Ctxt 5`).

7. Transaction measurement with 10 context value string of size 100 Bytes (`Ctxt 10`).

8. Transaction measurement with 20 context value string of size 100 Bytes (`Ctxt 20`).

H.3   Test environment

All tests were executed on an Intel® Core™ i7-3517UE CPU (CompuLab Intense PC) running under Linux (Ubuntu-based) in 64bit using kernel 3.13.0-24-generic.

H.4   Agent bindings

The following figure shows the transaction measurement overhead for the simulated application and the ARM optional features as described in section ARM optional features for the ARM 4.0 C-, Java- and C#-Agent.
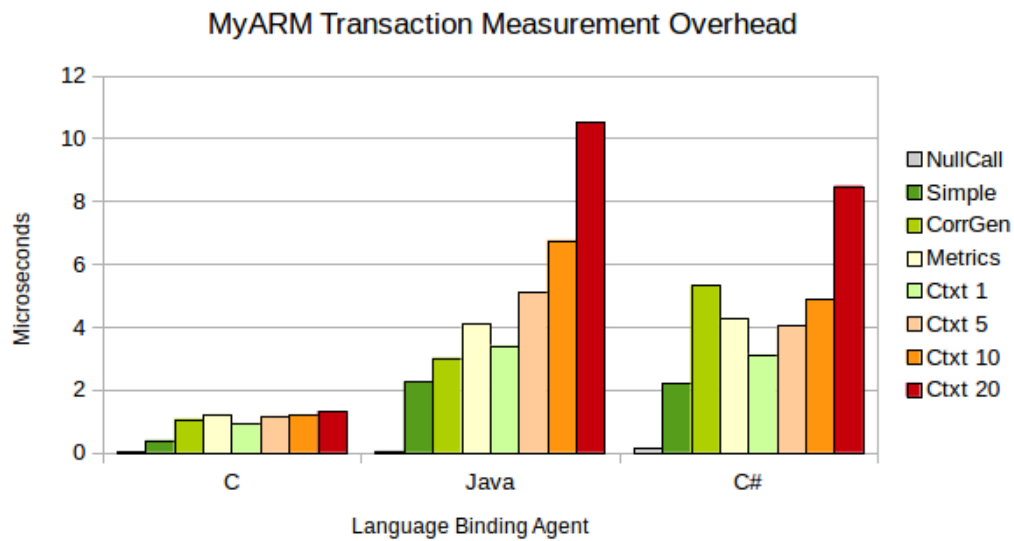
Figure H.1: MyARM ARM 4.0 transaction measurement overhead

H.5 ARM 4.0 C Binding

H.5.1 Overhead

This section describes the transaction measurement overhead of the ARM 4.0 C binding using the `arm_start_transaction()` and `arm_stop_transaction()` function calls.

Binding	NullCall	Simple	CorrGen	Metrics	Ctxt 1	Ctxt 5	Ctxt 10	Ctxt 20
C	0.016	0.388	1.073	1.229	0.936	1.164	1.212	1.320

Table H.1: **MyARM average transaction measurement overhead for the ARM 4.0 C binding (in microseconds))**

The shown results are the average transaction measurement overhead values from the last 10 MyARM builds of the version MyARM 4.0.x.0. See the following table for all detailed results:

H.5.1.1   Details

VERSION	NULLCALL	SIMPLE	CORRGEN	METRICS	CTXT 1	CTXT 5	CTXT 10	CTXT 20
4.0.5685.0	0.009	0.434	1.090	1.194	0.900	1.171	1.205	1.327
4.0.5684.0	0.012	0.325	1.042	1.153	0.893	1.102	1.156	1.321
4.0.5683.0	0.011	0.336	1.005	1.196	0.897	1.095	1.165	1.257
4.0.5682.0	0.018	0.485	1.126	1.314	0.957	1.162	1.217	1.302
4.0.5679.0	0.019	0.420	1.082	1.234	0.961	1.195	1.249	1.336
4.0.5677.0	0.017	0.399	1.081	1.259	0.967	1.187	1.214	1.315
4.0.5676.0	0.022	0.343	1.116	1.262	0.957	1.191	1.233	1.317
4.0.5675.0	0.017	0.372	1.068	1.249	0.937	1.199	1.239	1.375
4.0.5674.0	0.016	0.380	1.037	1.222	0.923	1.186	1.220	1.324
4.0.5672.0	0.021	0.382	1.083	1.207	0.971	1.151	1.219	1.324

Table H.2: **MyARM transaction measurement overhead for the ARM 4.0 C binding (in microseconds)**

H.6   ARM 4.0 Java Binding

H.6.1   Overhead

This section describes the transaction measurement overhead of the ARM 4.0 Java binding using the `ArmTransaction.start()` and `ArmTransaction.stop()` method calls.

BINDING	NULLCALL	SIMPLE	CORRGEN	METRICS	CTXT 1	CTXT 5	CTXT 10	CTXT 20
Java	0.059	2.255	3.012	4.103	3.408	5.110	6.756	10.524

Table H.3: **MyARM average transaction measurement overhead for the ARM 4.0 Java binding (in microseconds))**

The shown results are the average transaction measurement overhead values from the last 10 MyARM builds of the version MyARM 4.0.x.0. See the following table for all detailed results:

H.6.1.1   Details

VERSION	NULLCALL	SIMPLE	CORRGEN	METRICS	CTXT 1	CTXT 5	CTXT 10	CTXT 20
4.0.5685.0	0.059	2.327	2.961	3.952	3.687	5.043	6.399	10.466
4.0.5684.0	0.059	2.159	2.721	4.030	3.149	4.967	6.507	10.362
4.0.5683.0	0.041	2.122	2.899	3.788	3.197	5.013	6.509	10.400
4.0.5682.0	0.069	2.291	3.248	4.166	3.527	5.261	7.043	10.578
4.0.5679.0	0.059	2.346	3.189	4.111	3.440	5.280	6.836	10.529
4.0.5677.0	0.070	2.264	2.997	4.240	3.382	5.037	7.076	10.368
4.0.5676.0	0.047	2.236	2.921	4.204	3.419	5.165	6.882	10.619
4.0.5675.0	0.052	2.269	3.212	4.199	3.352	5.077	6.746	10.381
4.0.5674.0	0.063	2.274	3.008	4.264	3.428	5.053	6.749	10.764
4.0.5672.0	0.066	2.263	2.967	4.075	3.503	5.200	6.812	10.770

Table H.4: **MyARM transaction measurement overhead for the ARM 4.0 Java binding (in microseconds)**

H.7   ARM 4.0 C# Binding

H.7.1   Overhead

This section describes the transaction measurement overhead of the ARM 4.0 C# binding using the `IArmTransaction.start()` and `IArmTransaction.stop()` method calls.

BINDING	NULLCALL	SIMPLE	CORRGEN	METRICS	CTXT 1	CTXT 5	CTXT 10	CTXT 20
C#	0.159	2.208	5.330	4.282	3.124	4.048	4.890	8.488

Table H.5: **MyARM average transaction measurement overhead for the ARM 4.0 C# binding (in microseconds))**

The shown results are the average transaction measurement overhead values from the last 10 MyARM builds of the version MyARM 4.0.x.0. See the following table for all detailed results:

H.7.1.1   Details

Version	NullCall	Simple	CorrGen	Metrics	Ctxt 1	Ctxt 5	Ctxt 10	Ctxt 20
4.0.5685.0	0.128	2.056	5.219	5.510	4.083	4.876	4.664	7.844
4.0.5684.0	0.129	1.791	4.832	3.708	3.994	3.085	4.695	9.915
4.0.5683.0	0.133	1.914	5.084	5.238	2.384	3.437	4.727	8.159
4.0.5682.0	0.169	2.256	5.428	4.401	2.761	5.635	5.195	8.021
4.0.5679.0	0.177	2.403	5.589	4.296	2.732	3.719	5.025	8.505
4.0.5677.0	0.180	2.365	5.727	4.078	2.724	3.979	4.903	10.188
4.0.5676.0	0.176	2.394	5.418	4.213	2.780	3.362	4.842	8.477
4.0.5675.0	0.219	2.993	5.908	4.078	2.786	5.436	5.205	7.983
4.0.5674.0	0.141	1.935	5.130	3.691	2.421	3.353	4.603	7.778
4.0.5672.0	0.140	1.969	4.969	3.602	4.576	3.598	5.036	8.009

Table H.6: **MyARM transaction measurement overhead for the ARM 4.0 C binding (in microseconds)**

Appendix I

Agent reference

This chapter provides a quick reference to the MyARM agent bindings and describes possible error codes and log messages.

I.1 C Language Binding

The ARM 4.0 C language binding [ARM4C] defines error level return codes (negative) and warning level return codes (positive). The following sections describe the possible MyARM return codes:

I.1.1 Error return codes

MYARM_ERROR_NONE (0) – This is not really an error code; it indicates that the operation succeeded successfully.

MYARM_ERROR_NO_MEMORY (-1) – MyARM tried to allocate some memory but this failed.

MYARM_ERROR_NO_OUTPUT_ID (-2) – The pointer to the output `arm_id_t` is null in one the following API calls:

- *arm_register_application()*
- *arm_register_transaction()*
- *arm_register_metric()*

MYARM_ERROR_NO_OUTPUT_HANDLE (-3) – The output handle in one of the following API calls was null:

- *arm_start_application()*
- *arm_start_transaction()*
- *arm_block_transaction()*

MYARM_ERROR_NO_OUTPUT_CORR (-4) – The pointer to the output correlator was null in the *arm_generate_correlator()* API call.

MYARM_ERROR_NO_OUTPUT_VAR (-5) – The pointer to the output variable was null in one of the following API calls:

- *arm_get_correlator_length()*
- *arm_get_arrival()*
- *arm_is_charset_supported()*

MYARM_ERROR_NO_TRAN_DEF (−6) – A null pointer transaction id was passed to one of the following API calls:

- *arm_generate_correlator()*
- *arm_report_transaction()*
- *arm_start_transaction()*

MYARM_ERROR_INVALID_APP_HANDLE (−7) – One of the following API calls were called with an invalid application start handle:

- *arm_stop_application()*
- *arm_report_transaction()*
- *arm_start_transaction()*

This can be the case if an uninitialized handle is used (e.g. application was not started) or the appropriate application was stopped before (passing a handle a second time to *arm_stop_application()* is an error).

MYARM_ERROR_INVALID_TRAN_HANDLE (−8) – One of the following API calls was called with an invalid transaction start handle:

- *arm_stop_transaction()*
- *arm_update_transaction()*
- *arm_discard_transaction()*
- *arm_bind_thread()*
- *arm_unbind_thread()*
- *arm_block_transaction()*
- *arm_unblock_transaction()*

This can be the case if an uninitialized handle is used (e.g. transaction was not started) or the appropriate transaction was stopped before (passing a handle a second time to *arm_stop_transaction()* is an error).

MYARM_ERROR_NO_CORR (−9) – The *arm_get_correlator_flags()* was called with a null pointer for the correlator.

MYARM_ERROR_INVALID_ENCODING (−10) – The *arm_register_application()* was called with a charset that is not supported by MyARM.

MYARM_ERROR_NO_APP_DEF (−11) – The *arm_start_application()* was called with null pointer for the application id.

MYARM_ERROR_NOT_IMPLEMENTED (−12) – This error code is returned by any ARM 4.0 C API call when MyARM could not be initialized correctly. This is an indication of a misconfiguration of MyARM. Be sure all needed environment variables are correctly set.

MYARM_ERROR_NO_NAME (−13) – The mandatory name was not provided (null pointer) to one of the following API calls:

- *arm_register_application()*
- *arm_register_transaction()*
- *arm_register_metric()*

MYARM_ERROR_INVALID_METRIC_FORMAT (-14) – Either the *arm_register_metric()* was called with an invalid metric format or the metric format of a metric passed within the `arm_subbuffer_metric_values_t` on the following functions does not match its metric binding:

- *arm_report_transaction()*
- *arm_start_transaction()*
- *arm_stop_transaction()*
- *arm_update_transaction()*

MYARM_ERROR_INVALID_METRIC_USAGE (-15) – The *arm_register_metric()* was called with an invalid metric usage.

MYARM_ERROR_INVALID_APP_ID (-16) – An invalid application id was passed to one of the following API calls:

- *arm_register_transaction()*
- *arm_register_metric()*
- *arm_start_application()*

This is often the case if *arm_register_application()* failed for some reason.

MYARM_ERROR_INVALID_TRAN_ID (-17) – An invalid transaction id was passed to one of the following ARM 4.0 API calls:

- *arm_report_transaction()*
- *arm_start_transaction()*
- *arm_generate_correlator()*

This is often the case if *arm_register_transaction()* failed for some reason. Or when using the ARM 2.0 API and an invalid transaction id was passed to *arm_start()*.

MYARM_ERROR_INVALID_INDEX (-21) – Context values and metrics are managed as arrays within the MyARM agent. An index passed to MyARM was out of bounds.

MYARM_ERROR_LIMIT_REACHED (-23) – MyARM tried to get a resource but the limit for the resource (transaction instance or metric instance) was reached.

MYARM_ERROR_INVALID_START_HANDLE (-24) – One of the following ARM 2.0 API calls was called with an invalid start handle:

- *arm_update()*
- *arm_stop()*

This can be the case if an uninitialized handle is used (e.g. transaction was not started) or the appropriate transaction was stopped before (passing a handle a second time to *arm_stop()* is an error).

MYARM_ERROR_INVALID_DATA_BUFFER_FORMAT (-25) – One of the following ARM 2.0 API calls was called with an invalid data buffer format:

- *arm_getid()*
- *arm_update()*
- *arm_stop()*

This can be the case if an uninitialized data buffer is used.

MYARM_ERROR_INVALID_API_CALL (−26) – An ARM 4.0 API function was called before any application definition was registered (*arm_register_application()*) or after the last application definition was destroyed (*arm_destroy_application()*).

MYARM_ERROR_INVALID_CURRENT_CORRELATOR (−28) – *arm_report_transaction()* was called with an invalid current correlator. The transaction was dropped.

MYARM_ERROR_INVALID_BLOCK_HANDLE (−29) – The *arm_unblock_transaction()* method was called with an invalid block handle. It seems that there is a *arm_block_transaction()* call missing or the block handle is invalid.

I.1.2   Warning return codes

MYARM_WARNING_UNKNOWN_SUBBUFFER (1) – An unknown sub-buffer was passed to one of the ARM 4.0 C-API calls.

MYARM_WARNING_DIAG_DETAIL_WITH_GOOD_STATUS (2) – A diagnostic detail sub-buffer was passed to *arm_stop_transaction()* call, although the transaction status was good. This is not supported as specified by the standard.

MYARM_WARNING_NOT_STARTED (3) – *arm_stop_transaction()* was called but the transaction was not started by MyARM. This can be the case if something failed within *arm_start_transaction*. For example in low memory situations transactions are not started.

MYARM_WARNING_PROPERTY_COUNT_TOO_LARGE (6) – An `arm_subbuffer_tran_context_t` sub-buffer was passed to the following ARM API functions and the property count field was too large:

- *arm_report_transaction()*
- *arm_start_transaction()*
- *arm_stop_transaction()* (MyARM extension)

MYARM_WARNING_INVALID_PARENT_CORRELATOR (10) – *arm_report_transaction()* or *arm_start_transaction()* was called with an invalid parent correlator.

I.2   Log messages

I.2.1   Log message No. 1: CONFIG PROPERTY

Brief
Log message which reports used configuration properties

Level
config

Message
Property: <**Property**>='<**Value**>' read from config

Arguments

Property – Name of the configuration property

Value – Configured value for the property

Description
For each configuration property this message is issued with its used value. Therefore the complete MyARM configuration is logged to the configured logging destination.

I.2.2   Log message No. 2: CONFIG DEFAULT PROPERTY

Brief
Log message which reports used default configuration properties

Level
config

Message
Property: "<**Property**>" not found using default: '<**DefValue**>'

Arguments

Property – Name of the configuration property

DefValue – Default value used

Description
For each configuration property which can not be found in the current configuration, the used default value is reported with the log message.

I.2.3   Log message No. 3: CONFIG PROPERTY ERROR

Brief
Log message which reports an error for a configuration property

Level
error

Message
Property: <**Property**>='<**Value**>' not valid. Using default: '<**DefValue**>'

Arguments

Property – Name of the configuration property

Value – Configured value for the property

DefValue – Default value used

Description

Some configuration properties allow only a set of defined values. If an unsupported value was specified in the configuration this error message is issued.

I.2.4   Log message No. 4: CONFIG PROPERTY FILE

Brief

Reports which configuration property file was used

Level

config

Message

Property file '<**File**>' used

Arguments

File – Name of the configuration property file used

Description

MyARM can be configured using configuration property files. This log message reports which file is used to read the configuration. Therefore it allows you to check your MyARM environment.

I.2.5   Log message No. 5: CONFIG VAR ERROR

Brief

An environment variable expansion failed for the report property key and value

Level

error

Message

Environment variable expansion failed for <**Key**>='<**Value**>': Variable <**Variable**> <**Reason**>

Arguments

Key – Property key which caused the error

Value – Property value which caused the error

Variable – Variable which caused the error

Reason – Reason of the error

Description

The value of the reported property key contains a reference to an environment variable and the expansion of this reference failed for the reported reason.

I.2.6   Log message No. 6: CONFIG URL ERROR

Brief

The configuration URL has an error

Level

error

Message

Configuration URL '<**URL**>': <**Reason**>

Arguments

> **URL** – The URL which caused the error
>
> **Reason** – Reason of the error

Description

> MyARM uses configuration property files to read user defined configuration properties. A configuration property file is passed to MyARM via the `MYARM_CONFIG_URL` environment variable. The value of this variable is erroneous.

I.2.7 Log message No. 7: CONFIG NOT FOUND

Brief

> The configuration could not be found

Level

> error

Message

> Configuration not found: <**Reason**>

Arguments

> **Reason** – Reason of the error

Description

> The MyARM configuration property file could not be found.

I.2.8 Log message No. 8: CONFIG OUT OF RANGE

Brief

> The value of the configuration property is out of the given range. MyARM adjusted the used value

Level

> error

Message

> Property <**Key**>='<**Value**>' out of range [<**Min**>,<**Max**>] using <**Used**>

Arguments

> **Key** – Property key which caused the error
>
> **Value** – Property value which is out of range
>
> **Min** – Minimum allowed value
>
> **Max** – Maximum allowed value
>
> **Used** – Used value

Description

> The value of the reported property key is out of range and has been adjusted by MyARM.

Action

> Please configure the configuration property to an allowed value.

I.2.9   Log message No. 10: DYNAMIC OBJECT FAILED

Brief

A dynamic object (e.g. datasink or datasource) could not be loaded

Level

error

Message

Loading of dynamic object '<**Object**>' failed: '<**Reason**>'

Arguments

Object – The dynamic object which should be loaded

Reason – Reason code of failure

Description

MyARM uses a modular plugin concept for its datasinks and datasources. These plugins are written as dynamic objects which are loaded during runtime of the application. The reported dynamic object failed to load into memory for the reported reason.

Action

Make sure you have not mixed different MyARM versions, otherwise consult the MyARM support.

I.2.10   Log message No. 11: NO LICENSE

Level

error

Message

No license for <**Name**> available!

Arguments

Name – Name of the component the license check failed

Description

This message indicates that the named component is not licensed to you. It is possible that you mixed components of different MyARM editions.

Action

Please verify your MyARM installation and your license keys.

I.2.11   Log message No. 12: API ERROR

Brief

Reports an API error of the C, Java or C# API bindings

Level

error

Message

<**Method**>(): (rc='<**Code**>') '<**Description**>'. <**Details**>

Arguments

Method – Method or function name for which the API error occurred

Code – Return code of the API method or function

Description – Brief description of the error

Details – More details if available

Description

MyARM detected an error during an API method or function call. This can be caused by passing some invalid parameters, or some sort of ARM standard specification violation.

Action

Please read the MyARM users guide return code reference section for details on a specific error.

I.2.12   Log message No. 13: API WARNING

Brief

Reports an API warning of the C, Java or C# API bindings

Level

warning

Message

<**Method**>(): (rc='<**Code**>') '<**Description**>'. <**Details**>

Arguments

Method – Method or function name for which the API warning occurred

Code – Return code of the API method or function

Description – Brief description of the warning

Details – More details if available

Description

MyARM detected a situation which is not standard compliant during an API method or function call.

Action

Please read the MyARM users guide return code reference section for details on a specific warning.

I.2.13   Log message No. 14: ERROR

Level

error

Message

<**Message**>

Arguments

Message – Error message

Description

General purpose error message which has no special association to a specific MyARM component.

I.2.14   Log message No. 15: STATUS

Level

status

Message

<**Message**>

Arguments

Message – The status message

Description

General purpose status message. This status message is issued within the Web-Analysis server part.

I.2.15   Log message No. 16: ARM DATA DROPPED

Brief

Log message which reports dropped ARM data

Level

error

Message

Could not process ARM data (<**Contents**>) <**reason**>. ARM data has been dropped!

Arguments

Contents – Overview of dropped entity counts (definitions, transactions, etc)

reason – Description of the reason

Description

This message is issued when the ARM data cannot be passed to its datasink and the temporary file storage for storing ARM data has reached its limit or is disabled.

Action

To resolve this problem check why the datasink is not available (database, network) or enable temporary file storage.

I.2.16   Log message No. 17: ARM DATA DROPPED TOTAL

Brief

Log message which reports the total number of dropped ARM data

Level

error

Message

Total number of dropped ARM data: <**Contents**>

Arguments

Contents – Overview of dropped entity counts (definitions, transactions, etc)

Description

This message is issued when the MyARM agent is closed from the instrumented application and reports the total number of dropped ARM data (definitions, application- and transaction instances, metrics and log messages). This log message reports the sum of all dropped ARM data.

I.2.17 Log message No. 18: ARM TRAN CALLS STAT

Level
> info

Message
> Transaction function calls: start=<**StartCalls**>, stop=<**StopCalls**>, update=<**UpdateCalls**>, discard=<**DiscardCalls**>, active=<**ActiveTrans**>

Arguments

> **StartCalls** – Number of calls to the arm_start_transaction() function call
>
> **StopCalls** – Number of calls to the arm_stop_transaction() function call
>
> **UpdateCalls** – Number of calls to the arm_update_transaction() function call
>
> **DiscardCalls** – Number of calls to the arm_discard_transaction() function call
>
> **ActiveTrans** – Number of currently active transaction measurements

Description
> This message reports the number of calls to the appropriate ARM 4.0 C functions. It is useful to monitor start and stop calls as well as the number of currently active transaction measurements during development and test phase of the ARM instrumentation. Also this information gives a hint if MyARM C ARM 4.0 transaction functions are called at all (number of start and stop calls increases).

I.2.18 Log message No. 19: MODULE NOT SUPPORTED

Level
> error

Message
> Configured <**Type**> name='<**Name**>' not supported

Arguments

> **Type** – Type of module which is not supported (e.g. database, datasink or handler)
>
> **Name** – Name of the module

Description
> A configured database, datasink or handler is not supported by the current MyARM installation. Please check the users guide for supported modules.

I.2.19 Log message No. 20: MODULE CREATE ERROR

Level
> error

Message
> Cannot create <**Type**> name='<**Name**>': <**Reason**>

Arguments

> **Type** – Type of the module where a create error occurred (e.g. database, datasink or handler)
>
> **Name** – Name of the module type
>
> **Reason** – Reason for the creation error

Description

The creation of the named module (e.g. database, datasink or handler) was not successful. There are several possible reasons for such a failure:

- Module not found

- Wrong shared object module

- Out of memory

I.2.20 Log message No. 21: MODULE NOT CONFIGURED

Level

error

Message

<**Type**> not configured

Arguments

Type – Type of the module not configured (e.g. database, datasink or handler)

Description

Please check the log file which configuration property caused this error message. The last config log message above this message caused it. The log level should be set at least to 'config'.

I.2.21 Log message No. 22: APP NOT STOPPED

Level

warning

Message

application instance app_handle='<**AppHandle**>' not stopped

Arguments

AppHandle – The handle as returned by the arm_start_application() function

Description

Within the C language binding a previously started application was not stopped before the instrumented application terminated or the libarm4.so was unloaded.

Action

Please verify your ARM instrumentation of the appropriate application or notify the vendor of the instrumented application that there is an ARM instrumentation issue.

I.2.22 Log message No. 23: TRAN NOT STOPPED

Level

warning

Message

transaction instance tran_handle='<**TranHandle**>' not stopped

Arguments

TranHandle – The handle as returned by the arm_start_transaction() function

Description

Within the C language binding a previously started transaction was not stopped before the instrumented application terminated or the libarm4.so was unloaded.

Action

Please verify your ARM instrumentation of the appropriate application or notify the vendor of the instrumented application that there is an ARM instrumentation issue.

I.2.23   Log message No. 24: POOL INFO

Brief

Reports resource allocation and usage information

Level

info

Message

<**Name**> pool info: min=<**Min**>, max=<**Max**>, allocated=<**Alloc**>, current=<**Current**>, total=<**Total**>

Arguments

Name – Name of the resource pool

Min – Configured minimum number of allocated instances

Max – Configured maximum number of allocated instances

Alloc – Maximum allocated instances at one time

Current – Currently used instances at report time

Total – Total number of used instances

Description

MyARM manages various resources in so-called resource pools. The following named resource pools are currently used:

- **Transaction** for storing transaction measurement information
- **Metric** for storing ARM metric information
- **ARMData** for storing serialised ARM data

I.2.24   Log message No. 25: BLOCKED INFO DROPPED

Level

warning

Message

<**Number**> blocked response time information dropped. Reason: '<**Reason**>'

Arguments

Number – The number of dropped blocked times

Reason – The reason why the blocked times were dropped

Description

The specified number of blocked times were dropped due to the reported reason. However, the blocked time is summed up in the global blocked time of the transaction measurement.

I.2.25   Log message No. 26: LICENSED INSTALLS EXCEEDED

Brief
    Number of licensed installations was exceeded

Level
    error

Message
    Number of licensed installations (<**Number**>) exceeded

Arguments
    Number – Licensed installations

Description
    The number of licensed installations (system addresses) was exceeded and measurements were dropped.

Action

    • Please contact MyARM to increase the number of licensed installations

I.2.26   Log message No. 27: INIT FAILED

Level
    error

Message
    Initialisation failed in module='<**Module**>' level=<**Level**>

Arguments
    Module – The module which caused the initialisation error

    Level – The initialisation level the error occurred

Description
    During initialisation an error occurred and the ARM agent is disabled or the daemon or tools terminated directly. Please take a look into the log file to get the cause of the error.

I.2.27   Log message No. 28: AGENT STARTED

Level
    status

Message
    Started MyARM <**Agent**> <**Edition**> Edition Version <**Version**> for <**Platform**><**Licensee**>

Arguments
    Agent – Describes the started agent (e.g. 'ARM 4.0 for C' or 'ARM 4.0 for Java', etc)

    Edition – Describes the MyARM edition of the agent (e.g. 'Java', 'Professional', 'Enterprise', etc)

    Version – Describes the current version of the agent

    Platform – Describes the platform which the MyARM agent runs on

    Licensee – Describes the licensee of the MyARM agent if any

Description
    When this log message is issued the MyARM agent is up and running.

I.2.28   Log message No. 29: THREAD STARTED

Level
>   status

Message
>   <**Name**> thread started

Arguments

>   **Name** – Name of the started thread

Description
>   MyARM uses different threads to fulfil its tasks. This message shows that the named thread was
>   started successfully.

I.2.29   Log message No. 30: THREAD STOPPED

Level
>   status

Message
>   <**Name**> thread stopped

Arguments

>   **Name** – Name of the stopped thread

Description
>   MyARM uses different threads to fulfil its tasks. This message shows that the named thread was
>   stopped.

I.2.30   Log message No. 31: REPEATED MSG

Level
>   info

Message
>   Last log message repeated <**Count**> times

Arguments

>   **Count** – Count of the repeated message

Description
>   The previous log message was issued count number of times.

I.2.31   Log message No. 32: ENTITIES INFO

Level
>   info

Message
>   <**Stage**> entities: defs=<**Defs**>, apps=<**Apps**>, trans=<**Trans**>, metrics=<**Metrics**>,
>   logs=<**Logs**>

Arguments

Stage – The stage to provide entities information for. This can be 'Recorded', 'Processed', 'Stored', 'Read' or 'Forwarded'

Defs – Number of ARM definitions

Apps – Number of ARM application instances

Trans – Number of ARM transaction instances

Metrics – Number of ARM metrics

Logs – Number of log messages

Description
This message shows some internal information from the ARM agent about its handled entities so far. This log message is issued if the resource watchdog component is enabled.

I.2.32   Log message No. 34: DATABASE SQL ERROR

Level
error

Message
Error '<**Error**>' in SQL query='<**Query**>'

Arguments

Error – Describes the occurred error

Query – The SQL-Query which caused the error

Description
The reported SQL-Query failed. Please contact the MyARM support.

I.2.33   Log message No. 35: DATABASE CONNECTED

Level
status

Message
Database connected

Description
Indicates that a connection to the configured database is established.

I.2.34   Log message No. 36: DATABASE LOST CONNECTION

Level
status

Message
Lost connection, try to reconnect

Description
The current connection to the configured database was lost during execution of a SQL command. MyARM tries to reconnect to the database immediately.

I.2.35   Log message No. 37: TCPSINK CONNECT FAILED

Level
> error

Message
> Connect to myarmdaemon at '<**Host**>:<**Port**>' failed. Retry in <**Retry**> seconds. Reason: '<**Reason**>'

Arguments

> **Host** – The host name where the myarmdaemon runs on

> **Port** – The port where the myarmdaemon is listening for incoming new TCP sink connections

> **Retry** – Number of seconds until the next connection attempt to the myarmdaemon

> **Reason** – The reason why the connection failed

Description

I.2.36   Log message No. 38: TCPSINK ERROR

Level
> error

Message
> <**Message**>

Arguments

> **Message** – The TCP sink error message

Description
> This log message is issued whenever a protocol error between the TCP sink and the myarmdaemon is encountered. Also this message can occur if there exist network problems between the TCP sink and the myarmdaemon.

I.2.37   Log message No. 39: TCPSINK CONNECTION IDLE

Level
> status

Message
> Connection is idle since <**Idle**> seconds. Now disconnecting from myarmdaemon

Arguments

> **Idle** – Idle time in seconds of the TCP connection

Description
> The current TCP connection to the myarmdaemon is idle (not used) since the reported time in seconds and therefore the connection is closed.

I.2.38 Log message No. 40: TCPSINK CONNECTED

Level
status

Message
Connected to myarmdaemon

Description
The TCP sink connected to the myarmdaemon successfully. ARM data is now sent to the myarm-daemon.

I.2.39 Log message No. 41: TCPSINK COMMAND

Level
info

Message
Command '<**Cmd**>' processed

Arguments

Cmd – The command which was received and processed

Description
The TCP sink received the specified command from the myarmdaemon and processed it.

I.2.40 Log message No. 42: TCPSINK CONNECTING

Level
status

Message
Connecting to myarmdaemon at '<**Host**>:<**Port**>'

Arguments

Host – The host name where the myarmdaemon run on

Port – The port where the myarmdaemon is listening for incoming new TCP sink connections

Description
Indicates that the TCP sink tries to establish a new TCP connection to the myarmdaemon.

I.2.41 Log message No. 43: TCPSINK DISCONNECTED

Level
status

Message
Disconnected from myarmdaemon <**Reason**>

Arguments

Reason – A possible reason why the TCP sink disconnected from the myarmdaemon

Description
Indicates that the TCP connection to the myarmdaemon was closed.

I.2.42   Log message No. 44: CANNOT OPEN FILE

Level
> error

Message
> Cannot open file '<**File**>'. Reason: '<**Reason**>'

Arguments

> **File** – The name of the file which could not be opened

> **Reason** – Reason of the failure

Description
> The reported file could not be opened due to the reported reason.

I.2.43   Log message No. 45: CANNOT READ FROM FILE

Level
> error

Message
> Cannot read from file '<**File**>'. Reason: '<**Reason**>'

Arguments

> **File** – The file name where the error occurred

> **Reason** – The reason of the error

Description
> During reading of a file (ARM data or runtime configuration files) an error occurred.

I.2.44   Log message No. 46: FILESTORAGE STATE

Level
> status

Message
> State changed from [<**old**>] to [<**new**>]

Arguments

> **old** – Old file storage state

> **new** – New file storage state

Description
> The file storage module has changed its state from the old to the new state. A state change indicates
> that the processing of ARM data within the file storage stopped (e.g. the new state is INACTIVE) or
> is started again (e.g. the new state is SCANDIR or READNEXT).

I.2.45 Log message No. 47: FILESTORAGE CORRUPT FILE

Level
  error

Message
  File '<**Name**>' is corrupt

Arguments

  Name – File name which is corrupt

Description
  A corrupted file was detected.

I.2.46 Log message No. 48: FILESTORAGE FILE NOT PROCESSED

Level
  info

Message
  File '<**Name**>' not processed due to '<**Reason**>'

Arguments

  Name – Name of the file which was not processed

  Reason – The reason why it was not processed

Description
  The file storage component could not process the reported file.

I.2.47 Log message No. 49: FILESTORAGE FILE PROCESSED

Level
  info

Message
  Processed '<**Name**>' file. Defs='<**Defs**>', Apps='<**Apps**>', Trans='<**Trans**>', Total files='<**Files**>'

Arguments

  Name – Name of the processed file

  Defs – Number of definitions dropped (application-, transaction- or metric-definition)

  Apps – Number of application instances dropped

  Trans – Number of transaction instances dropped

  Files – Total number of files processed so far

Description
  Informational message about a processed ARM data file.

I.2.48   Log message No. 50: CANNOT OPEN DIRECTORY

Level

error

Message

Can't open directory <**Name**>'. Reason: '<**Reason**>'

Arguments

Name – Name of the directory which could not be opened

Reason – The reason why it was not opened

Description

The configured directory could not be opened for reading.

Action

Please check the configuration and make sure the process has neccessary permissions.

I.2.49   Log message No. 51: HANDLER CREATED

Level

status

Message

<**Name**> handler created

Arguments

Name – Name of the handler

Description

This message logs that a handler was created and is being used.

I.2.50   Log message No. 52: CHECK DIRECTORY FAILED

Level

error

Message

Directory check for '<**Dir**>' failed: <**Reason**>

Arguments

Dir – Path to the directory which was checked

Reason – The description why the check failed

Description

Some parts of MyARM check the existence and permission of the configured directories. The directory must exist and the user on behalf of which MyARM was executed needs write permission for that directory.

Action

- Please create the directory if it does not exist
- Please set the correct write permissions

I.2.51   Log message No. 53: FILESINK ARMDATA INFO

Brief

Reports a processed ARM data file

Level

info

Message

ARM data file: '<**File**>' contains <**Contents**>

Arguments

File – Name of the processed file

Contents – Number of ARM data entities stored in the ARM data file

Description

This message reports the successful processing of an ARM data file by the filesink. The number of contained entities are reported.

I.2.52   Log message No. 54: FILESINK CANNOT MOVE FILE

Level

error

Message

Cannot move file '<**File**>' to '<**Dir**>': Reason '<**Reason**>', dropped ARM data: <**Contents**>

Arguments

File – Name of the file which was not moved

Dir – Directory name where the file should be moved to

Reason – The reason of the error

Contents – Number of dropped ARM data entities stored in the ARM data file

Description

This message is issued when an ARM data file could not be moved to its destination directory.

Action

- Please check the existence and write permissions of the destination directory

I.2.53   Log message No. 55: FILESINK MIN FREE

Level

error

Message

Minimum free disk space reached '<**Free**><=<**Min**>', dropped ARM data: <**Contents**>

Arguments

Free – Current free disk space

Min – Configured minimum disk space

Contents – Number of dropped ARM data entities stored in the ARM data file

Description

    This message is issued when the disk space gets low and drops below the configured minimum disk space of the file datasink. Please check if the myarmdaemon is running and processes stored files. Also check other system resources which may waste disk space.

I.2.54   Log message No. 56: FILESINK MAX USED

Level

    error

Message

    Maximum used disk space reached '<**Used**>>=<**Max**>', dropped ARM data: <**Contents**>

Arguments

    Used – Currently used ARM data disk space

    Max – Configured maximum of used disk space

    Contents – Number of dropped ARM data entities stored in the ARM data file

Description

    This message is issued when the used disk space of ARM data exceeds its configured maximum limit. Please check if the myarmdaemon is running and processes stored files. Also check the destination datasink (e.g. the database or myarmdaemon) is up and running.

I.2.55   Log message No. 57: SOCKET TOO MANY CLIENTS

Level

    error

Message

    Server socket <**Host**>:<**Port**> maximum number of clients <**Max**> reached. Deny connect request from <**Client**>

Arguments

    Host – The host/interface where the myarmdaemon is listening on

    Port – The port where the myarmdaemon is listening for incoming new connections

    Max – The maximum number of allowed clients

    Client – The host/interface and port of the client

Description

    The reported server socket has reached its configured maximum number of clients and a new connection request was denied.

I.2.56   Log message No. 58: SOCKET ERROR

Level

    error

Message

    <**Message**>: Reason: '<**Reason**>'

Arguments

    Message – The detailed message of the socket error

Reason – The reason of the error

Description
   An error occurred within the TCP sockets component.

I.2.57   Log message No. 59: SOCKET LISTENING

Level
   status

Message
   <**Name**> socket listening <**State**>

Arguments

   Name – The name of the server socket

   State – 'enabled' or 'disabled' the listening socket

Description
   Indicates that the specified listening socket changed its state.

I.2.58   Log message No. 60: SOCKET ACCEPT

Level
   status

Message
   Accept new <**Name**> <**Client**>

Arguments

   Name – The listening socket name which has accepted a new client

   Client – The client socket and its associated connection id

Description
   Indicates that on the named listening socket a new client was accepted.

I.2.59   Log message No. 61: SOCKET CLOSED

Level
   status

Message
   Closed <**Name**> <**Client**>

Arguments

   Name – The listening socket name for which a client connection was closed

   Client – The closed client socket

Description
   Indicates the on the named listening socket a client connection was closed.

I.2.60   Log message No. 62: SOCKET DEAD

Level

    error

Message

    client is dead closing connection

Description

    Indicates that a client connection was detected to be dead and closed. This occurs when the myarm-daemon does not receive a keep-alive message from the client in the expected time interval.

I.2.61   Log message No. 63: DAEMON STARTED

Level

    status

Message

    myarmdaemon started

Description

    The myarmdaemon process is up and running.

I.2.62   Log message No. 64: DAEMON TERMINATING

Level

    status

Message

    Got '<**Cmd**>' terminating

Arguments

    Cmd – Stop command or a quit process signal

Description

    The myarmdaemon got a process quit signal or a stop command to terminate.

I.2.63   Log message No. 65: DAEMON TERMINATED

Level

    status

Message

    myarmdaemon terminated

Description

    The myarmdaemon process terminated.

I.2.64   Log message No. 66: DAEMON CHANGED MAX OPEN FILES

Level

    warning

Message

    Changed max open files from <**Old**> to <**New**>

Arguments

> **Old** – Old maximum number of open files
>
> **New** – New maximum number of open files

Description

> The configuration of the myarmdaemon exceeds the default maximum number of allowed open files and was changed to the new maximum number of open files.

Action

> To avoid this warning change the OS default maximum number of open files.

I.2.65   Log message No. 67: DAEMON TCP CLOSE CONNECTION

Level

> info

Message

> Closing connection due to '<**Reason**>'

Arguments

> **Reason** – Closing reason

Description

> A client connection was closed due to the given reason.

I.2.66   Log message No. 68: DAEMON TCP SINK INFO

Level

> info

Message

> Sink info received: armdata.buffer.size=<**BufSize**>, flow control=<**FlowCtrl**>, split armdata.buffer=<**SplitBuf**>, keepalive=<**KeepAlive**>

Arguments

> **BufSize** – The ARM data buffer size of the TCP sink client
>
> **FlowCtrl** – Boolean which indicates if the TCP sink supports flow control ('on') or not ('off')
>
> **SplitBuf** – Boolean which indicates if the TCP sink supports splitting of ARM data buffers
>
> **KeepAlive** – Keep alive interval in seconds of the TCP sink

Description

> Message which reports the TCP sink protocol information.

I.2.67   Log message No. 69: DAEMON TCP KEEPALIVE

Level

> info

Message

> Got keep alive message

Description

> A keep alive message was received from a client connection.

I.2.68   Log message No. 70: RUNTIME CONFIG ACTIVATED

Brief

Informs about activation of a new runtime configuration

Level

status

Message

Activated new runtime config file: '<**File**>'

Arguments

File – Name of the runtime config file

Description

This message logs a successful activation of a new runtime configuration. The new runtime configuration is used from now on.

I.2.69   Log message No. 71: RUNTIME CONFIG FAILED

Brief

Informs about a failed activation of a new runtime configuration

Level

error

Message

Failed to activate new runtime config file: '<**File**>'

Arguments

File – Name of the runtime config file

Description

This message logs a failed activation of a new runtime configuration. The old configuration will still be used.

I.2.70   Log message No. 72: RUNTIME CONFIG ENTRY CREATED

Brief

Informs about creation of a new runtime configuration entry

Level

info

Message

Runtime <**Type**> '<**Name**>' created

Arguments

Type – Type of runtime config entry

Name – Name of runtime config entry

Description

This message logs a successful creation of a new runtime configuration entry. The new entry is activated when the runtime configuration activated message is logged.

I.2.71   Log message No. 73: RUNTIME CONFIG ENTRY FAILED

Brief

The new runtime configuration entry could not be created

Level

error

Message

Failed to create runtime <**Type**> '<**Name**>'. Reason: '<**Reason**>'

Arguments

Type – Type of runtime config entry

Name – Name of runtime config entry

Reason – Reason of the failure

Description

This message logs a failure regarding the creation of a new runtime configuration entry.

I.2.72   Log message No. 74: RUNTIME CONFIG SAVED

Level

status

Message

Saved new runtime config to file '<**File**>'

Arguments

File – Name of the runtime config file which was saved

Description

This message logs that a new runtime configuration was saved to the reported file.

I.2.73   Log message No. 75: RUNTIME EVENT

Level

warning

Message

Runtime event triggered <**Event**>

Arguments

Event – Description of triggered runtime event

Description

This message logs that a new runtime event was triggered due to the current runtime configuration.

I.2.74   Log message No. 76: RUNTIME NOTIFICATION

Level

info

Message

Runtime notification executed '<**Notification**>' with status='<**Status**>

Arguments

> **Event** – Description of the executed runtime notification
>
> **Status** – Return status of the notification execution process (depends on the action executed)

Description
> This message logs that a new runtime notification was executed.

I.2.75 Log message No. 77: PROCESS STARTED

Level
> status

Message
> <**Name**> process started

Arguments

> **Name** – Name of the started process

Description
> This message logs that a new process was started.

I.2.76 Log message No. 78: PROCESS TERMINATED

Level
> status

Message
> <**Name**> process terminated with exit code <**Code**>

Arguments

> **Name** – Name of the terminated process
>
> **Code** – The exit code of the terminated process

Description
> This message logs that a process terminated and returned the given exit code.

I.2.77 Log message No. 79: DATABASE CONNECT FAILED

Level
> status

Message
> Cannot connect to '<**Database**>' database: <**Reason**>

Arguments

> **Database** – Name of the database
>
> **Reason** – The reason of the connection failure

Description
> Indicates that a connection to the configured database failed due to `Reason`.

I.2.78 Log message No. 82: CANNOT CREATE DIRECTORY

Level

error

Message

Cannot create directory '<**Dir**>'. Reason: '<**Reason**>'

Arguments

Dir – The name of the directory which could not be created

Reason – Reason of the failure

Description

The reported directory could not be created due to the reported reason.

I.2.79 Log message No. 83: CANNOT MOVE FILE

Level

error

Message

Cannot move file '<**File**>' to '<**DestFile**>': Reason '<**Reason**>'

Arguments

File – Name of the file which was not moved

DestFile – Destination file name (note needs to be on the same disk)

Reason – The reason of the error

Description

This message is issued when an archive file could not be moved to its destination directory.

Action

- Please check the existence and write permissions of the destination directory

I.2.80 Log message No. 84: ARCHIVE FILE INFO

Brief

Reports the contents overview of the stored MyARM data file

Level

info

Message

MyARM data file: '<**File**>'<**Contents**>

Arguments

File – Name of the processed file

Contents – If not empty shows an overview of the contents of the file

Description

This message reports the number of contained entities of the created MyARM data file.

I.2.81   Log message No. 85: INFO

Level
>info

Message
><Message>

Arguments

>**Message** – The information message

Description
>General purpose information message.

Appendix J

Third Party Licenses

J.1 Apache Software Foundation License

```
                          Apache License
                   Version 2.0, January 2004
                  http://www.apache.org/licenses/

    TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

    1. Definitions.

        "License" shall mean the terms and conditions for use, reproduction,
        and distribution as defined by Sections 1 through 9 of this document.

        "Licensor" shall mean the copyright owner or entity authorized by
        the copyright owner that is granting the License.

        "Legal Entity" shall mean the union of the acting entity and all
        other entities that control, are controlled by, or are under common
        control with that entity. For the purposes of this definition,
        "control" means (i) the power, direct or indirect, to cause the
        direction or management of such entity, whether by contract or
        otherwise, or (ii) ownership of fifty percent (50%) or more of the
        outstanding shares, or (iii) beneficial ownership of such entity.

        "You" (or "Your") shall mean an individual or Legal Entity
        exercising permissions granted by this License.

        "Source" form shall mean the preferred form for making modifications,
        including but not limited to software source code, documentation
        source, and configuration files.

        "Object" form shall mean any form resulting from mechanical
        transformation or translation of a Source form, including but
        not limited to compiled object code, generated documentation,
        and conversions to other media types.
```

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!) The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

     http://www.apache.org/licenses/LICENSE-2.0

```
   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
```

J.2   GNU Lesser General Public License

```
              GNU LESSER GENERAL PUBLIC LICENSE
                  Version 2.1, February 1999

 Copyright (C) 1991, 1999 Free Software Foundation, Inc.
 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL.  It also counts
 as the successor of the GNU Library Public License, version 2, hence
 the version number 2.1.]


                           Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

  This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it.  You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

  When we speak of free software, we are referring to freedom of use,
not price.  Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

  To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights.  These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

  For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you.  You must make sure that they, too, receive or can get the source
code.  If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
```

with the library after making changes to the library and recompiling
it. And you must show them these terms so they know their rights.

  We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

  To protect each distributor, we want to make it very clear that
there is no warranty for the free library. Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

  Finally, software patents pose a constant threat to the existence of
any free program. We wish to make sure that a company cannot
effectively restrict the users of a free program by obtaining a
restrictive license from a patent holder. Therefore, we insist that
any patent license obtained for a version of the library must be
consistent with the full freedom of use specified in this license.

  Most GNU software, including some libraries, is covered by the
ordinary GNU General Public License. This license, the GNU Lesser
General Public License, applies to certain designated libraries, and
is quite different from the ordinary General Public License. We use
this license for certain libraries in order to permit linking those
libraries into non-free programs.

  When a program is linked with a library, whether statically or using
a shared library, the combination of the two is legally speaking a
combined work, a derivative of the original library. The ordinary
General Public License therefore permits such linking only if the
entire combination fits its criteria of freedom. The Lesser General
Public License permits more lax criteria for linking other code with
the library.

  We call this license the "Lesser" General Public License because it
does Less to protect the user's freedom than the ordinary General
Public License. It also provides other free software developers Less
of an advantage over competing non-free programs. These disadvantages
are the reason we use the ordinary General Public License for many
libraries. However, the Lesser license provides advantages in certain
special circumstances.

  For example, on rare occasions, there may be a special need to
encourage the widest possible use of a certain library, so that it becomes
a de-facto standard. To achieve this, non-free programs must be
allowed to use the library. A more frequent case is that a free
library does the same job as widely used non-free libraries. In this
case, there is little to gain by limiting the free library to free
software only, so we use the Lesser General Public License.

  In other cases, permission to use a particular library in non-free

programs enables a greater number of people to use a large body of
free software. For example, permission to use the GNU C Library in
non-free programs enables many more people to use the whole GNU
operating system, as well as its variant, the GNU/Linux operating
system.

  Although the Lesser General Public License is Less protective of the
users' freedom, it does ensure that the user of a program that is
linked with the Library has the freedom and the wherewithal to run
that program using a modified version of the Library.

  The precise terms and conditions for copying, distribution and
modification follow. Pay close attention to the difference between a
"work based on the library" and a "work that uses the library". The
former contains code derived from the library, whereas the latter must
be combined with the library in order to run.

                    GNU LESSER GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License Agreement applies to any software library or other
program which contains a notice placed by the copyright holder or
other authorized party saying it may be distributed under the terms of
this Lesser General Public License (also called "this License").
Each licensee is addressed as "you".

  A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.

  The "Library", below, refers to any such software library or work
which has been distributed under these terms. A "work based on the
Library" means either the Library or any derivative work under
copyright law: that is to say, a work containing the Library or a
portion of it, either verbatim or with modifications and/or translated
straightforwardly into another language. (Hereinafter, translation is
included without limitation in the term "modification".)

  "Source code" for a work means the preferred form of the work for
making modifications to it. For a library, complete source code means
all the source code for all modules it contains, plus any associated
interface definition files, plus the scripts used to control compilation
and installation of the library.

  Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope. The act of
running a program using the Library is not restricted, and output from
such a program is covered only if its contents constitute a work based
on the Library (independent of the use of the Library in a tool for
writing it). Whether that is true depends on what the Library does
and what the program that uses the Library does.

  1. You may copy and distribute verbatim copies of the Library's

complete source code as you receive it, in any medium, provided that
you conspicuously and appropriately publish on each copy an
appropriate copyright notice and disclaimer of warranty; keep intact
all the notices that refer to this License and to the absence of any
warranty; and distribute a copy of this License along with the
Library.

   You may charge a fee for the physical act of transferring a copy,
and you may at your option offer warranty protection in exchange for a
fee.

   2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) The modified work must itself be a software library.

    b) You must cause the files modified to carry prominent notices
    stating that you changed the files and the date of any change.

    c) You must cause the whole of the work to be licensed at no
    charge to all third parties under the terms of this License.

    d) If a facility in the modified Library refers to a function or a
    table of data to be supplied by an application program that uses
    the facility, other than as an argument passed when the facility
    is invoked, then you must make a good faith effort to ensure that,
    in the event an application does not supply such function or
    table, the facility still operates, and performs whatever part of
    its purpose remains meaningful.

    (For example, a function in a library to compute square roots has
    a purpose that is entirely well-defined independent of the
    application. Therefore, Subsection 2d requires that any
    application-supplied function or table used by this function must
    be optional: if the application does not supply it, the square
    root function must still compute square roots.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or

collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

  3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

  Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

  This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

  4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

  If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

  5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

  However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

  When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The

threshold for this to be true is not precisely defined by law.

   If such an object file uses only numerical parameters, data
structure layouts and accessors, and small macros and small inline
functions (ten lines or less in length), then the use of the object
file is unrestricted, regardless of whether it is legally a derivative
work. (Executables containing this object code plus portions of the
Library will still fall under Section 6.)

   Otherwise, if the work is a derivative of the Library, you may
distribute the object code for the work under the terms of Section 6.
Any executables containing that work also fall under Section 6,
whether or not they are linked directly with the Library itself.

   6. As an exception to the Sections above, you may also combine or
link a "work that uses the Library" with the Library to produce a
work containing portions of the Library, and distribute that work
under terms of your choice, provided that the terms permit
modification of the work for the customer's own use and reverse
engineering for debugging such modifications.

   You must give prominent notice with each copy of the work that the
Library is used in it and that the Library and its use are covered by
this License. You must supply a copy of this License. If the work
during execution displays copyright notices, you must include the
copyright notice for the Library among them, as well as a reference
directing the user to the copy of this License. Also, you must do one
of these things:

   a) Accompany the work with the complete corresponding
   machine-readable source code for the Library including whatever
   changes were used in the work (which must be distributed under
   Sections 1 and 2 above); and, if the work is an executable linked
   with the Library, with the complete machine-readable "work that
   uses the Library", as object code and/or source code, so that the
   user can modify the Library and then relink to produce a modified
   executable containing the modified Library. (It is understood
   that the user who changes the contents of definitions files in the
   Library will not necessarily be able to recompile the application
   to use the modified definitions.)

   b) Use a suitable shared library mechanism for linking with the
   Library. A suitable mechanism is one that (1) uses at run time a
   copy of the library already present on the user's computer system,
   rather than copying library functions into the executable, and (2)
   will operate properly with a modified version of the library, if
   the user installs one, as long as the modified version is
   interface-compatible with the version that the work was made with.

   c) Accompany the work with a written offer, valid for at
   least three years, to give the same user the materials
   specified in Subsection 6a, above, for a charge no more
   than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy
from a designated place, offer equivalent access to copy the above
specified materials from the same place.

e) Verify that the user has already received a copy of these
materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the
Library" must include any data and utility programs needed for
reproducing the executable from it. However, as a special exception,
the materials to be distributed need not include anything that is
normally distributed (in either source or binary form) with the major
components (compiler, kernel, and so on) of the operating system on
which the executable runs, unless that component itself accompanies
the executable.

It may happen that this requirement contradicts the license
restrictions of other proprietary libraries that do not normally
accompany the operating system. Such a contradiction means you cannot
use both them and the Library together in an executable that you
distribute.

7. You may place library facilities that are a work based on the
Library side-by-side in a single library together with other library
facilities not covered by this License, and distribute such a combined
library, provided that the separate distribution of the work based on
the Library and of the other library facilities is otherwise
permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work
based on the Library, uncombined with any other library
facilities. This must be distributed under the terms of the
Sections above.

b) Give prominent notice with the combined library of the fact
that part of it is a work based on the Library, and explaining
where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute
the Library except as expressly provided under this License. Any
attempt otherwise to copy, modify, sublicense, link with, or
distribute the Library is void, and will automatically terminate your
rights under this License. However, parties who have received copies,
or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not
signed it. However, nothing else grants you permission to modify or
distribute the Library or its derivative works. These actions are
prohibited by law if you do not accept this License. Therefore, by
modifying or distributing the Library (or any work based on the
Library), you indicate your acceptance of this License to do so, and

all its terms and conditions for copying, distributing or modifying the Library or works based on it.

  10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

  11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

  12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

  13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version,

but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation. If the Library does not specify a
license version number, you may choose any version ever published by
the Free Software Foundation.

   14. If you wish to incorporate parts of the Library into other free
programs whose distribution conditions are incompatible with these,
write to the author to ask for permission. For software which is
copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this. Our
decision will be guided by the two goals of preserving the free status
of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

                            NO WARRANTY

   15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

   16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Libraries

   If you develop a new library, and you want it to be of the greatest
possible use to the public, we recommend making it free software that
everyone can redistribute and change. You can do so by permitting
redistribution under these terms (or, alternatively, under the terms of the
ordinary General Public License).

   To apply these terms, attach the following notices to the library. It is
safest to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least the
"copyright" line and a pointer to where the full notice is found.

    <one line to give the library's name and a brief idea of what it does.>
    Copyright (C) <year> <name of author>

    This library is free software; you can redistribute it and/or
    modify it under the terms of the GNU Lesser General Public
    License as published by the Free Software Foundation; either
    version 2.1 of the License, or (at your option) any later version.

    This library is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary. Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice

That's all there is to it!

List of Configurations

List of Figures

List of Example Output

Bibliography

[ARM2] THE OPEN GROUP: *Technical Standard: Systems Management: Application Response Measurement (ARM) API* , Berkshire, United Kingdom, 1998, ISBN: 1-85912-211-6

[ARM3] THE OPEN GROUP: *Technical Standard: Application Response Measurement, Issue 3.0 - Java Binding* , Berkshire, United Kingdom, 2001, UK ISBN: 185912 232 9, US ISBN: 1931624 04 6

[ARM4C] THE OPEN GROUP: *Technical Standard: Application Response Measurement (ARM) Issue 4.0, V2 - C Binding* , Berkshire, United Kingdom, 2004, ISBN: 1931624380 17, 23, 173

[ARM4J] THE OPEN GROUP: *Technical Standard: Application Response Measurement (ARM) Issue 4.0, V2 - Java Binding* , Berkshire, United Kingdom, 2004, ISBN: 1931624399 17

Index