

# Installation Guide

Installation Guide for Fabasoft app.telemetry 2015  
Update Rollup 3

***Fabasoft***<sup>®</sup>

Copyright ©

Fabasoft R&D GmbH, A-4020 Linz, 2015.

All rights reserved. All hardware and software names used are registered trade names and/or registered trademarks of the respective manufacturers.

These documents are highly confidential. No rights to our software or our professional services, or results of our professional services, or other protected rights can be based on the handing over and presentation of these documents.

Distribution, publication or duplication is not permitted.

# Contents

<b>1 Introduction</b>	<b>6</b>
<b>2 Prerequisites for Linux Systems</b>	<b>7</b>
2.1 Prerequisites for Fabasoft app.telemetry Agent	7
2.2 Prerequisites for Fabasoft app.telemetry Web Service	7
2.3 Prerequisites for other app.telemetry Modules	7
2.4 Additional prerequisites for Fabasoft app.telemetry Server	7
2.5 Configuration of SNMP	8
<b>3 Installation on Linux Systems</b>	<b>9</b>
3.1 Installation of Fabasoft app.telemetry Agent	9
3.1.1 Installation of Fabasoft app.telemetry Software-Telemetry Web Service	9
3.1.2 Installation of additional (optional) Fabasoft app.telemetry Modules	10
3.2 Installation of Fabasoft app.telemetry Server	10
3.2.1 Loading Fabasoft app.telemetry License	10
3.2.2 Configuration of Apache Web Server for Fabasoft app.telemetry	11
3.2.3 Enable Data compression	12
<b>4 Prerequisites for Microsoft Windows Systems</b>	<b>13</b>
4.1 Prerequisites for Fabasoft app.telemetry Agent	13
4.2 Prerequisites for Fabasoft app.telemetry Web Service	13
4.3 Prerequisites for Fabasoft app.telemetry Server	13
<b>5 Installation on Microsoft Windows Systems</b>	<b>14</b>
5.1 Installation of Fabasoft app.telemetry Agent	14
5.1.1 Fabasoft app.telemetry Agents on Microsoft Windows Cluster Nodes	15
5.1.2 Installation of Fabasoft app.telemetry Software-Telemetry Web Service	15
5.2 Installation of Fabasoft app.telemetry Server	18
5.2.1 Loading Fabasoft app.telemetry License	21
<b>6 Start and Use the Fabasoft app.telemetry Client</b>	<b>22</b>
6.1 Configuration of Infrastructure	23
6.1.1 Creating Agents	26
6.1.2 Detection of Applications	27
6.1.3 Creating Service Checks	27
6.1.4 Special Service Checks and their Parameters	28
6.2 Using Software-Telemetry Log Pools and Top X Reports	32
6.2.1 Top X Report / Drill-Down Analysis	34
6.2.2 Configure a Database for Software-Telemetry Logs	35
6.2.3 Filtered Log Pools	35
6.2.4 View on Log Pools	38

6.2.5 Syslog Log Pools	39
6.2.6 Microsoft Windows Eventlog Forwarder	39
6.3 Inbox for Incoming Feedbacks	40
6.4 Feedback Forms Designer	41
6.5 Configure Web Timing	43
6.6 Configure Web Timing with Feedback Dialog	44
6.7 Configure Notifications for Status Changes	45
6.7.1 Configure Notification Schedules	46
6.8 Configure Service Level Definitions	47
6.9 Using Dashboards and Charts	48
6.9.1 Chart and Data Source Types	48
6.9.2 General Chart Properties	54
6.9.3 Dashboard View	55
<b>7 Product Version Upgrades</b>	<b>56</b>
7.1 Upgrading to Version 2015	56
7.1.1 Special Notes for Upgrades on RHEL/CentOS 6 with SELinux	56
7.1.2 Upgrade Procedure on Linux Systems	56
7.1.3 Upgrade Procedure on Microsoft Windows Systems	57
7.1.4 Configuration changes for Fabasoft Folio when using Software-Telemetry Web Service	57
<b>8 Fabasoft app.telemetry Architecture</b>	<b>59</b>
8.1 Fabasoft app.telemetry Server	59
8.1.1 Fabasoft app.telemetry Server Service	59
8.1.2 Fabasoft app.telemetry Worker Service	59
8.1.3 Fabasoft app.telemetry Webservice	59
8.1.4 Fabasoft app.telemetry Ecomm Service	59
8.2 Fabasoft app.telemetry Agent	60
8.3 Fabasoft app.telemetry Client	60
8.4 Secure Communication	60
8.4.1 Certificate Management	60
8.4.2 Trusted Certificates	61
8.4.3 Failover/Standby configuration	61
<b>9 Appendix</b>	<b>61</b>
9.1 Installing Internet Information Services on Microsoft Windows Server 2008 / 2012	61
9.2 Enabling SSL after installation of Fabasoft app.telemetry Server	63
9.2.1 SSL Configuration on Microsoft Windows (IIS)	63
9.2.2 SSL Configuration on Linux (Apache)	64
9.3 Debugging and Logging	65

9.3.1 Microsoft Windows	65
9.3.2 Linux	65
9.3.3 Fabasoft app.telemetry Client - Logging	66
9.4 Special Configuration Parameters	66
9.4.1 Configuration of Listening Ports for Fabasoft app.telemetry Agent/Server	66
9.4.2 Configuration of Software-Telemetry Data Directory (Server)	67
9.4.3 Configuration of Software-Telemetry Session Export Directory (Server)	68
9.4.4 Configuration of Disk Cache and Memory Buffer for the Agent	68
9.4.5 Configuration of Database Rollforward Logs on the app.telemetry Server	70
9.5 Configure Sendmail to Forward E-mails	71
9.6 Fabasoft app.telemetry with SELinux	72
9.7 Using PostgreSQL Database for app.telemetry Server on Microsoft Windows	73
9.8 Configuration of PostgreSQL Database for app.telemetry Server on Linux	73
9.9 Installing Fabasoft app.telemetry on RHEL/CentOS 7	75
9.9.1 Start/Stop/Status of Daemons	75
9.9.2 Enable Autostart of Daemons	75
9.9.3 Customize Startup Parameters (Core Dump Settings)	75

# 1 Introduction

The Fabasoft app.telemetry consists of four software components:

- Fabasoft app.telemetry server – including the following services/daemons
  - Server
  - Worker
  - EComm
  - Web service
- Fabasoft app.telemetry agent
- Fabasoft app.telemetry web browser client (hosted as web service on app.telemetry server)
- Fabasoft app.telemetry Software-Telemetry web service (also known as *WebAPI*)

A Fabasoft app.telemetry installation is always made up of one central Fabasoft app.telemetry server (which can be installed either on Linux or Microsoft Windows) and many Fabasoft app.telemetry agents (one agent for every target server – could be Linux or Microsoft Windows).

Heterogeneous installations with Fabasoft app.telemetry agents on different platforms are supported.

The Fabasoft app.telemetry Software-Telemetry web service (also known as WebAPI) is required for End-2-End instrumented applications using the Fabasoft app.telemetry JavaScript SDK which will send the telemetry data from an instrumented web page to the web service which will forward the data to the app.telemetry agent. In such an environment the app.telemetry web service has to be installed on some agent platforms additionally to the agent package.

Fabasoft app.telemetry also provides some optional software components that might be installed as required for some special use cases:

- Fabasoft app.telemetry web server instrumentation modules (Apache, IIS)
  - mod\_telemetry (instrumentation module for Apache)
  - telemetry\_iis (instrumentation module for IIS)
- Fabasoft app.telemetry syslog forwarder (rsyslog event forwarder module)

## 2 Prerequisites for Linux Systems

Before starting the installation of Fabasoft app.telemetry, some additional software components are required and have to be installed.

Verify that none of the following software packages are missing on the target servers or install the missing RPMs from the distribution installation media (Linux install DVD).

### 2.1 Prerequisites for Fabasoft app.telemetry Agent

- `net-snmp-libs` (installed by default OS installation)
- `net-snmp` (optional – only required to monitor SNMP values)
  - requires: `lm_sensors`
- `curl`
- `libxml2`
- `libgcrypt`

### 2.2 Prerequisites for Fabasoft app.telemetry Web Service

- Web server (Apache)
  - `httpd`
  - `apr`
  - `apr-util`
- `curl`

### 2.3 Prerequisites for other app.telemetry Modules

The app.telemetry Apache module (`mod_telemetry`) requires the Apache web server and all of its requirements.

The app.telemetry syslog forwarder module is only available for RHEL/CentOS 6.x and 7.x and requires the rsyslog daemon (`rsyslog` >= 5.8.10).

### 2.4 Additional prerequisites for Fabasoft app.telemetry Server

**Note:** The Fabasoft app.telemetry server installation also requires fulfilling all Fabasoft app.telemetry agent prerequisites.

- Web server (Apache)
  - `httpd`
  - `apr`
  - `apr-util`
- `postgresql-libs`
- `net-snmp-utils` (optional – for easier testing of SNMP configuration)
- `curl`
- `minizip`

## 2.5 Configuration of SNMP

The SNMP protocol is required to query essential system information from Linux-based systems. So a correctly configured net-snmp is needed on every Linux-based Fabasoft app.telemetry agent.

Configure the SNMP read community with your desired secure passphrase. (The write-community is not required by Fabasoft app.telemetry software.)

The default configuration file is located in `/etc/snmp/snmpd.conf`

### Configuration (`/etc/snmp/snmpd.conf`)

```
# 1. possibility: by using global read-only community
rocommunity <passphrase>
# default: rocommunity public
# ----- OR -----
# 2. possibility: by using system views
com2sec notConfigUser default <passphrase>
# default: com2sec notConfigUser default public

group notConfigGroup v1 notConfigUser
group notConfigGroup v2c notConfigUser
view systemview included .1
view systemview excluded .1.3.6.1.2.1.6

access notConfigGroup "" any noauth exact systemview none none
```

Some SNMP versions have troubles traversing the complete SNMP tree and may crash (if for example file system mount points are unavailable). A solution is to exclude the following SNMP subtree: 1.3.6.1.2.1.6.

### Example

```
view systemview included .1
view systemview excluded .1.3.6.1.2.1.6
```

You should enable automatic startup of SNMP daemon on every server and restart the daemon to get the changes take effect.

### Autostart SNMPD Daemon

```
chkconfig --level 345 snmpd on
/etc/init.d/snmpd restart
```



## 3 Installation on Linux Systems

Install the Fabasoft app.telemetry packages from the product media using the `rpm`-command. The server (`apptelemetryserver-<version>.rpm`) has to be installed once in the infrastructure and the agent (`apptelemetryagent-<version>.rpm`) on every system (also on the server).

The Fabasoft app.telemetry server installation also requires a locally installed app.telemetry agent and Software-Telemetry web (`apptelemetryweb`) package.

Since version 2012 Fall Release Fabasoft app.telemetry supports SELinux on RHEL/CentOS 6 and with Fabasoft app.telemetry 2015 also RHEL/CentOS 7. For other platforms SELinux must be turned off in order to use Fabasoft app.telemetry.

**Note:** Installing Fabasoft app.telemetry on a RHEL/CentOS 6 or 7 system with enabled SELinux will also install the SELinux files for Fabasoft app.telemetry and load the SELinux policy on the target system. **Note:** Loading SELinux policies may take a while and will slow down the RPM installation process!

For more details about Fabasoft app.telemetry with SELinux see chapter 9.6 “Fabasoft app.telemetry with SELinux” in appendix.

Since Fabasoft app.telemetry 2015 we support the installation on a RHEL/CentOS 7 platform. The installation procedure is very similar than before. But the daemon startup is now handled by the **SystemD** tools. For more details on installing or using Fabasoft app.telemetry on RHEL/CentOS 7 see chapter 9.9 “Installing Fabasoft app.telemetry on RHEL/CentOS 7” in appendix.

### 3.1 Installation of Fabasoft app.telemetry Agent

#### Installation (Linux Shell)

```
[root@localhost]# rpm -ihv apptelemetryagent-<version>.<os>.x86_64.rpm
Preparing...             ##### [100%]
 2:apptelemetryagent     ##### [100%]
Starting apptelemetryagentd: [ OK ]
```

The daemon is started automatically after the installation.

#### 3.1.1 Installation of Fabasoft app.telemetry Software-Telemetry Web Service

Optionally you can install the Fabasoft app.telemetry Software-Telemetry web service (also known as WebAPI) additionally on some agent platforms. (This web service is required for End-2-End instrumented applications using the Fabasoft app.telemetry JavaScript SDK.)

The RPM package for the Fabasoft app.telemetry Software-Telemetry web service is located in the installation folder `TelemetryWeb` and is named `apptelemetryweb-<version>.<os>.x86_64.rpm`.

#### Installation (Linux Shell)

```
[root@localhost]# rpm -ihv apptelemetryweb-<version>.<os>.x86_64.rpm
Preparing...             ##### [100%]
 2:apptelemetryweb       ##### [100%]
```

### 3.1.2 Installation of additional (optional) Fabasoft app.telemetry Modules

Optionally you can install the Fabasoft app.telemetry Apache module (mod\_telemetry) from the installation folder `Telemetry-Modules` which provides instrumentation for general requests handled by the Apache web server. If the requests are not part of another application instrumentation you can configure a new Software-Telemetry log pool for the Apache web server (with the application filter “Apache”) including the specific log definition file `.../Telemetry-Modules/LogDefinitions/webserver-logdefinition-apache.xml`.

Another optional app.telemetry module is the syslog forwarder module required for capturing syslog events on behalf of a normal Software-Telemetry log pool. This feature requires the Linux syslog daemon `rsyslog` ( $\geq 5.8.10$ ) and is only available for RHEL/CentOS 6 or 7. In order to get syslog events as telemetry requests you have to create a new log pool with the specific application filter “Fabasoft app.telemetry” as application name and “Syslog Forwarder” as application tier name and import the specific log definition file `.../Telemetry-Modules/LogDefinitions/syslogforwarder-logdefinition.xml`.

## 3.2 Installation of Fabasoft app.telemetry Server

### Installation (Linux Shell)

```
[root@localhost]# rpm -ihv apptelemetryagent-<version>.<os>.x86_64.rpm
apptelemetryserver-<version>.<os>.x86_64.rpm apptelemetryweb-
<version>.<os>.x86_64.rpm

Preparing...                               ##### [100%]
 1:apptelemetryagent                       ##### [ 33%]
 2:apptelemetryweb                         ##### [ 67%]
 3:apptelemetryserver                      ##### [100%]

Run "/opt/app.telemetry/bin/serversetup.sh" to configure the
app.telemetry server the first time.
```

The daemons are started automatically after the installation.

Then call the setup script to configure the Fabasoft app.telemetry server for initial use and follow the instructions on screen. This script will set up the system as described in the following sub chapters automatically.

### Setup (Linux Shell)

```
/opt/app.telemetry/bin/serversetup.sh
```

**Note:** The configuration steps described in the following sub chapters are only informational and for special needs (a default configuration is already performed by the `serversetup.sh` script).

### 3.2.1 Loading Fabasoft app.telemetry License

Put the Fabasoft app.telemetry license on the Fabasoft app.telemetry server into the directory `/etc/app.telemetry` and save it with the name `license.lic`.

After a valid license is stored in `/etc/app.telemetry/license.lic` restart the Fabasoft app.telemetry server: `/etc/init.d/apptelemetryserverd restart`

### 3.2.2 Configuration of Apache Web Server for Fabasoft app.telemetry

This chapter describes the configuration steps of the Apache web server for the Fabasoft app.telemetry server for a manual installation. The `serversetup.sh` script will perform these basic configuration steps automatically with default settings.

#### 3.2.2.1 Starting Apache Web Server

The Fabasoft app.telemetry server requires the Apache web server to be installed, configured and running. Start the Apache web server (httpd) and set it up for automatic startup:

##### Autostart Apache Daemon

```
chkconfig --level 345 httpd on
/etc/init.d/httpd start      # or restart
```

#### 3.2.2.2 Configuration of Basic Authentication

Open the file `/etc/httpd/conf.d/apptelemetrywebserver.conf` and add these lines inside the `<Directory>`-tag:

##### Configuration (/etc/httpd/conf.d/apptelemetrywebserver.conf)

```
AuthType Basic
AuthName "app.telemetry Basic Authentication"
AuthUserFile /etc/app.telemetry/htpasswd
AuthGroupFile /etc/app.telemetry/htgroup
Require group apptelemetryadministrators apptelemetryusers
              apptelemetrydashboardusers apptelemetrylogpoolusers
              apptelemetrywebformusers
```

After adding these lines you must create an `htpasswd` file and specify valid user credentials by means of running the following command.

##### Setup User Credentials (Linux Shell)

```
htpasswd -c /etc/app.telemetry/htpasswd username
```

After finalizing these actions you have to restart the Apache web server (httpd).

#### 3.2.2.3 User/Group Permissions

The app.telemetry group privileges separate user permissions with a membership check of the login username against the following Apache groups:

- `apptelemetryadministrators`: full administrative access
- `apptelemetryusers`: read-only access to all data
- `apptelemetrydashboardusers`: only able to view dashboards assigned to this group and public-marked dashboards
- `apptelemetrylogpoolusers`: only able to view telemetry data and reports for log pools assigned to this group

- `apptelemetrywebformusers`: only able to view feedback inbox, forms and website configuration

#### Example of group file (`/etc/app.telemetry/htgroup`)

```
apptelemetryadministrators: user1 user2
apptelemetryusers: user5 user6 user7
apptelemetrydashboardusers: user8 user9
apptelemetrylogpoolusers: user10 user11
apptelemetrywebformusers: user12 user13
```

All those user accounts have to be created via the `htpasswd` command in the `htpasswd` file as mentioned above. Changes in the group membership require reloading the Apache configuration.

**Note:** User accounts with domain login (e.g. “APM\pool1”) have to be listed in the group file with escaped backslash (`\\`)

#### Example of group file with escaped domain-accounts (`/etc/app.telemetry/htgroup`)

```
apptelemetryadministrators: root admin
apptelemetrylogpoolusers: loggy APM\\pool1 APM\\pool2
limitedaccess: APM\\pool1 APM\\pool2
```

### 3.2.3 Enable Data compression

To enable data compression between Fabasoft `app.telemetry` web server and the Fabasoft `app.telemetry` client add the following lines to the apache configuration of the Fabasoft `app.telemetry` server (`/etc/httpd/conf.d/apptelemetrywebserver.conf`) inside the directory directive:

#### Configuration of Data Compression (`/etc/httpd/conf.d/apptelemetrywebserver.conf`)

```
<Directory "/opt/app.telemetry/web">
...
  <IfModule mod_deflate.c>
    SetOutputFilter DEFLATE
  </IfModule>
</Directory>
```

## 4 Prerequisites for Microsoft Windows Systems

Before starting the installation of Fabasoft app.telemetry, some additional software components are required and have to be installed.

Verify that none of the following software components are missing on the target servers or install the missing components from the distribution installation media (Microsoft Windows install CD/DVD).

### 4.1 Prerequisites for Fabasoft app.telemetry Agent

- Microsoft XML Core Services 6.0 (MSXML)
  - (required for web service availability checks only)

### 4.2 Prerequisites for Fabasoft app.telemetry Web Service

- Web server (Microsoft Internet Information Services – IIS)

### 4.3 Prerequisites for Fabasoft app.telemetry Server

- Microsoft XML Core Services 6.0 (MSXML)
- Web server (Microsoft Internet Information Services – IIS)
  - For details see the detailed chapter for the appropriate platform in the appendix

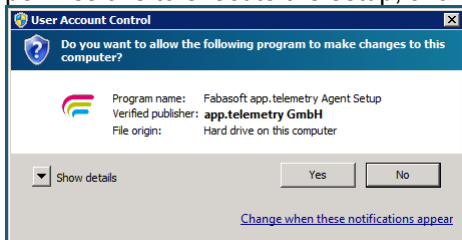
## 5 Installation on Microsoft Windows Systems

Install the Fabasoft app.telemetry packages (MSI setup packages) from the product media using the `setup.exe` files. The Fabasoft app.telemetry server has to be installed once in the infrastructure and the Fabasoft app.telemetry agent on every other system (not on the server where the agent is already included in the server package).

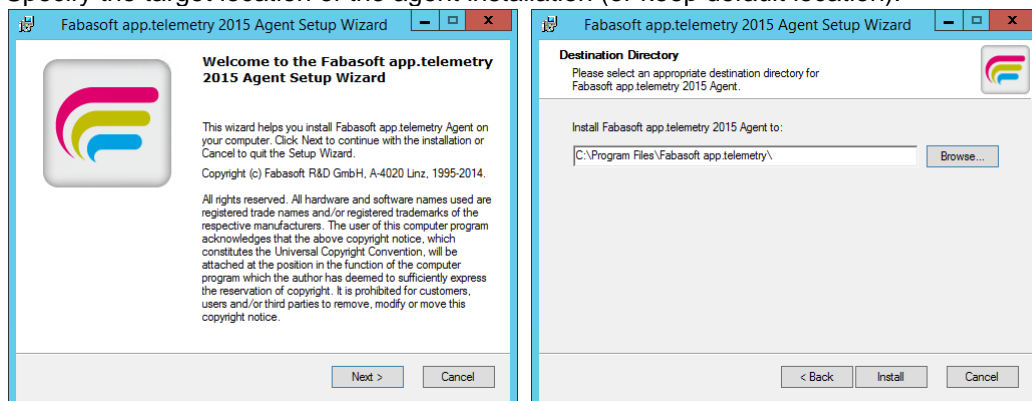
**Notice:** The `setup.exe` program may perform some upgrade tasks additionally to the MSI package, so it is important to start the setup by calling the `setup.exe` file of the appropriate installation directory.

### 5.1 Installation of Fabasoft app.telemetry Agent

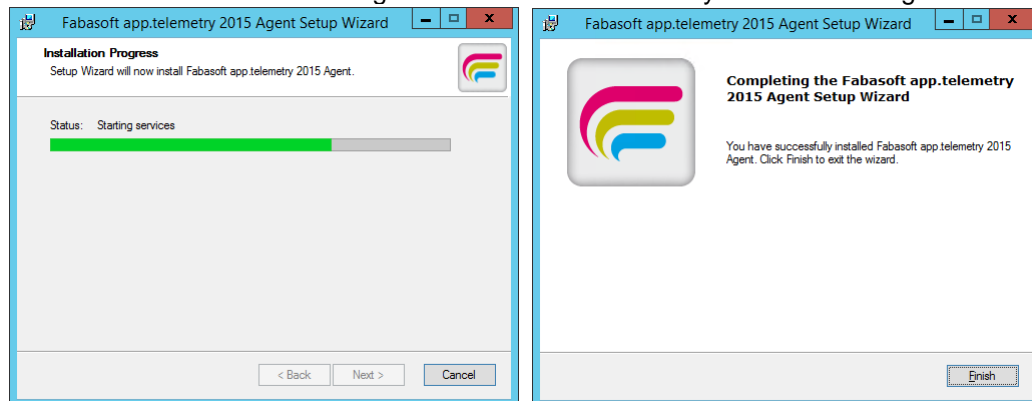
1. Use the setup wizard by invoking the `setup.exe` installation file from the “Agent” installation directory (for the appropriate windows architecture). If the “User Account Control” (UAC) asks for permissions to execute the setup, allow the setup execution with “Yes”.



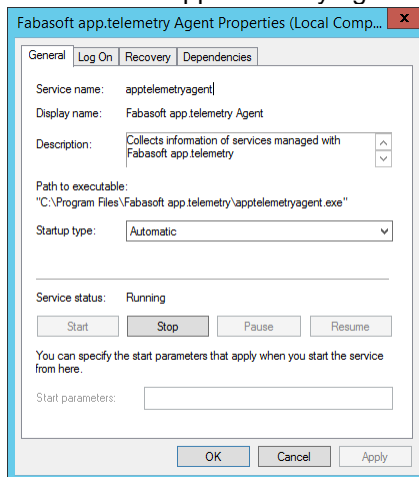
2. Specify the target location of the agent installation (or keep default location).



3. Finish the installation and the agent should work without any additional configuration.



4. The Fabasoft app.telemetry agent service is started automatically after the installation.



### 5.1.1 Fabasoft app.telemetry Agents on Microsoft Windows Cluster Nodes

The Fabasoft app.telemetry Agents enumerates the cluster groups on startup to provide the virtual hosts according to the cluster configuration. Therefore the cluster configuration has to be valid upon Fabasoft app.telemetry Agent startup and the Fabasoft app.telemetry Agent has to be restarted after changes to structure or IP addresses of cluster group have been made.

Since the cluster configuration under Microsoft Windows Clusters is provided by the “Cluster Service” process, it is necessary to establish a service dependency between “Fabasoft app.telemetry Agent” service and the “Cluster Service”. This can be done by the following command line:

```
sc config apptelemetryagent depend= ClusSvc
```

Since the cluster service will take some time until the cluster configuration will be available, the Fabasoft app.telemetry Agent will wait for the configuration to be available (implemented in Fabasoft app.telemetry 2013 Summer Release). The timeout for this startup delay is 30 seconds and may be configured setting the "ClusterStartupTimeout" DWORD registry key under "HKLM\SOFTWARE\Fabasoft app.telemetry\Agent" to the desired amount of seconds.

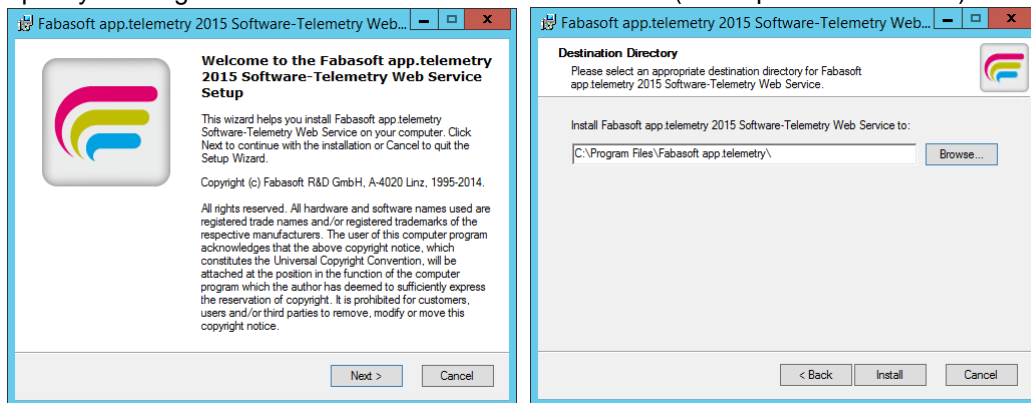
In Fabasoft app.telemetry Versions prior to 2013 Summer Release the Fabasoft app.telemetry Agent has to be restarted after cluster startup finished.

### 5.1.2 Installation of Fabasoft app.telemetry Software-Telemetry Web Service

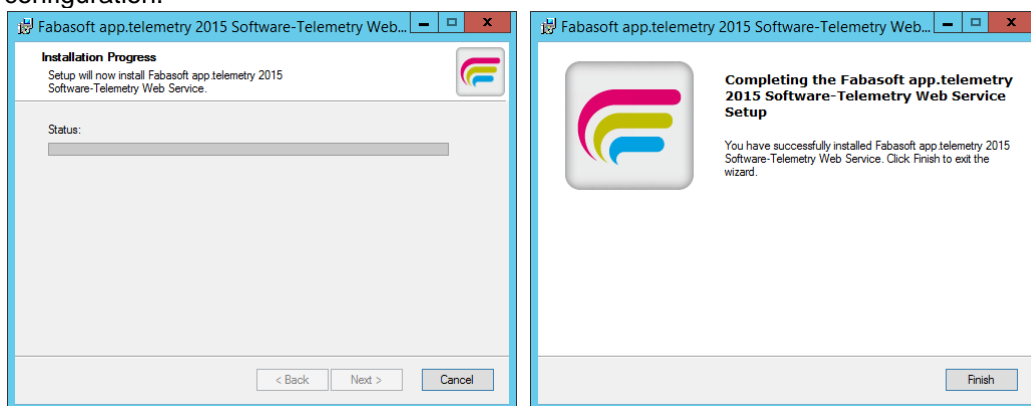
Optionally you can install the Fabasoft app.telemetry Software-Telemetry web service (also known as WebAPI) additionally on some agent platforms. (This web service is required for End-2-End instrumented applications using the Fabasoft app.telemetry JavaScript SDK.)

1. The installation package for the Fabasoft app.telemetry Software-Telemetry web service is located on the Fabasoft app.telemetry installation media in the folder “TelemetryWeb”. Invoke the setup wizard by starting the `setup.exe` installation file.

2. Specify the target location of the web service installation (or keep default location).

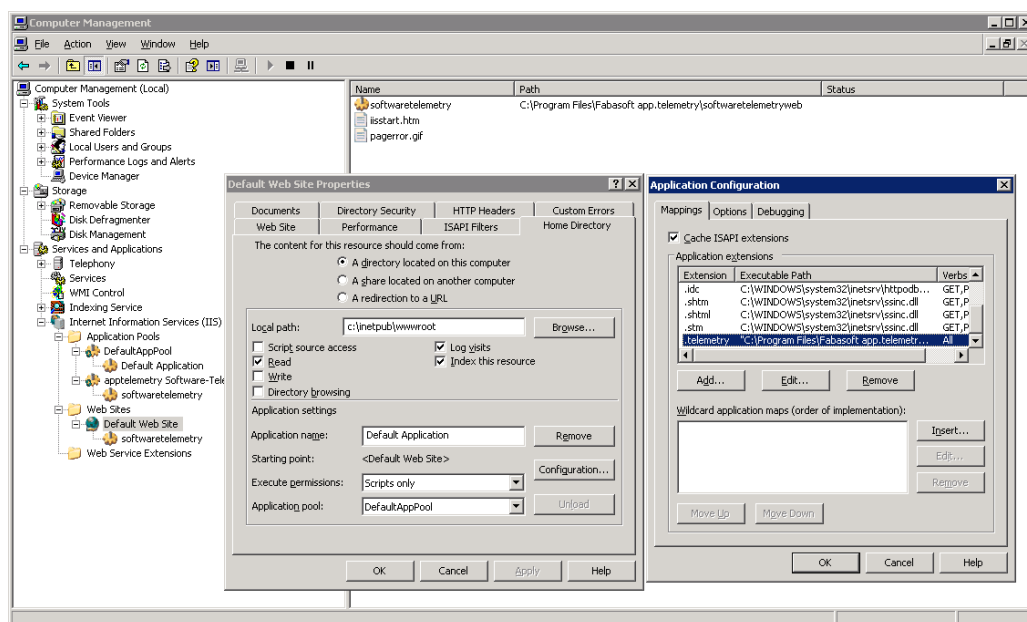


3. Finish the installation and the Software-Telemetry web service should run without any additional configuration.



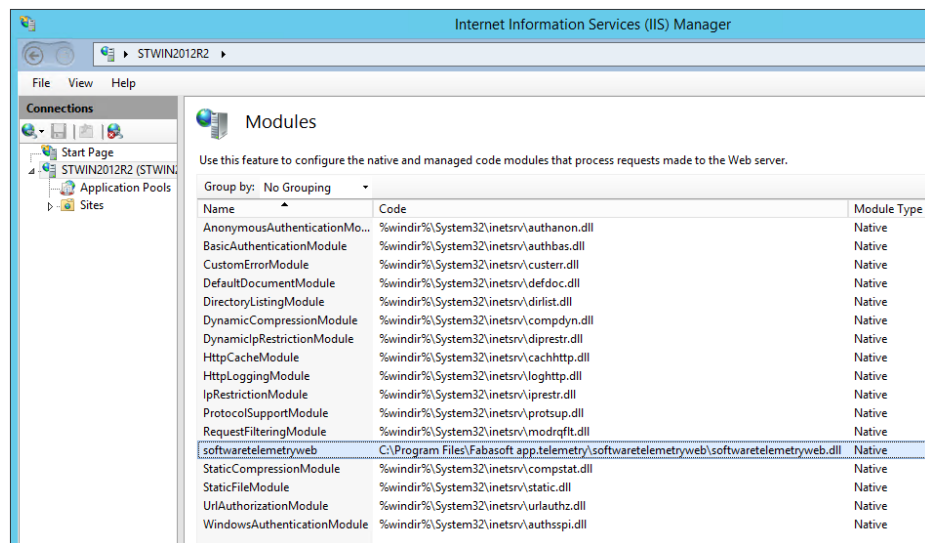
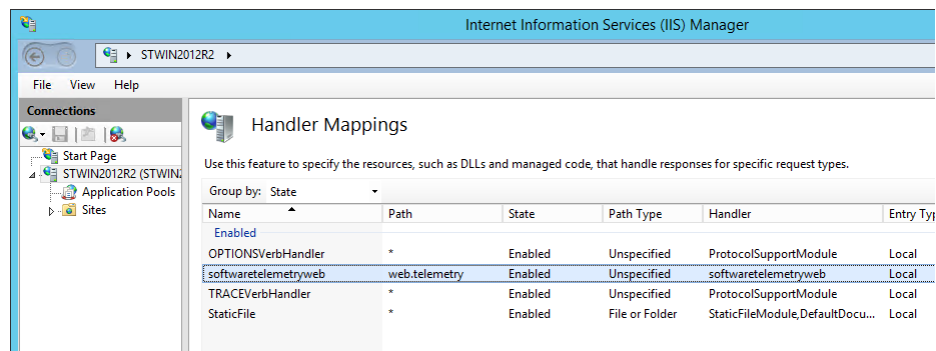
The Fabasoft app.telemetry Software-Telemetry web service is installed as global IIS handler catching any requests to URLs ending with the pattern “*web.telemetry*”.

On Microsoft Windows Server 2003 (Internet Information Services 6) the Software-Telemetry web service is installed as ISAPI extension with a configured application extension “*.telemetry*”.

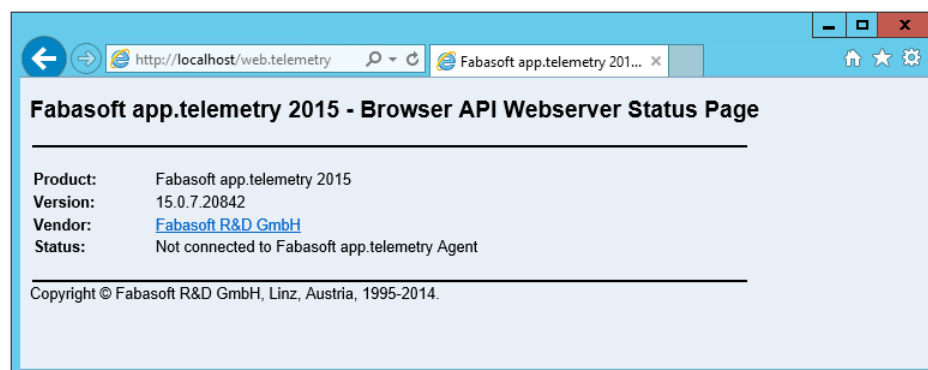




On Microsoft Windows Server 2008 or higher (Internet Information Services 7+) the Software-Telemetry web service is installed as native IIS module with a global handler mapping for “*web.telemetry*”.

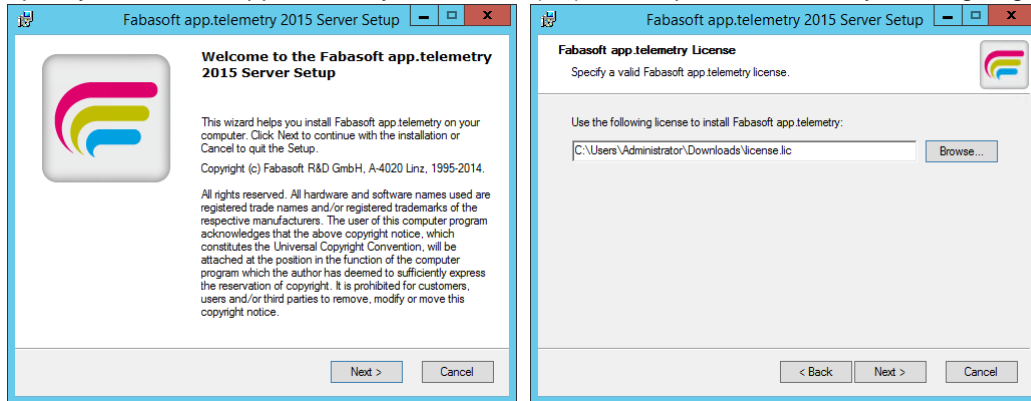


The correct installation of the Software-Telemetry web service can be verified by navigating with a web browser client to any URL ending with “*web.telemetry*” on this system. The result should be a status page.

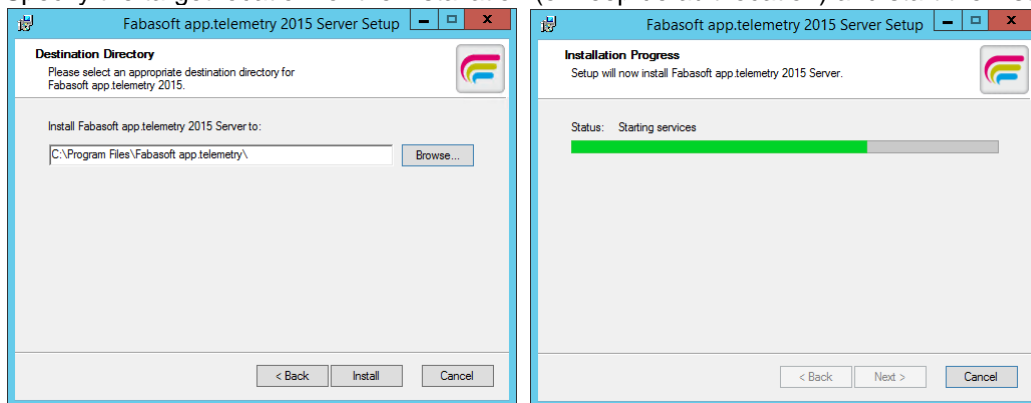


## 5.2 Installation of Fabasoft app.telemetry Server

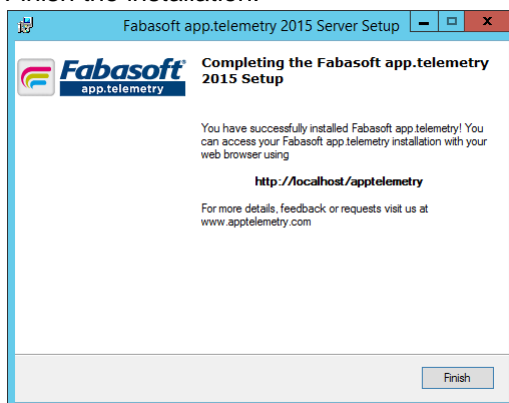
1. Use the setup wizard by invoking the setup.exe installation file from the “Server” installation directory (for the appropriate windows architecture).  
**Note:** The Fabasoft app.telemetry server package includes the app.telemetry agent and also includes the Software-Telemetry web service, so on the server platform you only have to install the server package!
2. Specify a Fabasoft app.telemetry license file (.lic) for the product version you are going to install.



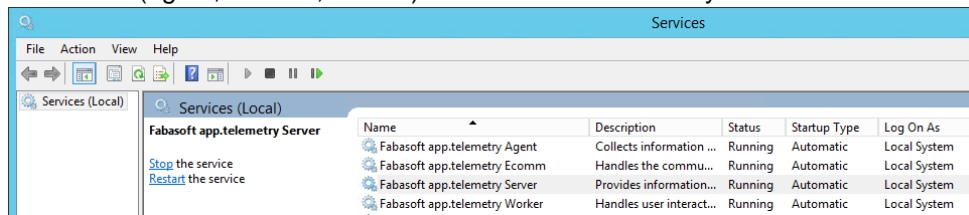
3. Specify the target location for the installation (or keep default location) and start the installation.



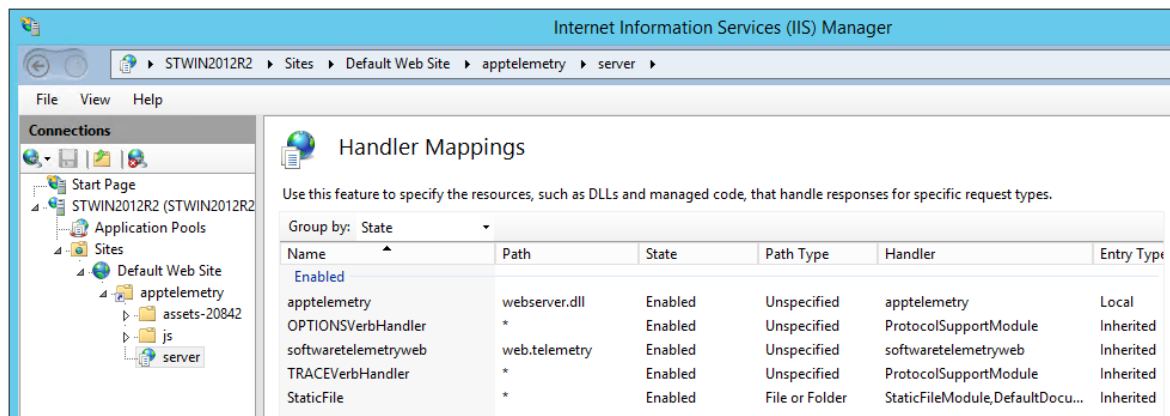
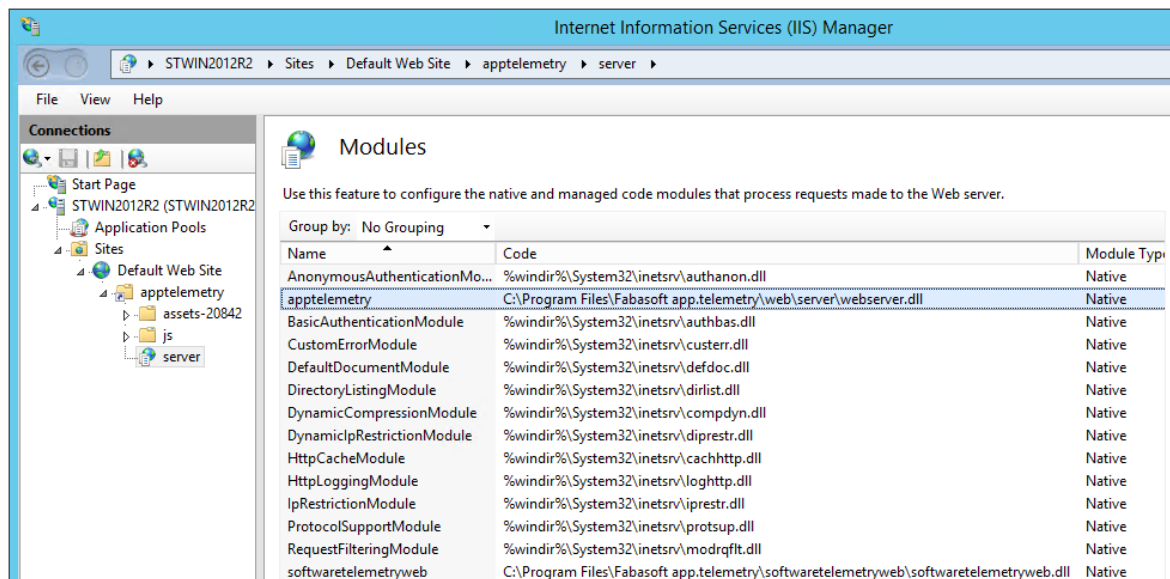
4. Finish the installation.



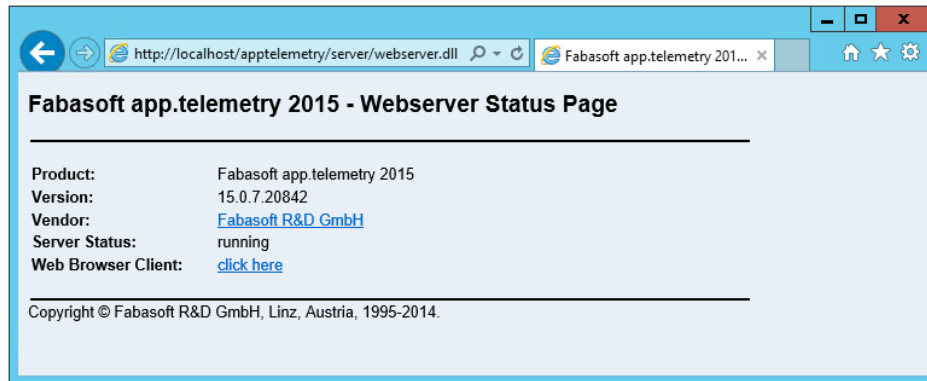
- The Fabasoft app.telemetry server service and all other services belonging to the server installation (agent, ecomm, worker) are started automatically after the installation.



The setup has created a new virtual directory “apptelemetry” in the Internet Information Services “Default Web Site” and also has registered the new “apptelemetry” native IIS module (on Microsoft Windows Server 2008 or higher)



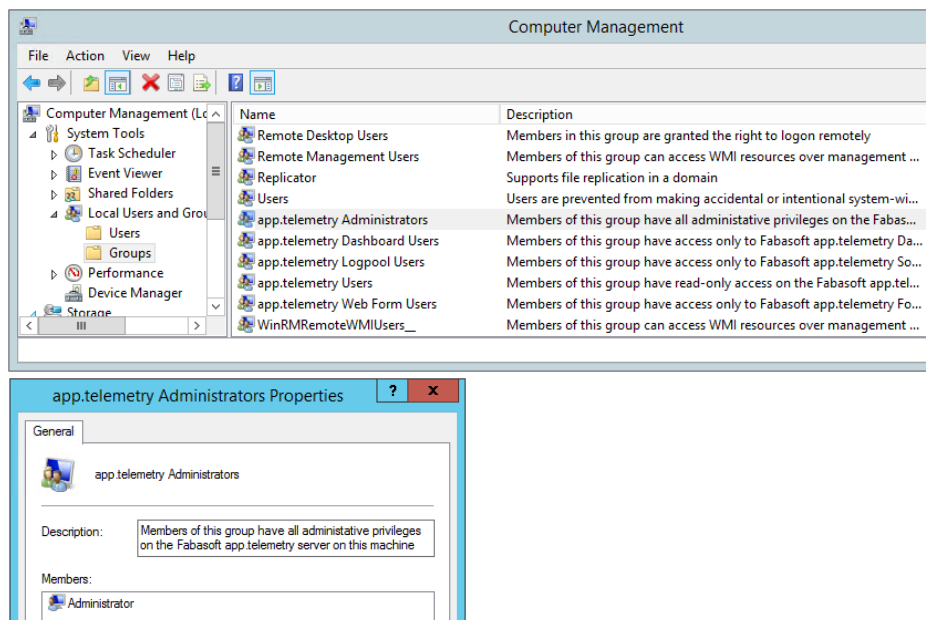
To test whether the Fabasoft app.telemetry web services is up and running, open the following URL in the web browser: <http://localhost/apptelemetry/server/webserver.dll>



Last but not least define the access permissions for client users to work with Fabasoft app.telemetry. All users that need to have access must be member of one of the following Microsoft Windows user groups:

- “app.telemetry Administrators”: full administrative access
- “app.telemetry Users”: read-only access to all data
- “app.telemetry Dashboard Users”: only able to view dashboards assigned to this group and public-marked dashboards
- “app.telemetry Logpool Users”: only able to view telemetry data and reports for log pools assigned to this group
- “app.telemetry Web Form Users”: only able to view feedback inbox, forms and website configuration

These groups are created by the setup process and the user account that has started the setup will be automatically added to the administrative group.



The default authentication methods for the web service are “*Integrated Windows authentication*” and “*Basic authentication*”. These settings can be changed in the “*Directory Security*” tab of the virtual directory “apptelemetry”.

### 5.2.1 Loading Fabasoft app.telemetry License

The Fabasoft app.telemetry server setup will ask for a license file only for a new installation and if no license file is present in the target directory.

If you want to update or install the license file later on, put the Fabasoft app.telemetry license on the Fabasoft app.telemetry server into the directory “C:\ProgramData\Fabasoft app.telemetry\” and save it with the name `license.lic`.

After a valid license is stored in “C:\ProgramData\Fabasoft app.telemetry\license.lic” restart the Fabasoft app.telemetry server with the Microsoft Windows Services manager.

## 6 Start and Use the Fabasoft app.telemetry Client

The Fabasoft app.telemetry web browser client will be automatically installed with the Fabasoft app.telemetry server setup in the virtual directory /apptelemetry/.

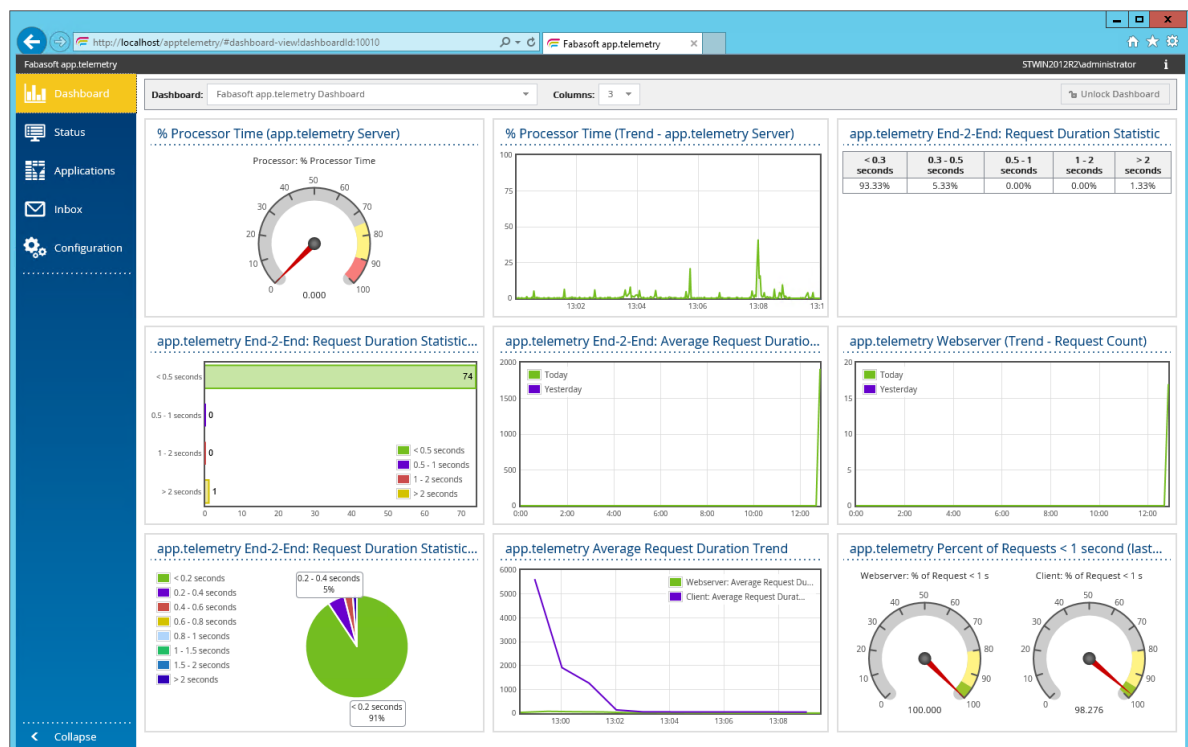
**You can access Fabasoft app.telemetry with your web browser:**  
<http://<app.telemetry Server>/apptelemetry/index.html>

The URL for the Fabasoft app.telemetry web browser client has the following format:

- For normal HTTP-connection: <http://<server-IP>/apptelemetry/index.html>
- For secure SSL-connection: <https://<server-IP>/apptelemetry/index.html>

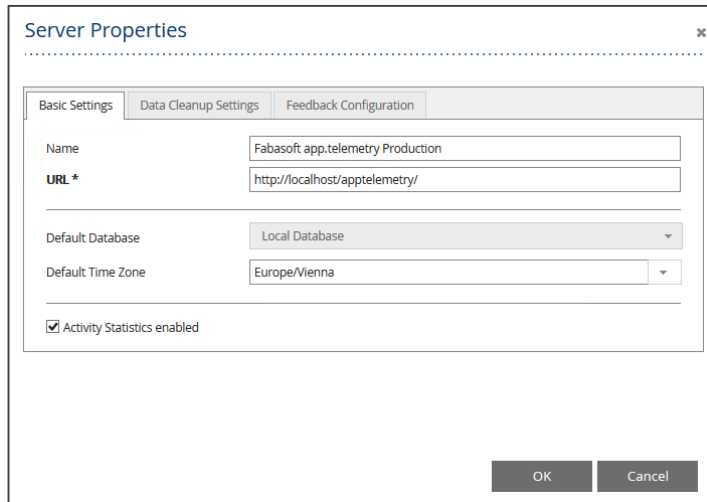
The virtual directory is by default “apptelemetry” as set up by the installer.

After these settings have been applied a startup infrastructure monitoring the Fabasoft app.telemetry server itself is loaded.



## 6.1 Configuration of Infrastructure

To configure applications, services, service checks and other parameters you have to switch the client to the infrastructure “Configuration” page by using the “*Configuration*” button from the view menu.



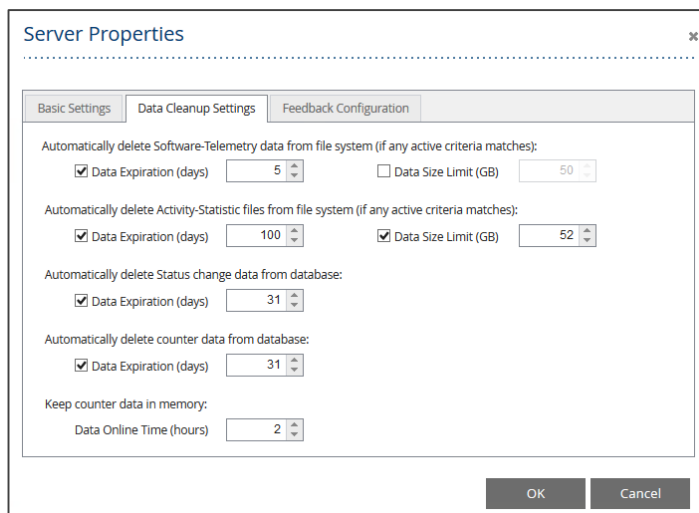
The screenshot shows the 'Server Properties' dialog box with the 'Basic Settings' tab selected. The fields are as follows:

Field	Value
Name	Fabasoft app.telemetry Production
URL *	http://localhost/apptelemetry/
Default Database	Local Database
Default Time Zone	Europe/Vienna
Activity Statistics enabled	<input checked="" type="checkbox"/>

At the bottom right are 'OK' and 'Cancel' buttons.

First of all you should configure some basic server properties by means of opening the edit dialog for the “Server Settings” object:

- The “**Name**” will be used as window title in the web browser and can be used to separate different app.telemetry installations more easily.
- The “**URL**” should include the full qualified host name of the app.telemetry server by which the installation can be reached from every client, because this URL is also used for some direct links to the installation (e.g. in notification e-mails).
- The “**Default Database**” is used everywhere where no specific database can be selected (e.g. form feedbacks, counter state history)
- The “**Default Timezone**” is used for this installation as base time zone.
- The “**Activity Statistics enabled**” checkbox activates the Inspect Activity Statistics feature.



The screenshot shows the 'Server Properties' dialog box with the 'Data Cleanup Settings' tab selected. The settings are as follows:

Category	Setting	Value
Automatically delete Software-Telemetry data from file system (if any active criteria matches):	<input checked="" type="checkbox"/> Data Expiration (days)	5
	<input type="checkbox"/> Data Size Limit (GB)	50
Automatically delete Activity-Statistic files from file system (if any active criteria matches):	<input checked="" type="checkbox"/> Data Expiration (days)	100
	<input checked="" type="checkbox"/> Data Size Limit (GB)	52
Automatically delete Status change data from database:	<input checked="" type="checkbox"/> Data Expiration (days)	31
Automatically delete counter data from database:	<input checked="" type="checkbox"/> Data Expiration (days)	31
Keep counter data in memory:	Data Online Time (hours)	2

At the bottom right are 'OK' and 'Cancel' buttons.

The Data Cleanup Settings help you to manage the data size of the app.telemetry Server.

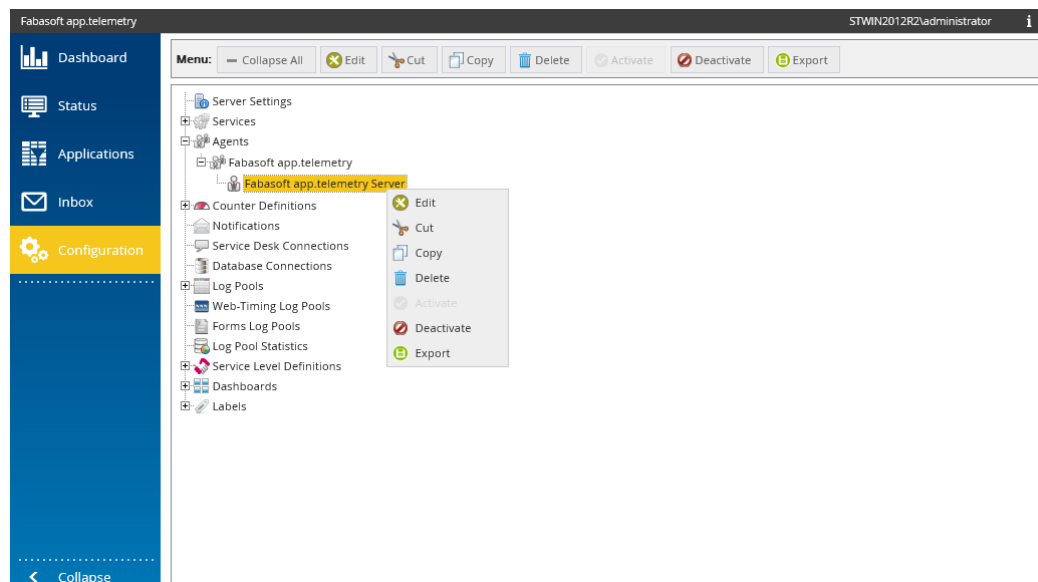
Software-Telemetry data is persisted on disk. Make sure to set **“Data Expiration (days)”** appropriate to allow users to analyze their requests and to limit the size of the data to fit the disk volume. On limited volumes you may even provide a **“Data Size Limit (GB)”** to remove data even faster e.g. in heavy load situations.

Activity Statistics are aggregated on disk and are intended to be kept longer than raw telemetry data to provide long term load statistics of your systems. These data are independent of the Software-Telemetry data. You may limit the available data pool by either setting a **“Data Expiration (days)”** value or a **“Data Size Limit (GB)”**.

Fabasoft app.telemetry automatically records a history of status changes of all counters on the default database. Set the **“Data Expiration (days)”** property to limit the amount of data stored.

With each counter check you can configure to record data in a database. Set the **“Data Expiration (days)”** property to automatically delete old status values. This is a global setting, which is applied to all counters.

Counter data is being kept in memory of the Fabasoft app.telemetry Server process to provide faster response drawing your counter charts. Limit the amount of data using the **“Data Online Timeout (hours)”** property to the duration required. To implement charts covering longer intervals (> 6 hours) persist the counter data on database and the database will provide the values.



Generally the infrastructure objects support the following actions to be accessed on any selected infrastructure object via a context menu which opens on right mouse click:

- **Edit:** all elements (except the root elements) can be edited in element-specific edit dialogs opened after this action is clicked.
- **New ...** Create a new object below the current selection.
- **Cut:** the currently selected object and store it in the clipboard.
- **Copy:** Store the currently selected object in the clipboard.
- **Paste:** Paste the object from the clipboard below the current selection (as child object).
- **Delete:** Delete the currently selected object and all children.  
**Note:** any object depending on the current to be deleted object will be also deleted (a warning message will list all the affected objects)

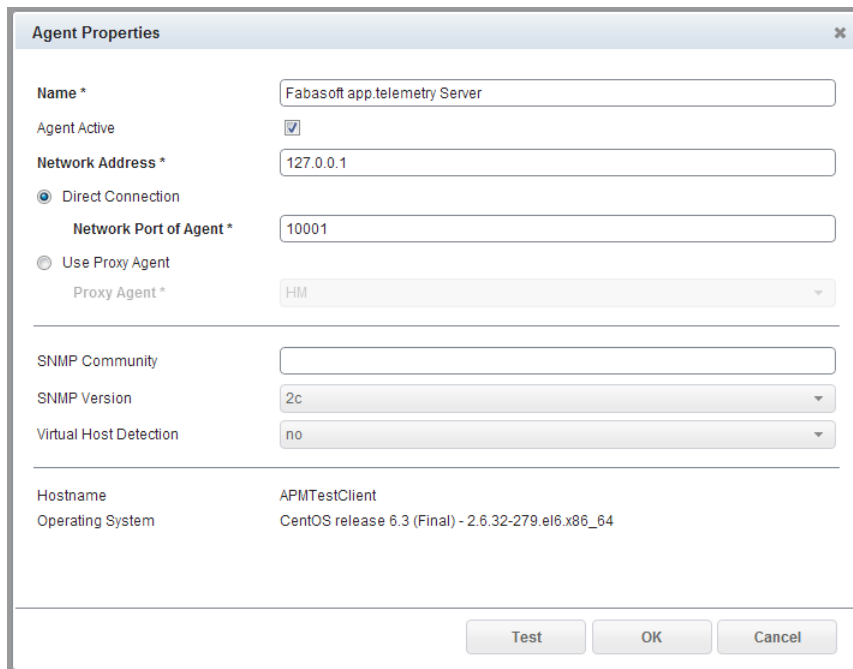


**Note:** With Winter Release 2012 a multi-selection was introduced which makes the configuration much easier. Just keep the CTRL-key pressed during selection of elements to select more than one element of the same type (in the same container – with same parent). You can “*cut*”, “*copy*”, “*paste*” and “*delete*” multiple elements (do not release the CTRL-key before pressing the action button).

### 6.1.1 Creating Agents

The first step is to add the Fabasoft app.telemetry agents (that you should have already installed in your infrastructure) by means of opening the “Agents” root group in the tree on the left side and adding agent groups and new agents with the “New” actions from context menu.

**Note:** Do not create two agent objects for the same physical server. The second app.telemetry agent opening a connection to the same agent will be marked as disconnected, because another agent is already connected.



The image shows a dialog box titled "Agent Properties" with a close button (X) in the top right corner. The dialog contains several fields and options for configuring an agent. The "Name" field is labeled "Name \*" and contains the text "Fabasoft app.telemetry Server". The "Agent Active" checkbox is checked. The "Network Address" field is labeled "Network Address \*" and contains "127.0.0.1". There are two radio buttons: "Direct Connection" (selected) and "Use Proxy Agent". Below "Direct Connection" is a "Network Port of Agent \*" field containing "10001". Below "Use Proxy Agent" is a "Proxy Agent \*" dropdown menu showing "HMI". Below these are three more fields: "SNMP Community" (empty), "SNMP Version" (dropdown showing "2c"), and "Virtual Host Detection" (dropdown showing "no"). At the bottom, there are two read-only fields: "Hostname" with value "APMTestClient" and "Operating System" with value "CentOS release 6.3 (Final) - 2.6.32-279.el6.x86\_64". At the very bottom are three buttons: "Test", "OK", and "Cancel".

#### 6.1.1.1 Set up SNMP Community for the Agent

Fabasoft app.telemetry agents that need to query counters using the net-snmp protocol use the specified “SNMP community” to connect to the SNMP daemon. If the SNMP community is not set, the default community “public” will be used. If the SNMP community is not set correctly for your SNMP system, timeouts may degrade the performance of the Fabasoft app.telemetry agent.

#### 6.1.1.2 Using Fabasoft app.telemetry Proxy Agent

Some situations require monitoring remote systems that do not have a Fabasoft app.telemetry agent installed. For such situations another Fabasoft app.telemetry agent can be used as proxy agent to perform remote calls using the net-snmp protocol.

In order to use this feature enter the remote network address of the target system that should be monitored and select another agent as “Proxy agent” from the combo box. Then enter the “SNMP community” to be used for the remote SNMP queries.

**“Virtual host detection”:** ... to detect on which server a virtual machine (VM) is running

In order to use the “virtual host detection” feature you have to configure a proxy agent for the virtual host server (VMware ESX Server, Microsoft Hyper-V) that is able to query the required fields (SNMP counters) from that server defining the remote IP address of that virtual host server.

**Note:** If you turn on the virtual host detection, be sure that the SNMP service on the target virtual host server (VMware ESX) is turned on and configured correctly, otherwise timeouts may degrade the performance of the Fabasoft app.telemetry agent.

### 6.1.2 Detection of Applications

Applications that are instrumented with the Fabasoft app.telemetry SDK and run on any system in your infrastructure where also an installed and configured app.telemetry agent is running will register themselves on the Fabasoft app.telemetry server and appear in the internal service group „New Services” in the infrastructure.

These service groups and services can be moved to the desired point in the infrastructure by means of cut and paste (via context menu actions in edit view).

### 6.1.3 Creating Service Checks

Every service should have at least one defined service check which is responsible for the status of this service. Otherwise the state will stay at the value “OK”.

For creating a new service check you have to select the service for which the check should be connected and use the “New Service Check” context menu action. A new dialog for selecting the desired service check type will be displayed.

After selecting the service check type a type-specific dialog for defining the check properties will be opened.

Enter the counter definition details (specific for that type) and switch to the check properties tab to finish the definition of that service check.

Optionally service checks for counter types can also be defined with limits for warning and critical state.

#### 6.1.3.1 Syntax for Counter Limits

If the limit field is kept empty, the result state of this counter will always be OK returning the counter value.

If warning and critical limits are defined the service check state will change to that state if the defined limits match – that is if an “above” limit is set and the current value exceeds that limit.

Examples:

- **Critical Limit:** Below: 200 ... if the value is 170 than the resulting state is critical
- **Warning Limit:** Above: 150 ... if the value is 170 than the resulting state is warning

- **Warning Limits:** Above: 50 and Below: 180  
+ **Critical Limit:** Above: 200 ... if the value is 170 then the resulting state is warning (but not critical)

### 6.1.3.2 Using a Limit Duration

If you don't want to get notified of each counter check value exceeding the limit, but only for those that stay longer than a given time period out of the defined limit, you have to define a duration value in seconds.

Duration is the value in seconds after which the state changes (if value exceeds the limit all this time)

Example for "% Processor CPU Usage"

- **Warning Limit:** Above: 80 with `Duration[sec]: 15` ... warn if value is higher than 80 for more than 15 seconds. (requires a check interval smaller than 15 sec)

### 6.1.4 Special Service Checks and their Parameters

#### 6.1.4.1 Counter Checks Using Existing Counter Definitions

Using a preexisting counter definition for a new service check is the easiest way to create one. Only select the counter from the list.

The counter definitions can be modified and extended via the root group "*Counter Definitions*".

#### 6.1.4.2 Counter Checks Using Formula

Define a formula with references to protocol-specific counters in the following format:  
{<protocol>:<targetvalue>}

For example:

- {snmp:.1.3.6.1.2.1.2.2.1.16.2}
- {winperf:\System\Threads}

Mathematical expressions with brackets () and operators (+, -, \*, /) can be used.

The following mathematical functions are also available for converting values:

- `delta()` ... calculates the difference between the last two values of a counter
- `delta32()` ... calculates the difference between the last two values of a counter and permits counter overflow on 32-bit raw-data
- `delta64()` ... calculates the difference between the last two values of a counter and permits counter overflow on 64-bit raw-data
- `deltaPerSecond()` ... calculates the average per second over the last check interval
- `delta32PerSecond()` ... calculates the average per second over the last check interval and permits counter overflow on 32-bit raw-data
- `delta64PerSecond()` ... calculates the average per second over the last check interval and permits counter overflow on 64-bit raw-data

#### 6.1.4.3 Linux /proc Counter Checks

Select the processor or the process and the desired counter to monitor.

These values are fetched from the Linux `/proc` file system.

#### 6.1.4.4 Linux Process Availability Checks

Select a desired Linux process from the list of currently running processes (on the remote Fabasoft app.telemetry agent).

It is monitored if this process is running.

#### 6.1.4.5 SNMP Counter Checks

Select an SNMP table, instance and counter from the lists to monitor.

The list contents are fetched from the remote Fabasoft app.telemetry agent.

Since 2013 Winter Release SNMP counters obtained from an SNMP table instance are stored in the app.telemetry infrastructure with a special syntax to keep the selected named instance also after possible table index changes: `<FULL-OID>["<named instance>",<table-depth of index column>, <OID of table-name column>]`

Example for SNMP Counter: `IF-MIB::ifTable > IF-MIB::ifInOctets > eth0`

- FULL-OID (of desired counter): `.1.3.6.1.2.1.2.2.1.10.2`
- Named instance: `"eth0"`
- Table-depth/length of index column: `1`
- OID of table-name column: `.1.3.6.1.2.1.2.2.1.2`
- ... this results in the following special counter definition:  
`.1.3.6.1.2.1.2.2.1.10.2["eth0",1,.1.3.6.1.2.1.2.2.1.2]`

**Note:** You can use these special OIDs also in counter checks using formulas or counter definitions to be independent from table index changes in SNMP tables. In order to get this special OID for your desired SNMP counter, just create a new SNMP counter, select the table, instance and counter from the list boxes – save the counter check – open it again by means of using the “edit” button and look up the SNMP OID field.

#### 6.1.4.6 Web Service Availability Checks

To define an HTTP check fill in all required fields of the counter definition details tab:

- **Target URL:** `http://.../`
- **Authentication settings:**
  - Anonymous Authentication
  - Kerberos/Integrated Authentication
    - On Microsoft Windows the service account is used for authentication (requires the agent service to be run under a domain account).  
**Note:** When changing the service account of the agent service, ensure that this account has enough permissions for other checks (e.g. for Windows Performance Counters – be member of “Performance Monitor Users” group) and to access all working directories used by the agent service (e.g. file counter checks).
  - Basic Authentication: with user name and password
  - Certificate (for Linux-based app.telemetry agents):
    - Define the certificate filename (`client1.pem`)
    - Certificate has to be deployed manually to the target agent into directory:  
`/etc/app.telemetry/`
    - The p12-certificate has to be converted into **pem-Format**:  
Certificate conversion: `openssl pkcs12 -in user.p12 -clcerts -out user.pem`

- Certificate (for Windows-based app.telemetry agents)
  - Define the certificate CN name (`clientcert`)
  - The certificate to be used has to be installed in the personal certificate store of the service user account of the Fabasoft app.telemetry agent. So if you want to define a HTTP Check using certificates, you have to change the service account of the agent service to a user account (the local system account has no personal certificate store associated) and import the certificate into the personal certificate store of that user.  
**Note:** Be careful when changing the service account of the agent service – for details see previous note.
  - To select a certificate out of multiple installed certificates, specify the CN of the certificate in the certificate parameter. Apart from that situation the certificate parameter is optional.
- Optional using an *HTTP-Proxy*
- *Display Result Value:* you can display different result values in the service check message column
- *Check Retry Count:* default = 3 ... this is the HTTP-retry count performed if the check fails (without any delay)
- *Check Server Certificate:* When checked, a SSL certificate will be validated and the check will fail if the certificate is not valid. Due to a bug in the underlying libcurl library each request may leak memory on the agent process performing web service checks with this option activated (LINUX only).

#### 6.1.4.7 Microsoft Windows Performance Counter Checks

Select a Microsoft Windows performance counter from the dynamically filled lists for object, instance and counter.

#### 6.1.4.8 Microsoft Windows Service Availability Checks

Select the desired Microsoft Windows process from the list of currently running processes (on the remote Fabasoft app.telemetry agent).

It is monitored if this service is in running state.

#### 6.1.4.9 Microsoft Windows Cluster Resource Availability Checks

If the Fabasoft app.telemetry agent is running on a Microsoft Windows cluster node you can select one of the dynamically filled Microsoft Windows cluster resources to monitor.

#### 6.1.4.10 Red Hat Cluster Suite Service Availability Checks

If the Fabasoft app.telemetry agent is running on a Red Hat cluster node you can select one of the dynamically filled Red Hat cluster suite services to monitor.

#### 6.1.4.11 Log Pool Statistics Counter Check

Select a specific log pool and get information about the performance of the requests inside the log pool.

You can limit the time range of requests regarded for this check calculation (in minutes) or use all available data from online log pool (in memory).

Decide whether to calculate the average request duration or the percentage of requests faster than <x> seconds.

Since Fabasoft app.telemetry 2012 Summer Release you can exclude requests from this calculation with a filter defining the maximum request duration (in seconds). All requests that last longer as this defined limit will not be regarded for this check calculation.

Since Fabasoft app.telemetry 2014 Fall Release a message “No Requests” is being displayed instead of no value when no requests have been processed for the given interval.

#### 6.1.4.12 TCP Ping Availability Checks

Enter the target host (IP address or resolvable DNS hostname) and target port (TCP port number) you want to check. Additionally you can choose whether to display only the state or also the duration (in milliseconds) the check did use. Optionally you can define data to be sent with the TCP package.

#### 6.1.4.13 ICMP Ping Availability Checks

Enter the target host (IP address or resolvable DNS hostname) you want to check and optionally specify a timeout (in seconds). Additionally you can choose whether to display only the state or also the duration (in milliseconds) the check did use.

#### 6.1.4.14 Counter Check Using File System

In some situations counter values are much more complex to obtain or may be the result of other program executions. Therefor app.telemetry provides the “*counter check using file system*” to check and read a value from a text file on the local file system of the app.telemetry agent host.

All text files readable by the app.telemetry agent user located in the following directory are regarded as possible file counter check results and can be selected from a list box while creating or editing a service check of type “counter check using file system”:

- **Microsoft Windows (Agent platform):** <All Users - APPDATA>\Fabasoft app.telemetry\status
  - **Windows 2003:** C:\Documents and Settings\All Users\Application Data\Fabasoft app.telemetry\status
  - **Windows 2008:** C:\ProgramData\Fabasoft app.telemetry\status
- **Linux (Agent platform):** /var/opt/app.telemetry/status

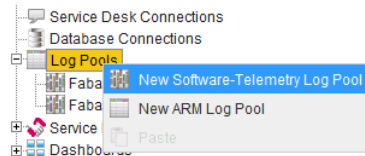
The first value in the first line of the status file will be used as counter check result value. Only numeric integer values are supported as result values. Decimal fraction parts will be cut off. Invalid value formats as well as textual values will be interpreted as simple text and cannot be validated with counter limits.

The validity of counter check results based on the change timestamp of the status file can be defined with the following two service check properties:

- **Check Interval [sec]:** the interval the value is updated from the status file (should be approximately the same as the status file is updated or at least twice often)
- **Value Expiration Factor:** the number of check intervals the result value is regarded as valid  $\langle \text{check interval} \rangle \times \langle \text{value expiration factor} \rangle = \langle \text{number of seconds} \rangle$  of max-age of status file (if the status file timestamp is older than that amount of time it is regarded as invalid and the counter check status is changed to warning “*last value too old*”. This mechanism helps detecting problems with the update-script that generates and updates the status file.

## 6.2 Using Software-Telemetry Log Pools and Top X Reports

Create a new “Log Definition” in the edit view by means of using the context menu on the “Log Pools” root group.



Enter all required fields and define the desired log definition for your application.

Set a name for the definition and define for which applications (**Application Filters**) the logs should be collected.

**Note:** The application filter definition for different log definitions should not overlap, because the first log definition will get the events that match the filter and the others will not.

A screenshot of a dialog box titled 'Log Pool Properties (Software-Telemetry)'. It has three tabs: 'Log Pool Properties' (selected), 'Log Definition Columns', and 'Dynamic Instrumentation'. The 'Log Pool Properties' tab contains the following fields:

- Name \***: Fabasoft app.telemetry End-2-End
- Log Pool Active**: ☒
- Restrict Access to User/Group**: apptelemetryusers
- Application Filters \***:
  - Application**: Fabasoft app.telemetry
  - Application ID**: [\* don't care]
  - Application Tier**: Client
  - Tier ID**: [\* don't care]
- Keep Online Log Requests in Memory \***: 120 minutes (0 ... to disable online logs)
- Database (for Persisting Logs)**: mainDB
- Database Table Prefix**: APM\_E2E
- Permanent Software-Telemetry Log Level \***: Standard
- Enable Statistic Reports**: Precalculated Statistics
- Forward Reported Sessions to Service Desk**: [ do not forward to service desk ]

At the bottom are 'OK' and 'Cancel' buttons.

Specify the database where you want to store the Software-Telemetry logs. If the database persistence of the telemetry requests is not required specify “[do not write to database]”.

The amount of requests held in memory is set by the number control above (it is defined in minutes with a maximum of 120 minutes = 2 hours). If you want to view the requests for a longer period in the online log view, you need to define a database for persisting the logs.

Select the desired recording level for permanent Software-Telemetry data collection (outside of manually started sessions).



To turn on Top X reports for that log pool you need to set the property “Enable Statistic Reports” to either “*Precalculated Statistics*” or “*On Demand Statistics*” and you also need to have a log definition with statistical flags (measures, dimensions) defined.

“*Precalculated statistics*” are calculated for 10-minute intervals all the time and are stored in separate database tables so Top X reports on a big amount of telemetry data are much faster, but this type of statistics will also use more memory and disk space!

“*On Demand Statistics*” will be calculated only when required for the requested time range directly on the database). A database must be defined in order to use “On Demand Statistics”!

Specify the log pool columns structure of your application by means of switching to the second tab (“*Log Definition Columns*”) selecting the log definition XML file from the file system and finally pressing the “Import”-button. This application log columns definition file (XML) is distributed by the application vendor of the instrumented application.

**Log Pool Properties (Software-Telemetry)**

Log Pool Properties | **Log Definition Columns** | Dynamic Instrumentation

<input type="checkbox"/>	Name	Module	Event ID	Param	Type	Flags
<input type="checkbox"/>		XMLHttpRequest	10		Block Start	block start
<input type="checkbox"/>	Action	Web Browser	71	1	String	dimension
<input type="checkbox"/>	Control	Web Browser	76	1	String	dimension
<input type="checkbox"/>	Loginname	Web Browser	35	1	String	dimension
<input type="checkbox"/>	Network Requests	XMLHttpRequest	0		Int32	count, measure
<input type="checkbox"/>	Privileges	Web Browser	35	2	String	dimension
<input type="checkbox"/>	Processing Time	XMLHttpRequest	11		Duration	block end, add, measure
<input type="checkbox"/>	View	Web Browser	22	1	String	dimension

Del Columns Page 1 of 1 10 View 1 - 8 of 8

Import existing log pool columns definition from XML-file (replaces current columns!)

Browse... Import

OK Cancel

**Note:** Reported feedback sessions (reported with the Report-SDK function from inside any application) will only be processed if there exists any log pool for that application that invoked the report function (by means of a matching application filter) or at least a default log pool without any application filter restrictions. Such reported sessions can be automatically forwarded to a defined service desk (if defined with the drop-down box) – each reported session will be forwarded to that service desk (if defined).

## 6.2.1 Top X Report / Drill-Down Analysis

After you have configured a Software-Telemetry log pool with a valid log definition (application log columns including dimension and measures flags) and the “*Enable Statistic Reports*” flag is turned on the Top X report view can be used to analyze the performance and bottlenecks of your instrumented application.

Switch to the Application view select the desired log pool and “Go to report view of this log pool”.

Fabasoft app.telemetry

Application Overview Log Pool: Fabasoft app.telemetry End-2-End

Log Pool Information

Application Name Fabasoft app.telemetry

Application Tier Client

Open configuration of this log pool

View requests of this log pool

Go to report view of this log pool

Inspect activities in this log pool

Running Telemetry Sessions

Start telemetry session for this log pool

Reported Telemetry Sessions

On the report view you have to select any dimension you are interested in and report time range and start the calculation (which may require some time to be calculated depending on the data amount and time range).

The resulting report will contain all request of that time range grouped for same values of the selected dimension.

For a drill-down analysis you can select one row and add this dimension as filter value (by means of using the context menu action “*Add Filter*”). Afterwards change the selected dimension to another interesting dimension and recalculate the report via the “*START*” button. To view the request of a selected report row double-click the row to switch to the telemetry view applying all filters according to the selected row.

Fabasoft app.telemetry

Log Pool: Fabasoft app.telemetry End-2-End Report for Fabasoft app.telemetry End-2-End

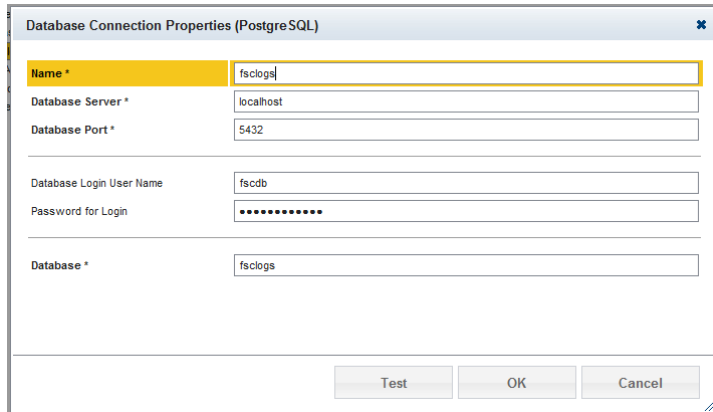
Dimension: Action Time Range: today Start: 2014-11-05 00:00 End: 2014-11-05 14:50 START

Filter: Add Filter Clear Filters View Requests

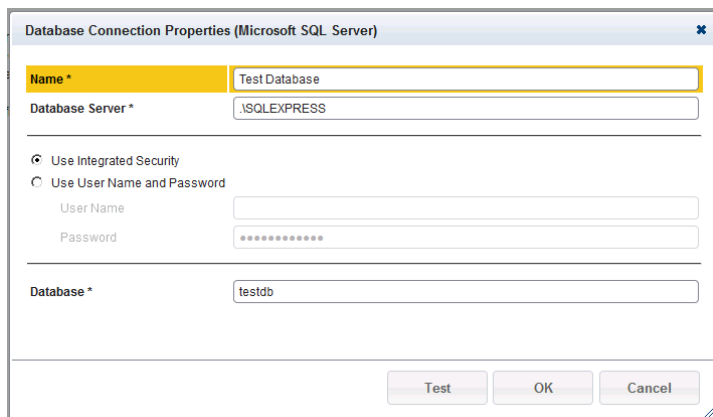
	Action	Count (#)	Duration (St)	Duration (AVG)	Duration (MAX)	Errors (SUM)	Network Requests (SUM)	Network Requests (AVG)	Processing Time (SUM)	Processing Time (AVG)
1	load dashboard configuration	1	9126 ms	9126.0000 ms	9126.0000 ms	0	3	3	244 ms	244.0000 ms
2	[not set]	4	3101 ms	775.2500 ms	2853.0000 ms	0	5	1	369 ms	123.0000 ms
3	Switch to Edit View	1	557 ms	557.0000 ms	557.0000 ms	0	2	2	334 ms	334.0000 ms
4	Switch to Application Overview	1	319 ms	319.0000 ms	319.0000 ms	0	3	3	141 ms	141.0000 ms
5	edit element	2	269 ms	134.5000 ms	269.0000 ms	0	1	0	76 ms	76.0000 ms
6	update charts	1	132 ms	132.0000 ms	132.0000 ms	0	1	1	118 ms	118.0000 ms

## 6.2.2 Configure a Database for Software-Telemetry Logs

On Linux systems specify the credentials and connection parameters for the PostgreSQL database. The PostgreSQL database must be set to a valid authentication mode – either set to local trusted access (“trust”) or to password based authentication (“password”).



On Microsoft Windows systems choose integrated or basic authentication, but basic authentication is not enabled on Microsoft SQL Server by default.



Define the database connection in the following format:  
<servername>\\<SQL-Server-Instance>

For example: “.\\SQLEXPRESS”: for local Microsoft SQL Server Express Edition.

Since version 2013 Summer Release the app.telemetry Server also supports the PostgreSQL database on Microsoft Windows (to be able to use a free and unlimited database). For details on how to install, configure and use PostgreSQL for Microsoft Windows see appendix chapter 9.7 “Using PostgreSQL Database for app.telemetry Server on Microsoft Windows”.

## 6.2.3 Filtered Log Pools

With Version 2014 Winter Release the filter rules for “*Filtered Log Pools*” have been reworked for easier editing and to support more complex filter expressions. With filtered log pools you can monitor and work on a specific subset of requests. A filtered log pool is based on a normal Software-Telemetry log pool and defines additional filter criteria. Filtered log pools store their subset of filtered requests in a separate database table therefore statistical analysis of filtered log pools also work as expected on that subset of data.

You can create a new “*Filtered Log Pool*” from inside the edit view by creating a new object below an existing base log pool or directly from the telemetry view by saving an existing filter set to a new or to an existing filtered log pool.

### Filter Definition:

The filter rules consist of a list of filter expressions which can be added or modified with a special sub dialog providing input hints and a list of properties (active, accept/deny, negate, comment). Any existing old filter expressions will be transformed into the new format and shown as “default” rule.

All incoming requests are checked against the filter rule chain from the top to the bottom. The first matching “accept”-rule will accept the request and stop checking any further rules in the chain. Note: You need at least one “accept”-rule at the bottom of the chain otherwise the incoming requests will not be accepted. Requests that are accepted by the filter rule chain are copied into the filtered log pool. The filter expressions can contain any column of the base log pool columns.

For easier filter definition you can try the effect of filters on the base log pool in the telemetry view and copy them over to the desired filtered log pool by means of using a sub-menu action of the “Save to ...” button.

Previously the filters have only supported a single equal comparison. We have removed this constraint by re-implementing the filters as full-featured custom filters that can be modified and combined on demand.

The filter rules for several different columns are connected with the AND-operator (&) which means every filter-rule must be fulfilled. A single filter rule can either be a simple expression or can now also be formed as complex rule with more than one expression. Those sub-expressions are by default connected with the OR-operator (match any rule) but can also be configured as AND-connected term (match all rules).

readtime >= 20 ms & readtime < 250 ms ✎ ✕ & application = FSCVENV@1.1001:TaskPaneApp or application = FSCVENV@1.1001:RefreshApp ✎ ✕

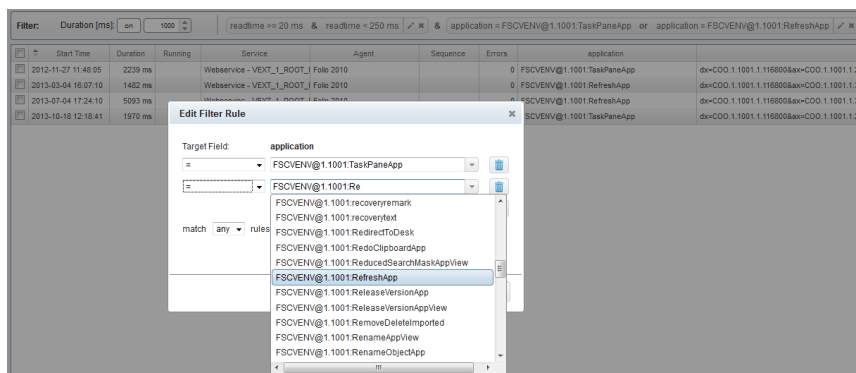
Filters can be added as before by selecting a single row (on report view) or a single table cell (on telemetry view) and pressing the “Add Filter” button or context menu entry. This action immediately

adds the filter to the filter bar with a default comparison operator (normally “=”). When adding a second filter value for the same column the existing filter rule for that column will be extended with a second OR-connected term.

In order to modify or extend an existing filter rule, just hit the edit button in the right actions area of the filter rule bar and an “Edit Filter Expression” dialog will be opened. This filter dialog is limited to the filter column the filter was defined for but can contain one or more filter expressions with different comparison operators and different values. Based on the column data type different operators and value hints are supported.

Special Data Type Handling:

- **Text** values (e.g. Loginname): support equals comparison, some kind of contains operators, regular expressions and auto-complete
- **Numeric** values (e.g. sendbytes): support numeric comparison (=, >, <, ...)
- **Duration** values (e.g. readtime): are formatted in milliseconds (ms) and support nearly the same numeric operators as numeric values (except equals comparison because of the inaccuracy of the high resolution time values)
- **Date/Time** values: are formatted as human readable date/time entries and are presented in a date/time-input control for easier date/time input. This type supports ordered comparison (=, >, <, ...)
- **Fabasoft Object** values (e.g. dstappview): are using internal COO-address formatting rules as well as name-resolution via fscdata.xml. The display value of the filter is the object name if available. The auto-complete input will allow you to find the appropriate Fabasoft object with a prefix-match. Because of the complex internal handling only equal-comparison is possible for that type.



## 6.2.4 View on Log Pools

With Version 2014 Summer Release a new extended type of log pool – the “View on Log Pool” was introduced. This new view log pool serves as the name implies as logical view on the base log pool with some special conditions.

The dialog box titled "View on Log Pool Properties" contains the following fields and controls:

- Name \***: Client User-View
- Log Pool Active**: ☒
- Restrict Access to User/Group**: APM-Users
- Base Log Pool**: Fabasoft app.telemetry End-2-End
- View Filter**: Add New Filter Rule
- Filter Expression**: Loginname = "\$username" (with edit and delete icons)
- Buttons**: OK, Cancel

A view log pool can be created on any other log pool by means of selecting the desired base log pool and creating a new “View on Log Pool” using the context specific menu button. You can specify a view filter expression based on any dimension column defining the condition which requests are shown in the view. Additionally to normal literal filter expressions a special dynamic value placeholder “\$username” can be used to limit a view on only those requests produced by the user itself.

**Example:** In order to allow your developer team members access to telemetry requests or feedbacks of their own the administrator has to create a new view on the base log pool, limit the access to an access group containing the developer accounts and create a view filter restricting the login name column the dynamic \$username filter value.

Additionally a privileged user can delegate any feedback (he has access to) to another user (support team, developer, ...). Just select a desired feedback on the Inbox view, click the “Route to ... User” toolbar button and add the desired login name of the target user who you want to give access to. This dialog can also be used to remove access delegation rights again.

The dialog box titled "Manage User Access Permissions for Feedback" is overlaid on a feedback list. The list has columns: Time, Status, Star Rating Control, login, my\_rating\_control, and description. A selected row shows: 2014-03-12 23:53:38, [icon], 1, pool2, 1, neues Feedb.

The dialog box contains the following fields and controls:

- Additional users with access:** pool1 (with a Remove button)
- Enter the login name (email address) of the user who you want to allow access to this forms.**
- Login Name / E-Mail**: developer (with an Add button)
- Note:** The added users will be able to fully access this feedback.
- Buttons:** OK, Cancel

## 6.2.5 Syslog Log Pools

Syslog log pools are a way to handle a big amount of syslog entries in a well-defined and structured way using the power of Software-Telemetry log pools in combination with powerful filter rules within filtered log pools.

In order to use Syslog log pools you have to install an app.telemetry syslog forwarder module on every target system you want to capture syslog entries from. For more details on prerequisites and installation instructions see the appropriate chapters above.

Once this apptelemetrysyslogforwarder daemon is configured and started you have to configure a new Software-Telemetry log pool (app.telemetry client - edit view) and set up the filters to “Fabasoft app.telemetry” as application name and “Syslog Forwarder” as application tier name. With the “Permanent Software-Telemetry Log Level” field you can configure the amount/severity of log entries to be captured (e.g.: with level “Detail” you will also get “Informational” log entries but not those of “Debug” severity).

The screenshot shows the 'Log Pool Properties (Software-Telemetry)' window. It has three tabs: 'Log Pool Properties', 'Log Definition Columns', and 'Dynamic Instrumentation'. The 'Log Pool Properties' tab is selected. The form contains the following fields and values:

- Name \***: Syslog Pool
- Log Pool Active**: ☒
- Restrict Access to User/Group**: (empty text box)
- Application Filters \***:
  - Application**: Fabasoft app.telemetry
  - Application ID**: [\* don't care]
  - Application Tier**: Syslog Forwarder
  - Tier ID**: [\* don't care]
- Keep Online Log Requests in Memory \***: 10 minutes (0 ... to disable online logs)
- Database (for Persisting Logs)**: SampleAppDB
- Database Table Prefix**: syslog
- Permanent Software-Telemetry Log Level \***: Standard
- Enable Statistic Reports**: On Demand Statistics
- Forward Reported Sessions to Service Desk**: N/A

At the bottom right, there are 'OK' and 'Cancel' buttons.

In order to reduce the memory usage for that log pool you should set up a database for persisting the syslog entries and set the limit for in-memory requests to a very low value as well as the setting for the statistics calculation to “On Demand Statistics”. A predefined log definition for that log pool is also available within the KIT (...\\Telemetry-Modules\\LogDefinitions\\syslogforwarder-logdefinition.xml) that should be imported via the 2<sup>nd</sup> tab (Log Definition Columns).

After you have set up your syslog log pool you can watch incoming syslog entries as telemetry requests in the created log pool and you also can analyze the entries via the report view.

In order to reduce the set of log entries you can create a filtered sub log pool with a defined set of filter rules (supporting complex expressions as well as regular expressions).

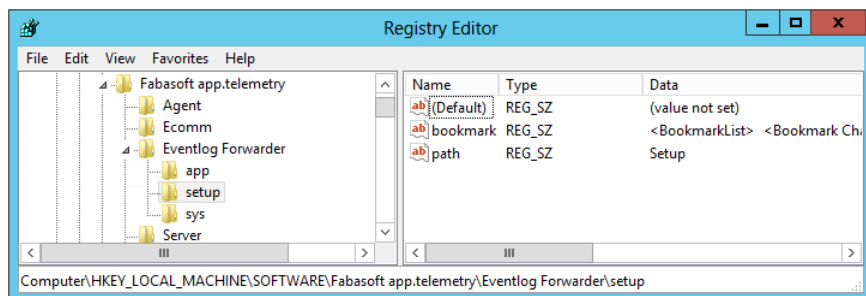
## 6.2.6 Microsoft Windows Eventlog Forwarder

Just like the syslog forwarder on linux, Fabasoft app.telemetry provides a Microsoft Windows service to read Eventlog entries. In order to install this service start the eventlogforwarder-setup.exe from

the `app.telemetry\Telemetry-Modules\WINDOWS_X64` folder of the Fabasoft `app.telemetry` installation package.

In order to configure a log pool receiving the eventlog data, create a new *Software-Telemetry Log Pool*, select “*Fabasoft app.telemetry*” as the *Application* and “*Eventlog Forwarder*” as the *Application Tier*. Choose a Permanent Software-Telemetry Log Level to filter event logs based on the event level. Import the `eventlogforwarder_logdefinition.xml` from the *LogDefinitions* folder into the *Log Definition Columns* will provide the common properties of the events in the request view. Create *Filtered Log Pools* or *View Log Pools* to filter eventlog entries as needed.

By default the Fabasoft `app.telemetry` Eventlog Forwarder will read events from the System and the Application source. In order to read event from other sources use the Registry Editor to add an additional key under `HKLM\SOFTWARE\Fabasoft app.telemetry\Eventlog Forwarder` and add a String Value named `path` holding the Name of the event log source.



The bookmark property is used by the eventlog forwarder to store the bookmark to the last entry which has been forwarded so that it knows where to continue reading data even if the service is restarted. You may force the retransmission of all entries by deleting the bookmark and restarting the eventlog forwarder service.

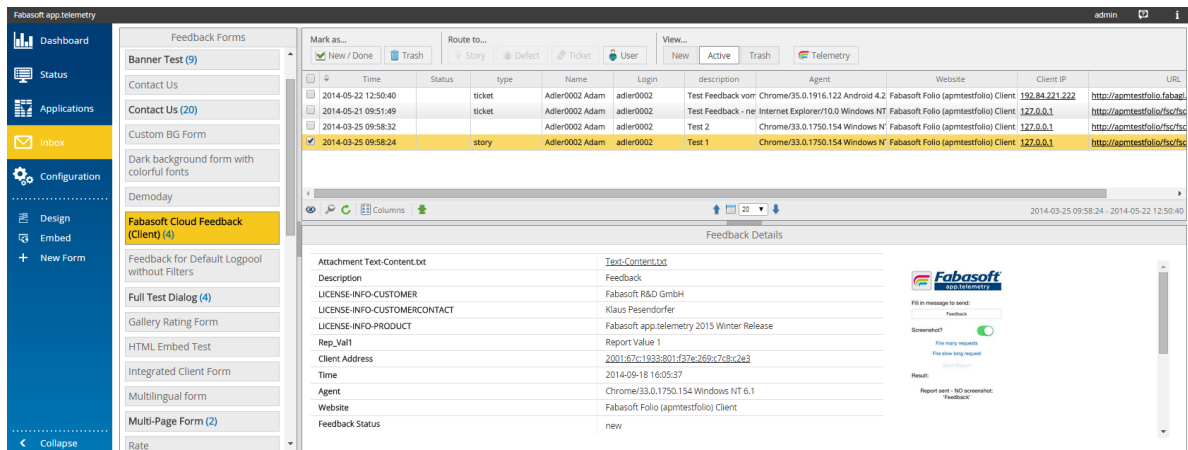
## 6.3 Inbox for Incoming Feedbacks

With version 2013 Fall Release a new view called “Inbox” was introduced showing any incoming feedbacks (also known as reported sessions sent by an instrumented application via the reporting API calls) as structured list with a quick preview of containing meta data (including a screenshot if available). The feedbacks will still be listed in the as reported session on the application log pool detail page.

The feedbacks are grouped by the instrumented application which sent the feedback. If there does not already exist a form log pool for an incoming feedback a new auto-generated form log pool will be created which cannot be designed via the forms designer.

**Note:** A Software-Telemetry log pool for the instrumented application (which sends the feedback) must exist in order to get feedbacks from the application.

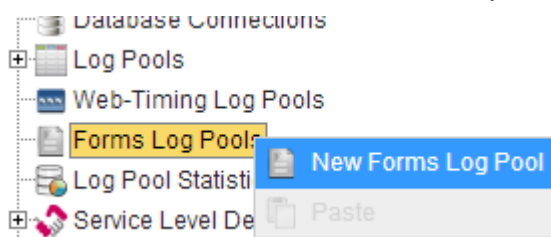




## 6.4 Feedback Forms Designer

Since version 2013 Fall Release a graphical forms designer was added to the on-premise version which helps you to design full-featured feedback forms by means of drag&drop fields into a form and customize the text, colors and styles.

In order to use these new feedback forms you have to follow the steps below:



1. Design a new form
  - 1.1. Navigate to the "Edit" view inside the app.telemetry client
  - 1.2. Select the infrastructure group "Forms Log Pools" and click the menu action "New Forms Log Pool" (or "Design" if you already have an existing designed form).
  - 1.3. Design your form
    - 1.3.1. Configure the base settings of your form (title, styling, fonts, colors, logo, background, ...) by means of using the "Customize Form" button.
    - 1.3.2. Add input fields to your form by means of dragging some of the available form controls from the left side to the form preview.

- 1.3.3. Enter a name and optionally a description and save the form with the “OK” button at the bottom.

Main Layout:

Customize Form ...

Add Finish Page ...

Form Controls:

- Multiline Text
- Text Input
- E-Mail
- Check Box
- Radio Button
- Check Box Group
- Combo Box
- Rating
- Readonly
- Rating Group
- Static Text
- Screenshot
- Embed HTML
- Separator
- New Page
- US State
- Masked Input

Drag & Drop Fields into the Form

To edit field properties move your mouse over the field in the preview. To change form title, background, logo or fonts click the "Customize Form" button on the left.

Send Feedback

Drag fields from the left and drop them here.

Cancel Send

Form Preview

Finish Form Creation

Choose a short display name for your form to distinguish them in the forms list.

Form Name

Form Description

## 2. Software Update:

- 2.1. Update all app.telemetry software (Server, Agent, WebAPI) in your infrastructure to 2013 Fall Release or later.
- 2.2. Update the JavaScript SDK (softwaretelemetry.js) used by your instrumented application (e.g. Folio web client) to the latest version shipped with the app.telemetry software package (can be found in the folder "Developer\JavaScript"). This update (to 2013 Fall Release or later) enables the SDK to load the feedback dialog resources from the WebAPI.

## 3. Enable the new feedback form (Form Log Pool) as report dialog for your instrumented application:

- 3.1. Either by assigning the "Forms Log Pool" as default form for a "Software-Telemetry Log Pool" (of an instrumented application) by means of using the "Set as default Form" menu action.
- 3.2. Or by manually adding the new "Form ID" (which can be found in the forms designer after editing the created form in the form properties at the bottom of the page) to the ReportDialog API-call (in your instrumented application) as last parameter:

```
apm.ReportDialog(null, _filter, _reportkey, _description, _parentNode, _metadata, _formOptions);
```

**Example:** `apm.ReportDialog(null, _filter, _reportkey, null, null, null, {formid: "FORM12345", language: "de"});`

Rating Group

Screenshot

Separator

US State

Finish Form Creation

Choose a short display name for your form to distinguish them in the forms list.

Form Name

Form Description

Form ID

FORM64058

4. Test sending new feedbacks from your instrumented application and watch the app.telemetry “Inbox” view for new incoming feedbacks.
5. In order to receive feedback notifications via e-mail you have to configure a default notification channel in the “Server Properties” dialog at the “Feedback Configuration” tab.

## 6.5 Configure Web Timing

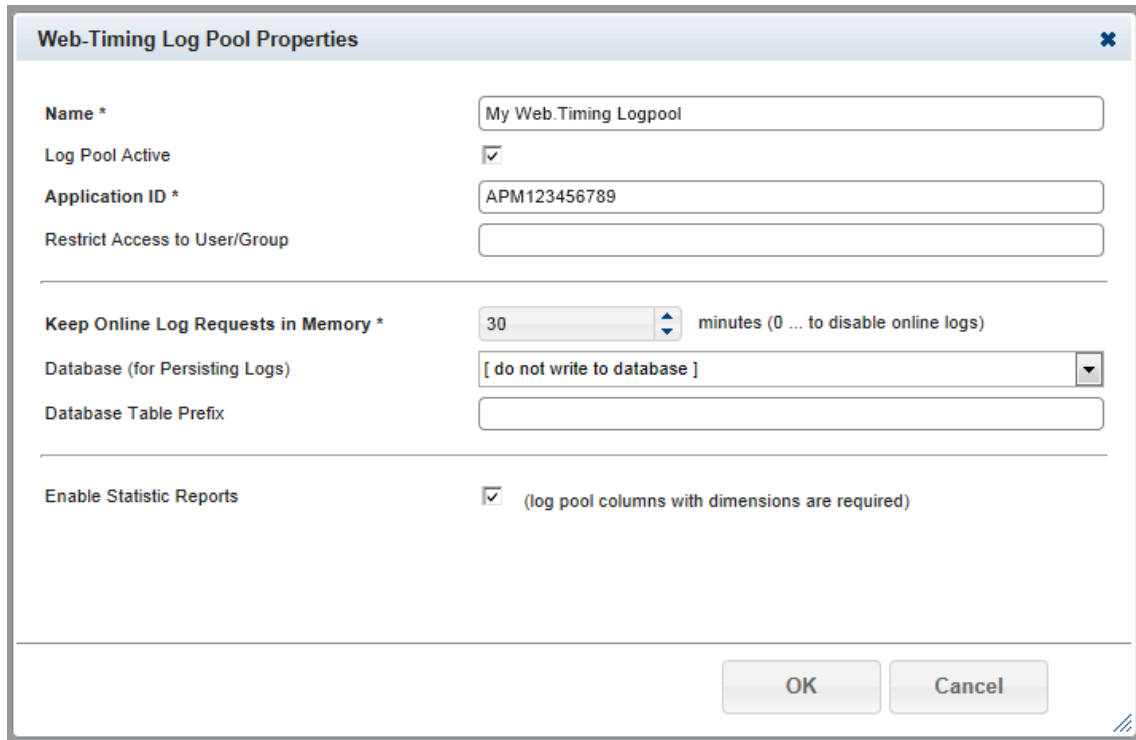
For detailed end-user experience monitoring including page loading times from your web browser client you have to include a JavaScript block into the web page you want to monitor and you have to create a “*Web Timing*” log pool in your app.telemetry infrastructure.

In order to enable the “Browser Telemetry” for your web page you have to put the browser-telemetry script somewhere reachable for your web server and then include the following script block in your web page HTML code (the best choice is to put the code in a HTML header template for all of your web pages).

### JavaScript Snippet to Include for Web Timing

```
<script type="text/javascript"> /**/<br/>  __apmcfg={<br/>    id:"APM12345", /* replace with your desired application id */<br/>    ts:new Date(),<br/>    url: "http://yourdomain.com/", /*replace URL with your<br/>app.telemetry WebAPI server*/<br/>    base: "///yourdomain.com/yourdirectory/" /*required for feedback<br/>button/dialog*/<br/>    /*,startupTimeout: 3000 // optional, default: 3000ms */<br/>  };<br/>/*]]&gt;*/&lt;/script&gt;<br/>&lt;script type="text/javascript"<br/>src="//yourdomain.com/yourdirectory/apm.js"&gt;&lt;/script&gt;<br/><!-- replace above script location with real URL where script is placed<br/>and reachable --&gt;</pre></div><div data-bbox="142 637 899 681" data-label="Text"><p>The script containing the logic for measuring the page speed and transmitting the data to the app.telemetry WebAPI is named “apm.js” and is available in the Fabasoft app.telemetry software kit under Developer\JavaScript.</p></div><div data-bbox="854 935 886 951" data-label="Page-Footer"><p>43</p></div>
```

The last step is to create a new “Web Timing Log Pool” in your app.telemetry infrastructure from within the edit view and filling in your chosen application id.



The image shows a dialog box titled "Web-Timing Log Pool Properties". It contains several fields and checkboxes for configuring a log pool. The fields are: "Name \*" with the value "My Web.Timing Logpool", "Log Pool Active" with a checked checkbox, "Application ID \*" with the value "APM123456789", "Restrict Access to User/Group" (empty), "Keep Online Log Requests in Memory \*" with a value of "30" and a unit of "minutes (0 ... to disable online logs)", "Database (for Persisting Logs)" with a dropdown menu showing "[ do not write to database ]", and "Database Table Prefix" (empty). There is also a checkbox for "Enable Statistic Reports" which is checked, with a note "(log pool columns with dimensions are required)". At the bottom right are "OK" and "Cancel" buttons.

## 6.6 Configure Web Timing with Feedback Dialog

Fabasoft app.telemetry also supports the integration of an end-user feedback dialog into your web application. The feedback dialog transmits the text message the user entered with additional metadata and the request information about the last clicks of the user with a reported session to the Fabasoft app.telemetry server (via app.telemetry WebAPI).

In order to use the feedback dialog for web pages monitored with Browser Telemetry API ensure that you have defined the “base” option in your browser telemetry code block (as mentioned above) included in your web page and put the following resources into the same directory on your web server (accessible by the web browser clients):

- apmdialog.js
- apmdialog.css
- since Version 2014 the images are loaded directly from the WebAPI

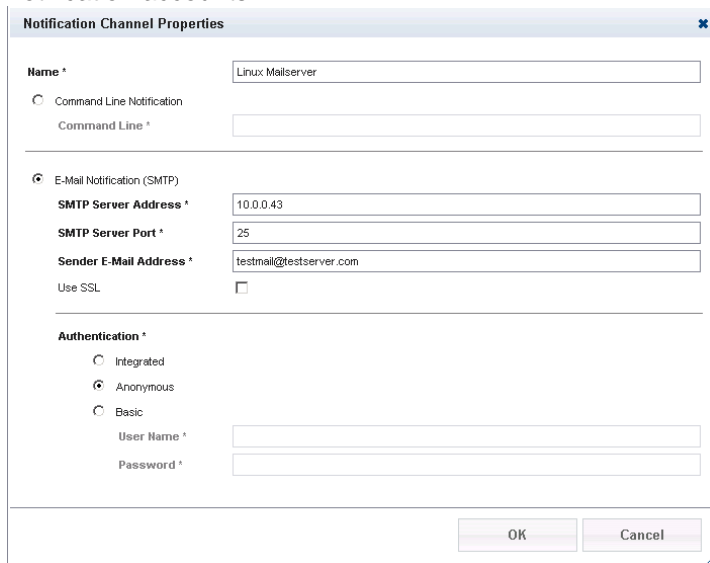
To include the feedback button in your page just include the following HTML snippet somewhere in your HTML page (template):

### HTML Snippet to Include for Feedback Button

```
<button class="apm-feedback-button apm-feedback-button-fixed apm-  
feedback-button-top apm-feedback-button-right">  
    <span class="apm-feedback-icon apm-ic-logo20"></span>  
    <span class="apm-feedback-button-text">Feedback</span>  
</button>
```

## 6.7 Configure Notifications for Status Changes

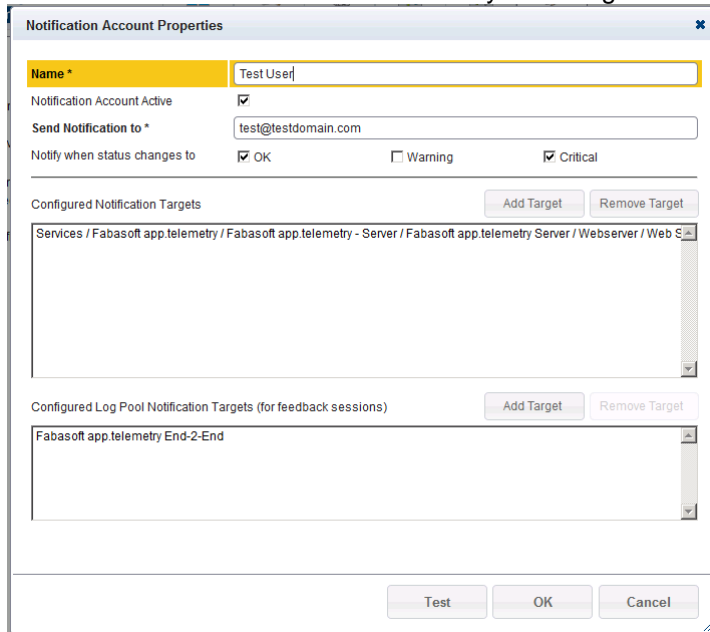
1. Create a “*Notification Channel*” defining the mail server used to send notifications to the desired notification accounts.



The "Notification Channel Properties" dialog box is shown. It has a title bar with a close button. The "Name" field is set to "Linux Mailserver". There are two radio buttons: "Command Line Notification" (unselected) and "E-Mail Notification (SMTP)" (selected). Under "E-Mail Notification (SMTP)", there are three text fields: "SMTP Server Address" (10.0.0.43), "SMTP Server Port" (25), and "Sender E-Mail Address" (testmail@testserver.com). There is a checkbox for "Use SSL" which is unchecked. Below these is an "Authentication" section with three radio buttons: "Integrated" (unselected), "Anonymous" (selected), and "Basic" (unselected). Under "Basic", there are text fields for "User Name" and "Password". At the bottom are "OK" and "Cancel" buttons.

**Note:** If you run the Fabasoft app.telemetry server on Linux, the e-mails are sent via the local sendmail process on the Linux server, so you have to setup sendmail to forward the e-mails to your desired SMTP server. It is not possible to configure the remote SMTP server on a Fabasoft app.telemetry Linux server via the Fabasoft app.telemetry client interface. For details how to configure sendmail to forward e-mails to an SMTP server see appendix.

2. Configure target “*Notification Accounts*” that should be notified of any service state change. Create a new notification account inside your configured notification channel.



The "Notification Account Properties" dialog box is shown. It has a title bar with a close button. The "Name" field is highlighted in yellow and set to "Test User". There is a checkbox for "Notification Account Active" which is checked. The "Send Notification to" field is set to "test@testdomain.com". There are three checkboxes for "Notify when status changes to": "OK" (checked), "Warning" (unchecked), and "Critical" (checked). Below this is a section for "Configured Notification Targets" with "Add Target" and "Remove Target" buttons. The list contains: "Services / Fabasoft app.telemetry / Fabasoft app.telemetry - Server / Fabasoft app.telemetry Server / Webserver / Web S". Below that is a section for "Configured Log Pool Notification Targets (for feedback sessions)" with "Add Target" and "Remove Target" buttons. The list contains: "Fabasoft app.telemetry End-2-End". At the bottom are "Test", "OK", and "Cancel" buttons.

3. Define the receiver's e-mail address and the service elements (service group, service, service check) that you want to be notified of any status change (you can select on which status change this account should be notified).
4. Additionally you can select log pools if you want to get notified on any feedback escalation sent for that application monitored with that log pool.

### 6.7.1 Configure Notification Schedules

Since Fabasoft app.telemetry 2010 Fall Release you could additionally define “*Notification Schedules*” for complex situations that require fine-grained notification settings based on SLA and holiday definitions and maintenance times.

Notification schedules define additionally to the notification account settings (“Notification Account Active” and “Notify when status changes to ... OK|Warning|Critical”) when a notification for a specific notification target (service group, service, service check, ...) is to be sent.

Notification schedules are based on a time zone, holiday definition and service level definition which help you to prevent sending of notifications during maintenance time ranges defined in the service level definition.

The screenshot shows the 'Notification Schedule Properties' dialog box with the 'Notification Schedule Properties' tab selected. The fields are as follows:

Field	Value
Name *	Web Server Notifications
Time Zone *	Europe/Vienna
Holiday Definition	Austrian Holidays (Template)
Service Level Definition	Web Server SLA
Send Notifications during Core Time	<input checked="" type="checkbox"/>
Send Notifications during Remaining Time	<input checked="" type="checkbox"/>
Send Notifications during Maintenance Time	<input type="checkbox"/>

Buttons: OK, Cancel

Additionally to these basic settings you could also define time ranges where you do not want to send any notifications (e.g.: recurrent time ranges for a weekend day, or non-recurrent time ranges for a holiday).

The screenshot shows the 'Notification Schedule Properties' dialog box with the 'Recurrent Time Ranges' tab selected. The table displays the following data:

Name	Type	Start Time	End Time	Start Series	End Series	Recurrent
Do not disturb on sunday	Don't Send Notifications	00:00	00:00	2010-10-20		Sun
Saturday Afternoon	Don't Send Notifications	16:00	20:00	2010-10-20		Sat

The screenshot shows the 'Notification Schedule Properties' dialog box with the 'Non-Recurrent Time Ranges' tab selected. The table displays the following data:

Name	Type	Start Date	Start Time	End Date	End Time
on holiday	Don't Send Notifications	2010-10-22	16:00	2010-10-25	08:00

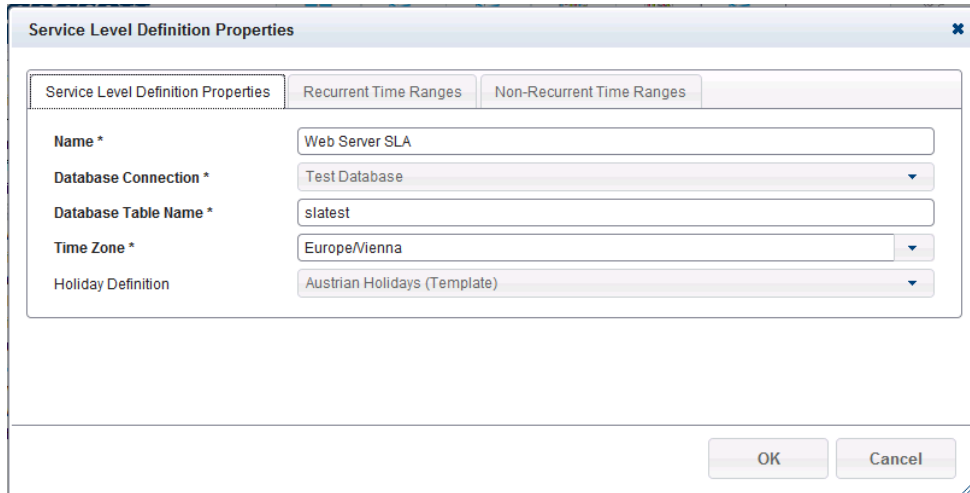
**Note:** The time range definitions for a notification schedule will take precedence over the time range settings in the used service level definition and non-recurrent time ranges also take precedence over recurrent time ranges.

**Note:** The notification schedule settings will only take effect if you specify valid notification targets. The notifications targets should match the targets configured for the notification account.

## 6.8 Configure Service Level Definitions

Service level definitions will help you monitor business critical services in order to fulfill defined service level agreements. Based on a service level definition, you can select service checks using this definition to monitor your service health. A service level definition requires a database to persist the service check state changes and a time zone which defines the base for all your time settings.

If you select to use a holiday definition then all holiday entries will be regarded as “remaining time” for the SLA calculations.



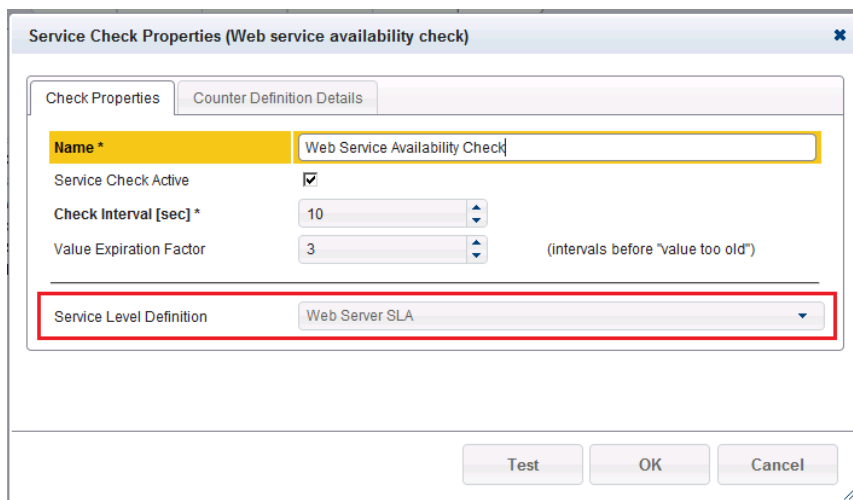
The screenshot shows the 'Service Level Definition Properties' dialog box. It has three tabs: 'Service Level Definition Properties', 'Recurrent Time Ranges', and 'Non-Recurrent Time Ranges'. The 'Service Level Definition Properties' tab is active. It contains the following fields:

- Name \***: Web Server SLA
- Database Connection \***: Test Database (dropdown)
- Database Table Name \***: slatest
- Time Zone \***: Europe/Vienna (dropdown)
- Holiday Definition**: Austrian Holidays (Template) (dropdown)

At the bottom right, there are 'OK' and 'Cancel' buttons.

A reasonable service level definition requires the definition of a “core time”. Generally this is a recurrent time range based on time ranges for your weekdays.

After the service level definition is completely configured you can attach it to any availability service check.



The screenshot shows the 'Service Check Properties (Web service availability check)' dialog box. It has two tabs: 'Check Properties' and 'Counter Definition Details'. The 'Check Properties' tab is active. It contains the following fields:

- Name \***: Web Service Availability Check
- Service Check Active**: ☒
- Check Interval [sec] \***: 10 (spinner)
- Value Expiration Factor**: 3 (spinner) (intervals before “value too old”)
- Service Level Definition**: Web Server SLA (dropdown, highlighted with a red box)

At the bottom, there are 'Test', 'OK', and 'Cancel' buttons.

Service checks with attached service level definition can be viewed in a dashboard chart to see the health in percentage of reached availability as defined in your SLA. (This needs to be configured explicitly.)

## 6.9 Using Dashboards and Charts

In order to create your own dashboard switch the web browser client to the edit view and create a new “*Dashboard*” object below the root group “*Dashboards*”.

**Note:** “*public*” dashboards are available to all app.telemetry users whereas non-public dashboards are not available for app.telemetry dashboard users (so dashboard users can only view public marked dashboards and nothing else within the app.telemetry web client).

### 6.9.1 Chart and Data Source Types

Inside this new dashboard object you can create new charts for the following data source types:

- Availability Check
- Counter Check
- Request Duration Categories
- Top-X Reports
- Top-X Log Pool Statistics
- Service Group State
- CSV File
- Remote

#### 6.9.1.1 Availability Check Chart

For charts with data source type “*Availability Check*” you have to select a service check with attached service level definition by means of using the “*Add Service Check*” button and selecting the check from the infrastructure tree.

If you select more than one service check for your availability chart, the SLA calculation will interpret the service as available if at least one of the selected service checks is available.

You could overwrite the used service level definition with the combo box in order to use any specific SLA definition or select “<default from check>” to use the SLA configured in your (first) service check.

The time range can be one of list of predefined time ranges or a custom date range.

Depending on your desired presentation form you can select percentage values or absolute time values.



### 6.9.1.2 Counter Check Chart

For charts with data source type “*Counter Check*” you have to select one or a list of service checks by means of using the “*Add Counter*” button.

Additionally you can decide on current values (for bar chart, gauge chart or similar) or a value trend (for line chart).

The counter checks can be reordered by means of using drag&drop with the mouse or by selecting a single check and press the keys CTRL+<key-up> to move the check upwards or CTRL+<key-

down> to move the check downwards. The ordering of the checks will be reflected in the ordering of the checks in the graphical chart representation (order, color).

The time range of values available for the chart is limited by the time app.telemetry keeps counter data in memory. This time can be configured in the “Data Cleanup Settings” of the “Server Settings” object, where you can set the “Data Online Time (hours)”. For long term counters (e.g. disk space usage trends) you can select a database in the Service Check properties where to put the values on. The “Counter Check Chart” will automatically read counter check data from the database if online data are not sufficient. The available time range is limited by the maximum count of 1000 values per counter. Configure longer time intervals in the Service Check properties to extend the available time range in the chart.

### 6.9.1.3 Request Duration Categories Chart

For charts with data source type “*Request Duration Categories*” you have to select a log pool and a list of request duration range classes to see how many requests for a defined time range class occur.

Chart Properties (Request Duration Categories)

Chart Properties | Data Source Properties

Log Pool \* Fabasoft app.telemetry End-2-End

Data Time Frame \* (from Memory Cache)

☒ All Values

☐ Limit Range last 60 minutes

Request Duration Range Classes \*

≥ < [sec] Add

0-0.5  
0.5-1  
1-2  
2- Remove Edit

Chart Values \*

☐ Percentage (%)

☒ Count

Filter Expression (SQL-like syntax) \*duration' < 600000000

OK Cancel

A popular representation for this type of chart is the pie chart.

**Note:** This data source/chart only shows requests from the log pools online memory cache (and not from database), so check your log pool settings if you see fewer requests than you expect.

With the optional filter expression you can exclude some unwanted requests from your chart. The example in the screenshot above (“duration” < 600000000) includes (by means of an SQL WHERE-clause) all requests with duration smaller than 60 seconds, so it excludes requests with a timeout of 60 seconds or more. The filter is based on the internal data structures of the app.telemetry server, so the duration for example is based on a 100ns-timescale.

### 6.9.1.4 Top-X Reports Chart

For charts with data source type “*Top-X Reports*” you have to select a log pool for your Top-X report charts.

Then you have to select a list of days (the chart is presented as 24-hour line for each day compared in the same chart). So you can for example compare request duration or request count behavior for different days or against a defined baseline day.

The chart values define which measure or dimension (count of different dimension values) is used for the Top-X calculation.

With the optional filter expression you can exclude some unwanted requests from your chart. The example in the screenshot below ("duration" < 600000000) includes (by means of an SQL WHERE-clause) all requests with duration smaller than 60 seconds, so it excludes requests with a timeout of 60 seconds or more. The filter is based on the internal data structures and database tables of the app.telemetry server, so the duration for example is based on a 100ns-timescale.

Chart Properties (Top-X Reports)

Chart Properties | Data Source Properties

Log Pool \* Fabasoft app.telemetry Webserver

Report Days (comparison of timeline per day/24 hours)

☐ Predefined Day yesterday

☐ Day - 7

☒ Date 2012-07-31

Add Day to List Remove Day from List

Current Selection \*

today  
yesterday  
day-7  
2012-07-31

Chart Values \*

☒ Measure Count

☐ Count for Dimension Method

DB Filter (SQL syntax) "duration" < 600000000

OK Cancel

#### 6.9.1.5 Top-X Log Pool Statistics Chart

Since version 2014 Spring Release the configuration of Top-X log pool statistic charts have been improved a lot. First of all you have to enable statistic reports for your log pool (using "Precalculated Statistics") and then you can create a new chart of type "Top-X Logpool Statistics" and choose the desired values from the combo boxes as shown in the example dialog below.

**Chart Properties (Top-X Logpool Statistics)**

Chart Properties | Data Source Properties

Log Pool \* Fabasoft app.telemetry End-2-End

Log Pool Statistic <Default Statistics>

---

Time Range (last ...) 288 intervals (resolution: 10 minutes)

Group By Action

Order By Duration (AVG) descending

DB Filter (SQL syntax)

Result Display Limit 0 (0 ... unlimited)

---

**Statistic Measures \***

Measure Name: Errors (SUM) Add

Count  
Duration (AVG)  
Network Requests (AVG)  
Processing Time (AVG)

Remove

OK Cancel

For special purpose you could still create and use your own log pool statistic definition (based on a log pool and a database) defining how some special statistics are calculated. A knowledgebase article (*“Log Statistics”*) describes this extended usage more detailed – for more help contact the Fabasoft support.

#### 6.9.1.6 Service Group State Chart

For charts with data source type *“Service Group State”* you have to select a list of service groups you want to explicitly see in the dashboard view for this chart.

**Chart Properties (Service Group State)**

Chart Properties | Data Source Properties

**Selected Service Groups** Add Service Group Remove Service Group

Services / Windows Services / Local Win08 Server

Services / Test Group (Critical)

Services / Test Group (Warning)

OK Cancel

The counter checks can be reordered by means of using drag&drop with the mouse or by selecting a single check and press the keys CTRL+<key-up> to move the check upwards or CTRL+<key-

down> to move the check downwards. The ordering of the checks will be reflected in the ordering of the checks in the graphical chart representation (order, color).

### 6.9.1.7 CSV Chart

Using CSV files you may integrate data from various data sources into your dashboard.

Put your data in a file in the following directory

- C:\ProgramData\Fabasoft app.telemetry\worker\chart (Microsoft Windows)
- /var/opt/app.telemetry/worker/chart (Linux)

The CSV file content has to be of following format:

- The first line contains the column description – with quoted labels separated with semicolons.
- The first column contains the time dimension (if applicable) – if you don't use time-based values skip this column with a leading semicolon in every line (also in the first header line)

Values must be separated by semicolons (";")

Quote text with double quotes (") and duplicate double quotes inside quoted texts.

The following example shows a CSV data file with time-based data. The time-series (first column) does not have a header (first cell in first row empty – leading semicolon) but all data rows start with a timestamp entry. This format can be used for line-charts or tables.

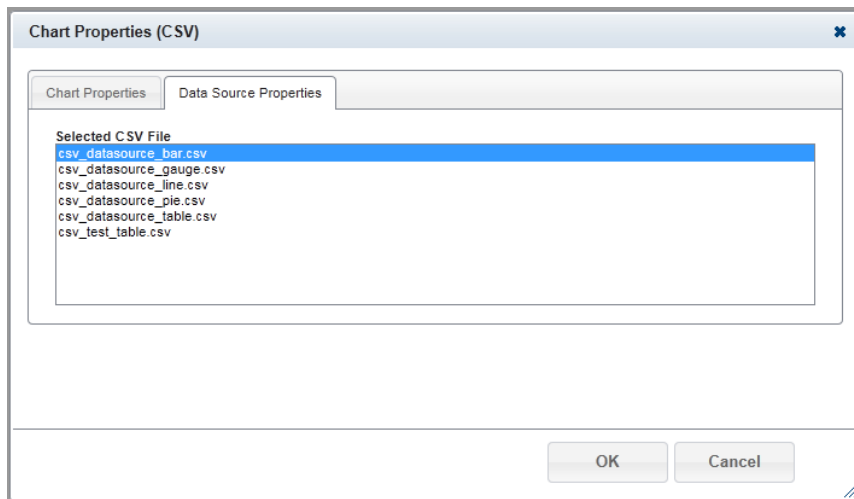
#### Example: CSV Chart Data File (time-based)

```
;"Data Set 1";"Data Set 2"  
"2010-01-15";"680";"419"  
"2010-01-16";"702";"458"  
"2010-01-17";"745";"491"  
"2010-01-18";"820";"516"
```

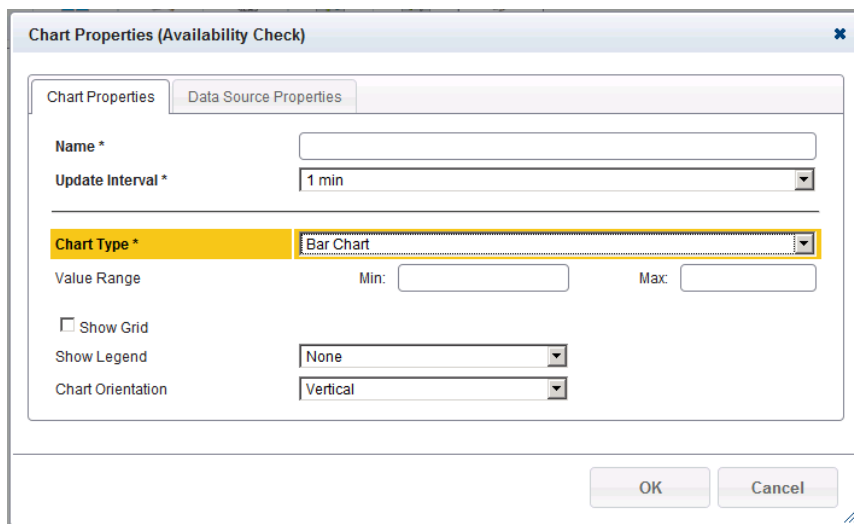
The following example shows a CSV data file with data that is not time based by skipping the first column with a leading semicolon (;) in every line. This format can be used for bar-charts, gauge-charts, pie-charts or a simple table.

#### Example: CSV Chart Data File (not time-based)

```
;"Disk C:\";"CPU (Total)";"Memory";"Network"  
;75;88;77;12
```



## 6.9.2 General Chart Properties



Depending on the selected data source and data properties different chart types with different chart properties are available:

- Line Chart
- Bar Chart
- Pie Chart
- Gauge Chart
- Table
- Status List

Service Group Status Chart	
✓	Local Win08 Server
✗	Test Group (Critical)
⚠	Test Group (Warning)

app.telemetry End-2-End: Request Duration Statistic				
< 0.3 seconds	0.3 - 0.5 seconds	0.5 - 1 seconds	1 - 2 seconds	> 2 seconds
58.06%	8.06%	30.65%	0.00%	3.23%



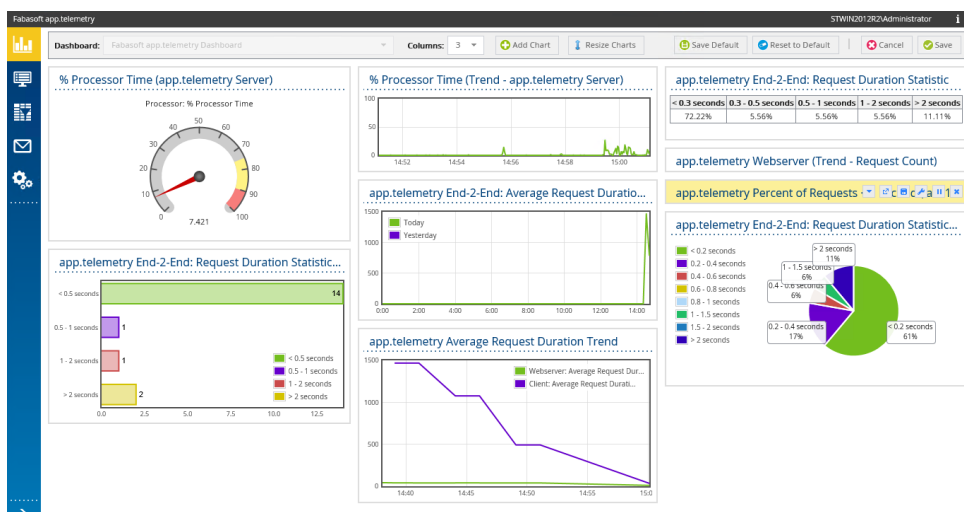
### 6.9.3 Dashboard View

In the dashboard view you can select which dashboard to view, with how many columns your dashboard should display your charts.

The dashboard view can be locked to prevent unintended modifications. In order to change the number of chart columns, the position and size of charts or to add or remove charts just click the “*Unlock Dashboard*” button to enable modifications.

To hide or remove a chart from your dashboard, just click the remove button in the top-right corner of a chart.

In order to add any hidden charts again to your dashboard use the “*Add Chart*” button. This will open a dialog to select the displayed charts.



You can reorder (drag&drop on the chart title bar) or resize the charts simply with the mouse. Your arranged set of charts can be saved as default initial setting for all new users viewing this dashboard with the “*Save Default*” button.

## 7 Product Version Upgrades

**Note:** A mixed installation with different Fabasoft app.telemetry product versions is not supported!

So you have to update the Fabasoft app.telemetry server, all Fabasoft app.telemetry agents and all Fabasoft app.telemetry web services (WebAPI) installations.

**Note:** Before starting the update process of the software, you should backup the configuration data of the Fabasoft app.telemetry server:

- Linux: `/etc/app.telemetry/infra.xml`
- Microsoft Windows: `C:\ProgramData\Fabasoft app.telemetry\infra.xml`

On updating the Fabasoft app.telemetry server (not a clean new installation), you have to import a new license file either via the web browser client interface by opening the settings dialog and uploading a new license or by manually replacing the old license file in the file system with a new license file in the server's configuration directory and restart the server process afterwards. For details see the following chapters:

- Linux: 3.2.1 Loading Fabasoft app.telemetry License
- Microsoft Windows: 5.2.1 Loading Fabasoft app.telemetry License

### 7.1 Upgrading to Version 2015

The installation of the new Fabasoft app.telemetry 2015 requires a new product license for the product name "*Fabasoft app.telemetry 2015*" and the product version "**15**". Since product version 2015 only the major version number is relevant for the license check.

#### 7.1.1 Special Notes for Upgrades on RHEL/CentOS 6 with SELinux

If you are upgrading an old existing app.telemetry installation on a RHEL/CentOS 6 system with activated SELinux you have to fully restart all app.telemetry and Apache daemons after the installation in order to get the SELinux rules applied correctly.

#### 7.1.2 Upgrade Procedure on Linux Systems

*The following installation paths have been changed for Linux installations from older product versions than "2010 Fall Release":*

- Base installation: `/opt/app.telemetry/`
- Configuration path: `/etc/app.telemetry/`
- Data path: `/var/opt/app.telemetry/`

#### Upgrade Process for Linux Environment:

1. Backup your configuration (especially your infrastructure `infra.xml`)
2. Upgrade your app.telemetry server by means of upgrading all app.telemetry RPMs with the rpm command `rpm -U (apptelemetryagent, apptelemetryserver, apptelemetryweb)`
3. Upgrade all app.telemetry agents (`rpm -U apptelemetryagent.rpm`) - and if the app.telemetry web service was installed on some agents that package has also to be
4. Install a new app.telemetry license file
5. Access the new app.telemetry web client via the new URL:  
<http://.../apptelemetry/index.html>



*Details for the Configuration of the Software-Telemetry web service module on Linux on upgrading from older product versions than “2010 Fall Release”:*

Because the Software-Telemetry web service is now implemented as web service filter (instead of as a virtual directory) you have to remove any old Software-Telemetry module configuration directives from your custom web server configuration files.

The following configuration is loaded by default with the new Software-Telemetry web service RPM package (apptelemetryweb.rpm):

#### Apache Configuration

```
LoadModule softwaretelemetryweb_module
/opt/app.telemetry/softwaretelemetryweb/softwaretelemetryweb.so
```

This module configures the global filter for "*web.telemetry*". For more details see the default configuration file: `/etc/httpd/conf.d/softwaretelemetryweb.conf`

For any embedded Apache (as in Fabasoft Folio) remove the old `strudlwebapi.conf` configuration file and any embedded configuration directive and replace it with the new Software-Telemetry web module filter ("`LoadModule ...`" as listed above) to the embedded Apache instance.

### 7.1.3 Upgrade Procedure on Microsoft Windows Systems

*The following installation paths and registry keys have been changed for Windows installations from older product versions than “2010 Fall Release”:*

- Default installation path: `C:\Program Files\Fabasoft app.telemetry\`
- Configuration parameters in the Windows registry: `HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft app.telemetry`
- Default path for telemetry data: `<AppDataDir>\Fabasoft app.telemetry\`
  - Microsoft Windows Server 2008: `C:\ProgramData\Fabasoft app.telemetry\`

#### Upgrade Process for Microsoft Windows Environment:

1. Backup your configuration (especially your infrastructure `infra.xml`)
2. Upgrade your app.telemetry server by means of running the `setup.exe` (**important:** the `setup.exe` program performs some upgrade tasks!) file from the "Server" installation folder which will upgrade all server components (server, agent, telemetry web service)
3. Upgrade all app.telemetry agents (upgrade the software package by means of running the `setup.exe` from the "Agent" installation folder) - and if the Software-Telemetry web service was installed on some agents that package has also to be updated (via the `setup.exe` from the "TelemetryWeb" installation folder)
4. Install a new app.telemetry license file
5. Access the new app.telemetry web client via the new URL:  
<http://.../apptelemetry/index.html>

### 7.1.4 Configuration changes for Fabasoft Folio when using Software-Telemetry Web Service

After the upgrade of Fabasoft app.telemetry you must alter the Path to the Software-Telemetry web service inside the configuration object of the actual Fabasoft Folio Domain to `../tmp/web.telemetry`.

*The reasons therefor are:*

1. The new Fabasoft app.telemetry web service (since 2010 Fall Release) is implemented as global web service filter listening on URLs ending with "*web.telemetry*".
2. The web directory where the Software-Telemetry web service is run must have "*Script & Executable*" permissions.
3. All requests to the app.telemetry web service must not be handled by the Fabasoft web service handler so the Fabasoft web server "*tmp*"-folder is used which provides static content and does not handle other requests.

## 8 Fabasoft app.telemetry Architecture

### 8.1 Fabasoft app.telemetry Server

The Fabasoft app.telemetry Server consists of 4 Processes providing different Services.

#### 8.1.1 Fabasoft app.telemetry Server Service

The Fabasoft app.telemetry Server service is responsible for the Communication with all Fabasoft app.telemetry Agents, which includes the collection of counter, service-check and telemetry data. Counter and service-check data are processed to provide a system status view and are persisted on a database as configured. Telemetry data are persisted, analyzed to provide a log pool based view of requests. These request data are persisted to the database as configured.

#### 8.1.2 Fabasoft app.telemetry Worker Service

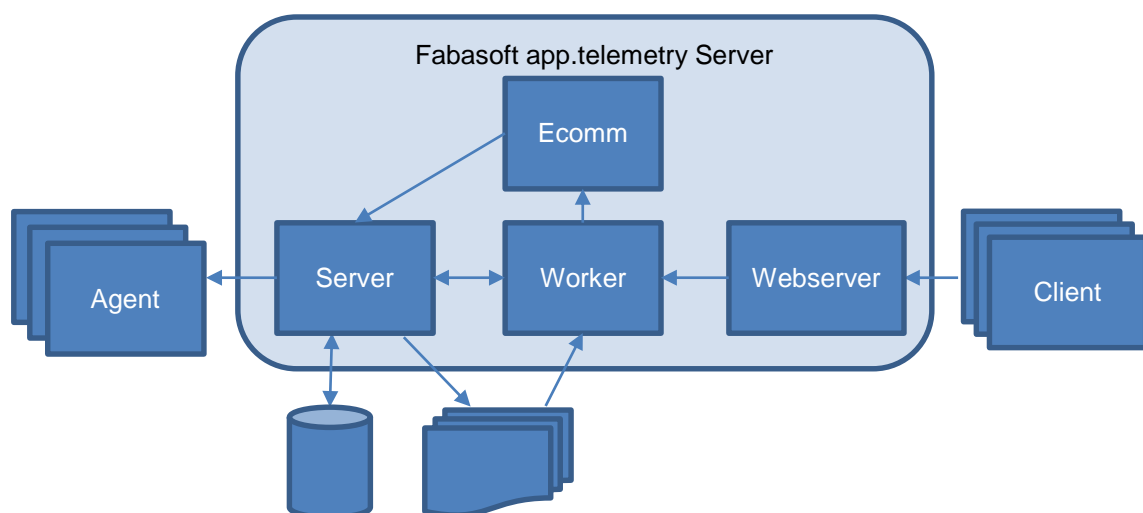
The Fabasoft app.telemetry Worker is responsible for providing Information to the Fabasoft app.telemetry Client through the Fabasoft app.telemetry Webservice. The Fabasoft Worker Service reads telemetry data from disk written by the Fabasoft app.telemetry Server and it can load requests in memory.

#### 8.1.3 Fabasoft app.telemetry Webservice

The Fabasoft app.telemetry Webservice receives the requests form the Fabasoft app.telemetry Clients and passes them to the worker for further processing.

#### 8.1.4 Fabasoft app.telemetry Ecomm Service

The Fabasoft app.telemetry Ecomm Service is responsible for handling outgoing communication requests using http/https transfer (e.g. sending a session to a service desk).



## 8.2 Fabasoft app.telemetry Agent

The Fabasoft app.telemetry Agent collects counter- and servicecheck-data and receives telemetry data from local applications. All data is buffered and streamed to the Fabasoft app.telemetry Server service on request.

## 8.3 Fabasoft app.telemetry Client

The Fabasoft app.telemetry Client connects via http/https to the Fabasoft app.telemetry Webserver. It loads static resources from the Webserver and dynamic data through JSON requests, which the Webserver forwards to the Fabasoft app.telemetry Worker.

## 8.4 Secure Communication

The communication between all the Fabasoft app.telemetry services is encrypted using TLS protocol with ciphers currently stated as secure. The communication between the Fabasoft app.telemetry Client and the Fabasoft app.telemetry Webserver uses http or https as provided by the webserver in use (Microsoft Internet Information Server or Apache Webserver). Configure http/https security as required using the webserver configuration parameters.

### 8.4.1 Certificate Management

By default, all Fabasoft app.telemetry Services create 2048 bit RSA self-signed certificates for secure communication. The certificates are stored locally as `cli_certificate.pem` and/or `srv_certificate.pem` on the machine under `/etc/app.telemetry/<servicename>` respectively `"C:\ProgramData\Fabasoft app.telemetry\<servicename>".` Validation of client certificates is being implemented by checking the sha2 hash of the client certificate against those permitted in the `trusted_certificates.cfg` file. The client certificates are automatically added to the trusted certificate of all Fabasoft app.telemetry server services.

Be careful when updating the Fabasoft app.telemetry Server service client certificate, because the `trusted_certificates.cfg` file on all agents has to be removed or rolled out, because the agent only adds the hash of the client certificate of the first server connecting to the app.telemetry Agent to the `trusted_certificates.cfg`. All connections with different certificates will be denied.

To generate new certificates, simply remove the certificate from the file system and restart the respective service.

To create a new self-signed for the use as a client certificate:

```
openssl genrsa -out key.pem 2048
openssl req -new -key key.pem -out key.csr
    ... provide organization parameters requested
openssl x509 -req -days 3650 -in key.csr -signkey key.pem -out certificate.pem
cat key.nopass.key certificate.pem > cli_certificate.pem
```

To compute the hash needed for the `trusted_certificates.cfg` file:

```
openssl x509 -fingerprint -in cli_certificate.pem -noout -sha256
```

Insert the fingerprint value at the beginning of a new line in the `trusted_certificates.cfg` file of the service that should trust that client certificate.

### 8.4.2 Trusted Certificates

When setting the `trusted_certificates.cfg` manually make sure to provide the following trusts:

Service	Trusts
Agent	Server
Server	Worker, Ecomm
Worker	Server, Webserver
Ecomm	Worker

Each service has to include the hashes of the `cli_certificates.pem` of all trusted servers in their `trusted_certificates.cfg`. For example the `server/trusted_certificates.cfg` has to include the hashes of `worker/cli_certificate.pem`, `ecomm/cli_certificate.pem` and `webserver/cli_certificate.pem`.

The default way to provide this is, that each service checks its own certificates on startup. If the certificate is absent or invalid it creates a new self-signed certificate and adds the hash of it to the `trusted_certificates.cfg` files of all required services.

The remote agents automatically trust the first service connecting to the agent by adding its client certificate hash to the new trust file. When changing the `server/cli_certificate.pem`, the `agent/trusted_certificates.cfg` has to be deleted or patched on all remote agents.

### 8.4.3 Failover/Standby configuration

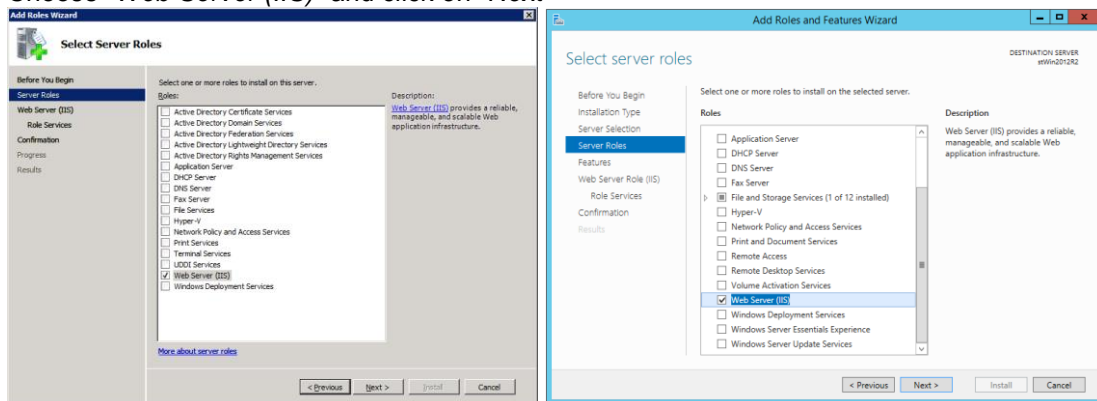
When setting up a Standby server, you have to deal with the problem, that your agents will only accept one server identified by its client certificate. So you either copy your server client certificate to the failover server or you have to distribute a `trusted_certificates.cfg` file containing all hashes of the `server/cli_certificates.pem` of all servers. The most common way to deal with this is to replace all certificates and `trusted_certificates.cfg` of the backup server with those of the original server.

## 9 Appendix

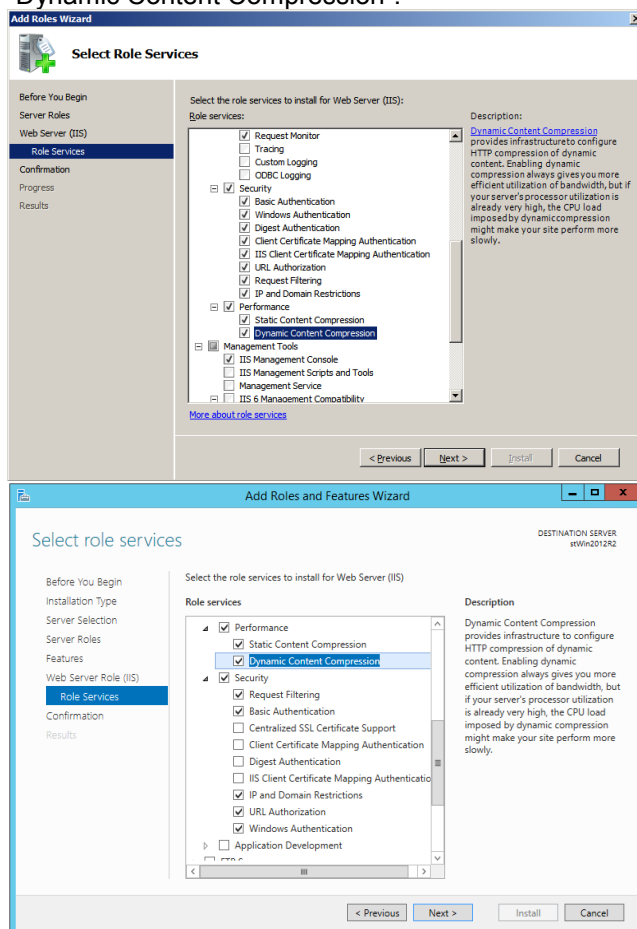
### 9.1 Installing Internet Information Services on Microsoft Windows Server 2008 / 2012

1. Install the role "*Internet Information Services (IIS)*" using the Server Manager and click on "*Add Roles*".

2. Choose “Web Server (IIS)” and click on “Next”



3. Enable all flags for Security (for details see prerequisites listed in our SPI) and set the flag for “Dynamic Content Compression”.

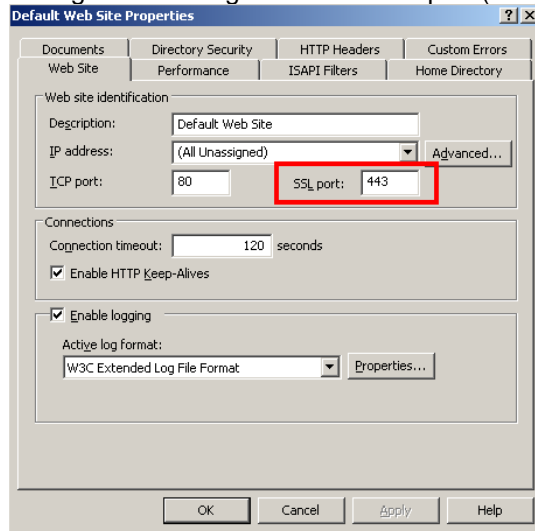


Click “Next” and finish the installation.

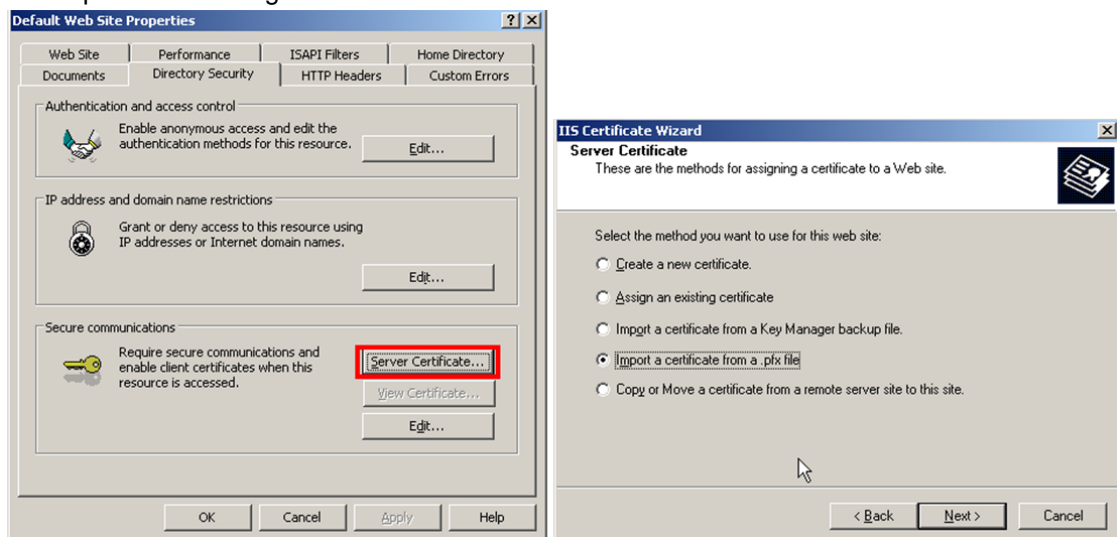
## 9.2 Enabling SSL after installation of Fabasoft app.telemetry Server

### 9.2.1 SSL Configuration on Microsoft Windows (IIS)

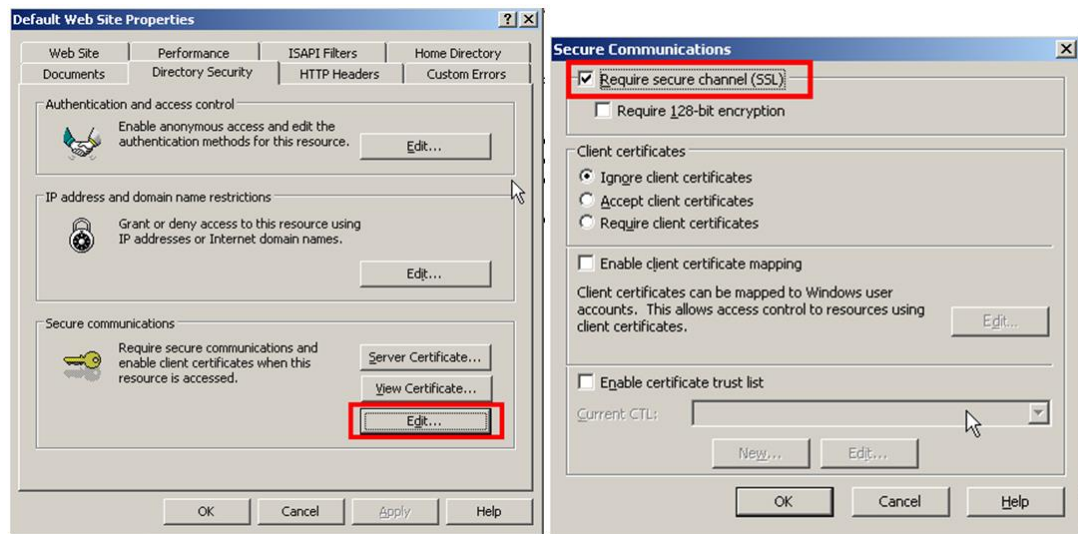
1. Switch to the „Default Web Site“-properties view and go to the “Web Site”-Tab inside your IIS-Manager and configure a valid SSL port (443).



2. Then switch to the “Directory Security”-tab and click on “Server Certificate”. In this wizard you can import/create/assign a certificate which is used for the SSL connection.



3. After doing this click on the “Edit” -Button and ensure that the checkbox “Require secure channel” is enabled.



### 9.2.2 SSL Configuration on Linux (Apache)

If you don't already have a certificate for the server you should obtain as X509 certificate file (.crt) and the appropriate key file (.key).

If it is not possible to get the certificate from a trusted certificate authority (CA) you could also generate a self-signed certificate:

#### Linux Shell Commands:

```
mkdir /etc/httpd/conf/ssl
openssl req -x509 -newkey rsa:1024 -keyout /etc/httpd/conf/ssl/server.key
-out /etc/httpd/conf/ssl/server.crt -days 365 -nodes
```

The next step is to set up the server certificate file in the Apache configuration. In normal cases there should be a default configuration file located in /etc/httpd/conf.d/ssl.conf.

#### Apache SSL Configuration (/etc/httpd/conf.d/ssl.conf)

```
LoadModule ssl_module modules/mod_ssl.so
Listen 443
...
<VirtualHost _default_:443>
    ...
    SSLEngine on
    SSLCertificateFile /etc/httpd/conf/ssl/server.crt
    SSLCertificateKeyFile /etc/httpd/conf/ssl/server.key
    ...
</VirtualHost>
```



Global trusted certificates will be accepted by the client without any warnings. If the server is secured by self-signed certificates, the client will pop up a warning dialog asking if the certificate should be trusted once or permanently.

## 9.3 Debugging and Logging

### 9.3.1 Microsoft Windows

On Microsoft Windows systems the Fabasoft app.telemetry agent and server components and the Software-Telemetry library will log important or critical events to the Microsoft Windows event log application log.

### 9.3.2 Linux

On Linux systems the Fabasoft app.telemetry agent and server components and the Software-Telemetry library will log important or critical events to the Linux syslog daemon.

The messages are logged with the application name ("*app.telemetry*") and daemon process name (apptelemetryagent or apptelemetryserver).

The default logging location is `/var/log/messages`.

The used log level can be changed for both processes in their configuration file:

- **app.telemetry agent:** `/etc/app.telemetry/agent.conf`
- **app.telemetry server:** `/etc/app.telemetry/server.conf`

The syntax for the configuration of the used log level is the following:

#### Log Level Configuration:

```
LogLevel <level>
#   Default setting is:
LogLevel LOG_WARNING
```

The default log level is `LOG_WARNING` which means that all events of this and higher levels (`ERR`, `CRIT`, `ALERT`, `EMERG`) are logged.

The available log levels are:

- `LOG_EMERG (0)`
- `LOG_ALERT (1)`
- `LOG_CRIT (2)`
- `LOG_ERR (3)`
- `LOG_WARNING (4)`
- `LOG_NOTICE (5)`
- `LOG_INFO (6)`
- `LOG_DEBUG (7)`

Additional to the logging of important process events to the syslog daemon the app.telemetry Linux processes can optionally be configured to write much more process trace output for debugging purpose. This `ProcessOutFile` can be configured in the configuration file of the desired process:

**Example configuration for apptelemetryserver:** `/etc/app.telemetry/server.conf`

```
# The ProcessOutFile property defines a file where the daemons debugging
output should be written.
# This property is disabled by default.
# The default installation contains a logrotate config for:
#   /var/log/app.telemetry/apptelemetryworkerd.log
ProcessOutFile /var/log/app.telemetry/apptelemetryworkerd.log
```

### 9.3.3 Fabasoft app.telemetry Client - Logging

When detecting any troubles using the Fabasoft app.telemetry web browser client with your desired web browser, first check if that web browser is supported by Fabasoft app.telemetry.

One step further for problem solving is using debugging tools for your web browser (e.g. Firebug for Mozilla Firefox or Google Chrome Developer Tools) and watch the debugging console output or the network requests.

## 9.4 Special Configuration Parameters

### 9.4.1 Configuration of Listening Ports for Fabasoft app.telemetry Agent/Server

If special environmental conditions require to use different listening ports for your Fabasoft app.telemetry agents (default = 10001) and server (default = 10000) change the default port numbers to your desired port numbers.

After changing the value for the Fabasoft app.telemetry agent restart the agent service (and update the agent configuration in the app.telemetry infrastructure by means of using the web client) and after changing the value for the Fabasoft app.telemetry server restart the server and the web server services.

#### 9.4.1.1 Configuration of Listening Ports on Linux

On Linux systems change the ports in the configuration files mentioned below and restart the agent/server daemons.

##### **app.telemetry Server Port Configuration (/etc/app.telemetry/server.conf)**

```
ListenPort 10000
```

##### **app.telemetry Agent Port Configuration (/etc/app.telemetry/agent.conf)**

```
ListenPort 10001
```

For the Fabasoft app.telemetry server port the Apache web server also has to be restarted after the changes have been saved.

#### 9.4.1.2 Configuration of Listening Ports on Microsoft Windows

On Microsoft Windows systems change the ports in the Microsoft Windows registry in the keys mentioned below and restart the agent/server services.

Start the Microsoft Windows Registry Editor and create the following registry keys:

### app.telemetry Registry Keys

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware app.telemetry\Agent
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware app.telemetry\Server
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware app.telemetry\Worker
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware app.telemetry\Ecomm
```

Create new *DWORD*-values with the name “ListenPort” and the desired port number (entered as decimal value!):

### app.telemetry Registry Keys for Listening Ports

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware app.telemetry\Agent\ListenPort =
10001
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware app.telemetry\Server\ListenPort =
10000
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware app.telemetry\Ecomm\ListenPort =
10003
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware app.telemetry\Worker\ListenPort =
10004
```

## 9.4.2 Configuration of Software-Telemetry Data Directory (Server)

The Fabasoftware app.telemetry server stores the Software-Telemetry data files required for deep analysis of Software-Telemetry requests (detailed data view) on the server’s hard disk.

These data files require (based on the infrastructure size and level of permanent Software-Telemetry) a medium or big amount of space on the disk. The files are stored in separate folders for each day, so that they can be deleted after some time. A later use of these files by means of import or restore from a backup is not supported by the Fabasoftware app.telemetry server.

### 9.4.2.1 Configuration of Software-Telemetry Data Directory on Linux

On Linux systems change the target path for storing the Software-Telemetry data in the server configuration file mentioned below and restart the server daemon.

#### Software-Telemetry Data Path Configuration (/etc/app.telemetry/server.conf)

```
SoftwareTelemetryDataPath /var/opt/app.telemetry/server/telemetry
```

The default location of these files/directories on Linux systems is  
/var/opt/app.telemetry/server/telemetry.

### 9.4.2.2 Configuration of Software-Telemetry Data Directory on Microsoft Windows

On Microsoft Windows systems change the target path for storing the Software-Telemetry data in the Microsoft Windows registry in the key mentioned below and restart the server service afterwards.

Create new “*String*”-value below the “*Server*” registry key with the name  
“SoftwareTelemetryDataPath” and the desired directory path:

### Software-Telemetry Data Path Registry Key

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware
app.telemetry\Server\SoftwareTelemetryDataPath = "<your Path>"
```

The default location of these files/directories on Microsoft Windows systems is located in the “*Application Data*” directory in the sub path “Fabasoftware app.telemetry\server\telemetry”.

For example on Microsoft Windows Server 2008: C:\ProgramData\Fabasoftware  
app.telemetry\server\telemetry.

### 9.4.3 Configuration of Software-Telemetry Session Export Directory (Server)

Feedback sessions or reported Software-Telemetry sessions are normally stored within the Software-Telemetry raw data therefore the session/feedback details will not be available any longer if the raw data directories of the corresponding day was deleted (e.g. if raw data cleanup after x days is configured).

The Fabasoftware app.telemetry server can be configured to export Software-Telemetry sessions automatically to a file system directory on the server’s hard disk. The responsible server setting is called “*SoftwareTelemetrySessionPath*” and can be configured depending on the target system as follows:

On Linux systems set the configuration parameter in the server configuration file as mentioned below and restart the server daemon.

### Software-Telemetry Session Path Configuration (/etc/app.telemetry/server.conf)

```
SoftwareTelemetrySessionPath /var/opt/app.telemetry/server/session-export
```

On Microsoft Windows systems add the configuration parameter as new Microsoft Windows registry key (“*String*”-value) with the desired directory path:

### Software-Telemetry Session Path Registry Key

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoftware
app.telemetry\Server\SoftwareTelemetrySessionPath = "<your Path>"
```

This feature is enabled by default using a default path beside the rawdata directory on the server (e.g. /var/opt/app.telemetry/server/sessions). You can disable this automatic session export by setting an empty value for the configuration key mentioned above.

**Note:** Software-Telemetry sessions will only exported once. Any old available sessions will be exported after the first restart of the app.telemetry server after the export directory was configured. Any new sessions will be exported after they are fully processed by the server to the directory configured at that time.

### 9.4.4 Configuration of Disk Cache and Memory Buffer for the Agent

The Fabasoftware app.telemetry agent stores the Software-Telemetry data sent by any instrumented application temporary in memory or on the hard disk (as configured) until the app.telemetry server fetches that data to process and persist it on the server’s hard disk.

When the server is not able to fetch the data (e.g. is too slow, or the network connection is down) the agent’s memory consumption will increase until a defined limit is reached. After that limit is reached the telemetry data will be written to the hard disk in a temporary folder which can also be limited with a maximum size limit.

With Fabasoft app.telemetry 2011 Spring Release an additional 3rd level cache has been implemented to provide a method to store request data on a remote medium in case the Fabasoft app.telemetry Server is not able to collect telemetry data for a longer period of time.

The following configuration parameters exist and can be used to change the default values:

- *TelemetryDiskCacheDirectory*: defines the data directory where the agent stores temporary data.
- *TelemetryDiskCacheSize*: defines the maximum disk cache size limit in megabytes (MB)
  - Minimum: 100 MB
  - Maximum: 100 GB (102400)
  - **Default:** 1 GB (1024)
- *TelemetryBufferSize*: defines the maximum agent memory buffer size limit in megabytes (MB)
  - Minimum: 20 MB
  - Maximum: 1 GB (1024)
  - **Default:** 100 MB
  - **Note:** this value is only an approximate value limiting the telemetry buffer in the agent process, the real memory consumption of the agent process may be higher (up to twice the defined limit)
- *TelemetryExtendedDiskCacheDirectory*: The directory path where the extended 3rd level cache should be stored. Default this value is not set and therefore not used.

The configuration of these parameters can be set on Microsoft Windows systems in the Windows registry and on Linux systems in the agent configuration file. A change of any parameter requires the agent process to be restarted.

#### How caches are written:

- Software-Telemetry data blocks received from individual processes are copied from the shared memory to the agent process memory and stored in a "per process" queue.
- The queuing thread fetches the data block from "per process" queues and apply compression. As compression is CPU bound this thread may utilize one CPU. On heavy data load compression may limit data throughput and agent memory consumption may raise.
- If the *TelemetryBufferSize* limit is reached, data blocks will be written to the disk cache (in *TelemetryDiskCacheDirectory* up to *TelemetryDiskCacheSize* MB).
- If the *TelemetryBufferSize* limit is reached and the disk cache is disabled or full, data blocks will be written to the extended disk cache in *TelemetryExtendedDiskCacheDirectory*. Each file is written continuously from memory cache, so the cache file size is at least *TelemetryBufferSize* MB but maximum 100MB in size.
- If the *TelemetryBufferSize* limit is reached and the disk cache is disabled or full and no extended disk cache is configured or the extended disk cache is not accessible, data will be dropped.

#### How caches are read:

- If data is stored in the standard disk cache, this data will be transferred to the server first.
- If the disk cache is empty, the content of the extended disk cache will be passed to the server file by file.
- If the disk cache is empty and no extended disk cache files are present the data will be fetched from the memory cache.

#### 9.4.4.1 Configuration of Disk Cache and Memory Buffer for the Agent on Linux

On Linux systems the temporary data settings can be modified with the agent configuration file located at `/etc/app.telemetry/agent.conf`

#### Example: Disk Cache and Memory Buffer Configuration (/etc/app.telemetry/agent.conf)

```
TelemetryDiskCacheDirectory /var/opt/app.telemetry/agent/telemetry
TelemetryDiskCacheSize 1024
TelemetryBufferSize 100
TelemetryExtendedDiskCacheDirectory /remote-NAS/temp-storage/agent-01
```

The default agent temporary data directory is /var/opt/app.telemetry/agent/telemetry.

Specify the size limits with integer numbers in megabytes (MB).

Changes of any parameter will not take effect until the agent process is restarted.

#### 9.4.4.2 Configuration of Disk Cache and Memory Buffer for the Agent on Microsoft Windows

On Microsoft Windows systems the temporary data settings can be modified in the Microsoft Windows registry in the key mentioned below.

Navigate to or create registry key:

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft app.telemetry\Agent
```

Create a new “*String*”-value for every of the following entries.

Specify the size limits with integer numbers in megabytes (MB) - (registry value of type String).

#### Example: Disk Cache and Memory Buffer Configuration Registry Keys

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft app.telemetry\Agent\
TelemetryDiskCacheDirectory = "C:\..."
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft
app.telemetry\Agent\TelemetryDiskCacheSize = "1024"
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft
app.telemetry\Agent\TelemetryBufferSize = "100"
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft
app.telemetry\Agent\TelemetryExtendedDiskCacheDirectory = "X:\remote-
NAS\temp-storage\agent-01"
```

The default agent temporary data directory is <ProgramData>/Fabasoft app.telemetry/agent/telemetry.

Changes of any parameter will not take effect until the agent process is restarted.

#### 9.4.5 Configuration of Database Rollforward Logs on the app.telemetry Server

To reduce the possibility of a data loss when the database storing request data is not available or when the Fabasoft app.telemetry Server has been stopped, a log containing all processed requests will be persisted on the file system. When the Fabasoft app.telemetry server restarts, all files will be processed to persist pending requests. Requests that cannot be written to the database will be copied into additional files, which can be reprocessed when the failure situation has been resolved.

To turn on the “*Database Rollforward Log*” you have to specify a folder, where the log files will be written to. For each log pool a subdirectory “LogFile” + database table prefix will be created. Each log file is named “logfile” + <timestamp> + “.dat” and may contain up to 10000 Requests. A log file will be deleted, when each requests has been either successfully committed to the database or has been copied to a “skipfile”. A “skipfile” is named “skipfile” + <timestamp> + “.dat”

and contains records that could not be written to the database. To retry writing records from a skipfile to the database, the "skipfile" must be renamed to "logfile" and the file will be reprocessed when the app.telemetry server will be restarted.

When “*Database Rollforward Logs*” are active, it is not necessary to keep all pending requests in memory, so the memory consumption of the app.telemetry server process in case of a slow or unavailable database will not increase so fast.

#### 9.4.5.1 Configuration of Database Rollforward Logs on Linux Systems

All configuration values can be defined in the app.telemetry server configuration file:  
`/etc/app.telemetry/server.conf`.

Define the properties and values as key-value pairs.

`SoftwareTelemetryLogfilePath`: The directory path where the directories for the log files will be created.

##### Example: Database Rollforward Log Configuration (`/etc/app.telemetry/server.conf`)

```
SoftwareTelemetryLogfilePath /var/logfiles...
```

#### 9.4.5.2 Configuration of Database Rollforward Logs on Microsoft Windows Systems

All configuration values can be defined as Microsoft Windows registry keys.

Create the desired registry value as “String”-value below the registry key  
`HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft app.telemetry\Server`

`SoftwareTelemetryLogfilePath`: The directory path where the directories for the log files will be created.

##### Example: Database Rollforward Log Configuration Registry Key

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Fabasoft  
app.telemetry\Server\SoftwareTelemetryLogfilePath = "C:\data\logs\..."
```

## 9.5 Configure Sendmail to Forward E-mails

Configure your remote SMTP server via the setting “`msh`” in `/etc/mail/submit.mc`

##### Example: `/etc/mail/submit.mc`

```
define(`confDIRECT_SUBMISSION_MODIFIERS', `C')dnl  
FEATURE(`accept_unresolvable_domains')dnl  
FEATURE(`accept_unqualified_senders')dnl  
FEATURE(`msh', `[10.20.30.40'])dnl
```

Configure your remote SMTP server via setting „`SMART_HOST`” in `/etc/mail/sendmail.mc`

##### Example: `/etc/mail/sendmail.mc`

```
define(`SMART_HOST',`10.20.30.40')  
FEATURE(`accept_unresolvable_domains')dnl
```

Recompile the sendmail configuration files (requires package `sendmail-cf`) and restart sendmail and the app.telemetry server process.

#### Linux Shell Commands:

```
make sendmail.cf
make submit.cf
/etc/init.d/sendmail restart
/etc/init.d/apptelemetryserverd restart
```

## 9.6 Fabasoft app.telemetry with SELinux

The basic support for SELinux on RHEL/CentOS 6 systems includes the following:

- The Fabasoft app.telemetry RPMs include the basic SELinux policy files.
- The files installed by Fabasoft app.telemetry RPMs are installed to `/opt/app.telemetry/selinux` and include the following file types:
  - `<process>.fc`: SELinux file context settings for `<process>`.
  - `<process>.te`: SELinux policy rules for `<process>`.
  - `<process>.if`: SELinux interface rules for `<process>`.
  - `<process>.pp`: precompiled SELinux policy module for `<process>` compiled for latest supported CentOS 6.x version (supported with Fabasoft app.telemetry server).
  - `<process>.sh`: Shell script helping to adapt SELinux policies for your environment.

The Fabasoft app.telemetry RHEL/CentOS 6 RPMs already include the basic SELinux configuration for running in a simple environment.

In some situations you may need to modify the app.telemetry SELinux policy for your systems. The following shell script may help you.

**Syntax:** `<process>.sh [ --update | --rebuild | --uninstall | --skip-build ]`

- `--skip-build`: update the SELinux policy module without rebuilding using the current policy file `/opt/app.telemetry/selinux/<process>.pp` and restore SELinux file contexts (FCs).
- `--rebuild`: rebuilding policy module based on local sources (`.fc`, `.te`, `.if`) and then reload new policy module and restore FCs.
- `--update`: check SELinux audit.log for any denies and ask to extend any missing rules. Then also rebuild, reload and restore FCs.
- `--uninstall`: uninstall the SELinux policy module for this `<process>`.

SELinux audit logs are normally logged to `/var/log/audit/audit.log`.

The Fabasoft app.telemetry server SELinux policy provides tunable *SELinux Booleans* for enabling or disabling security critical server features (default they are disabled):

- `allow_apptelemetryserver_cmdlinenotifications`: with this Boolean you can enable command line notifications for the server (running any shell script). Based on your command line notification script some more SELinux permissions may be required.

The current state of the SELinux Booleans can be checked with the command `getsebool <name>`.

To enable or disable these optional permissions use the following command: `setsebool [-P] <name>=true|false`. The optional `-P` flag makes the setting permanent (for reboots).



#### Linux Shell Commands:

```
> setsebool -P allow_apptelemetryserver_cmdlinenotifications=true
> getsebool allow_apptelemetryserver_cmdlinenotifications
allow_apptelemetryserver_cmdlinenotifications --> on
```

**Warning:** Be careful on changing any app.telemetry default settings (file system paths, network ports, etc.) to custom settings. This may break the default SELinux support and may require manual changing of the policy!

**Note:** On upgrading an existing app.telemetry installation with previously no SELinux support (<= 12.2) to a new app.telemetry installation with SELinux support (>= 12.3) you have to restart all app.telemetry daemons and the Apache daemon so that the SELinux rules are fully applied to all running services.

## 9.7 Using PostgreSQL Database for app.telemetry Server on Microsoft Windows

Since version 2013 Summer Release the app.telemetry Server also supports the PostgreSQL database on Microsoft Windows (to be able to use a free and unlimited database).

#### Installation:

1. Just install the PostgreSQL package for Microsoft Windows (Win x86-64) available from the website <http://www.postgresql.org/> on your Fabasoft app.telemetry server.
2. Add the path of the libpq.dll (e.g. "C:\Program Files\PostgreSQL\9.2\bin") to the system PATH variable.
3. Restart the app.telemetry server service.

#### Configuration:

4. Create a database and a database login role in PostgreSQL (e.g. using *pgAdmin* tool).
5. Add a "PostgreSQL Database Connection" below "Database Connections" using the edit view of the Fabasoft app.telemetry client. (The menu entry will only be available, if the Fabasoft app.telemetry server is able to successfully load the *libpq* library.)

The new database support has been tested with PostgreSQL version 9.2.4 but should work with all recent versions of PostgreSQL for Microsoft Windows (64 bit).

## 9.8 Configuration of PostgreSQL Database for app.telemetry Server on Linux

On Linux systems a PostgreSQL database can be used to persist data (like requests, Top-X reports, SLA data, etc.) of the app.telemetry server.

The Fabasoft app.telemetry server connects to the PostgreSQL database via network connection (TCP/IP), so the following PostgreSQL authentication settings are responsible for login checks:

- PostgreSQL database on Localhost: IPv4 local connections ... 127.0.0.1/32
- PostgreSQL database on remote host (remote Network IPv4)

The PostgreSQL authentication settings are managed by the *pg\_hba.conf* configuration file from the PostgreSQL data directory (normally located at: */var/lib/pgsql/data*).

There are two possibilities how to set up the PostgreSQL database authentication settings in order to use with the Fabasoft app.telemetry server:

- Trusted authentication (`trust`)
  - should only be set up for local connections - consider security issues
  - you do not have to specify a password within the app.telemetry client database connection properties
  - Test connection from the Linux shell by means of: `psql -d testdb -U testuser`
- Password authentication (`password`)
  - requires to set a password for the PostgreSQL database user
    - `ALTER ROLE testuser WITH PASSWORD 'mypwd';`
  - requires to specify the password within the app.telemetry client database connection properties
  - Test connection from the Linux shell by means of: `psql -d testdb -U testuser -W` and enter the defined password when prompted

#### Possible Problems / Misconfiguration:

- Password Authentication - but no password configured for database user
  - when using password authentication, ensure that you have set a password for your database user within the database management console (`psql`)
- Password Authentication - but no password specified for database user within app.telemetry client
- Authentication set for the wrong target
  - ensure that your authentication settings are specified for the correct network address (IPv4) - for local connections via the local network address host 127.0.0.1/32

#### Sample Configuration

- Configure PostgreSQL authentication settings by means of modifying `pg_hba.conf`:

##### Example Configuration: `/var/lib/pgsql/data/pg_hba.conf`

```
# TYPE      DATABASE      USER      CIDR-ADDRESS      METHOD
# "local" is for Unix domain socket connections only (not used with
# Fabasoft app.telemetry Server)
local      all             all               trust
# IPv4 local connections:
host       all             all             127.0.0.1/32      password
```

- Setup the database: `su postgres; psql`
  - Create a new database user/role: `CREATE ROLE testuser LOGIN;`
  - Create a new database instance: `CREATE DATABASE testdb OWNER testuser;`
  - Set the password for the database user: `ALTER ROLE testuser WITH PASSWORD 'mypwd';`
- Test database connection (you should exit from the previous `psql` session (`\q`) and from previous `su postgres` environment (`exit`))
  - `psql -d testdb -U testuser -W`
- Start the Fabasoft app.telemetry web browser client and switch to edit view
  - create a new database connection and enter the database properties
    - Database Server = localhost
    - Database Port = 5432
    - Database Login User Name = testuser

- Password for Login = mypwd
- Database = testdb
- Click the "Test" button to check the configuration

## 9.9 Installing Fabasoft app.telemetry on RHEL/CentOS 7

With Fabasoft app.telemetry 2015 we have introduced full support for RHEL/CentOS 7 so this platform can be used as app.telemetry agent platform (incl. WebAPI or Apache module) or to install the whole app.telemetry server on this platform or even to collect syslog messages by means of using the app.telemetry syslog forwarder.

**Note:** RHEL/CentOS 7 is using the *SystemD* tools to manage running services therefor other commands have to be used to start, stop and manage the app.telemetry daemons.

### 9.9.1 Start/Stop/Status of Daemons

On RHEL/CentOS 7 you have to use the `systemctl` command to start or stop a daemon or the get the current status.

#### Example daemon service commands using systemctl

```
systemctl start|status|stop <name>.service
systemctl start apptelemetryagent.service
systemctl status apptelemetryserver.service
systemctl stop apptelemetryworker.service
```

### 9.9.2 Enable Autostart of Daemons

The `systemctl` command can also be used to enable or disable automatic startup of a service.

#### Examples for enabling autostart using systemctl

```
systemctl enable|disable <name>.service
systemctl disable apptelemetryagent.service
systemctl enable apptelemetryserver.service
```

**Note:** The app.telemetry services will be enabled by default during the RPM installation.

In order to get a full list of supported `systemctl` commands see the man page (`man systemctl`).

### 9.9.3 Customize Startup Parameters (Core Dump Settings)

Similar to the daemon init scripts and their configuration files located in `/etc/sysconfig/<daemonname>` that gave you the possibility to customize startup parameters on RHEL/CentOS until version 6.x the *SystemD* tools also provide similar configuration hooks.

The default startup settings for the app.telemetry daemons are located in `/usr/lib/systemd/system/<name>.service` and define the basic startup parameters like automatic restart on failure.

Custom startup parameters can be defined by adding your own configuration file to the directory `/etc/systemd/system/<name>.service.d/`. These custom files will not be overwritten on RPM upgrades.

For example to turn on core dumps for a crashing app.telemetry agent process add following file:

```
/etc/systemd/system/apptelemetryagent.service.d/dumps.conf
```

```
[Service]  
LimitCORE=infinity
```

The Location where those core dumps are written can be customized in the global *SystemD* configuration file `/etc/sysctl.conf`:

```
/etc/sysctl.conf
```

```
# own core file pattern...  
kernel.core_pattern=/var/dumps/core.%e.%p.%h.%t
```