Knowledge Base

Fabasoft app.telemetry Knowledge Base

Fabasoft®

Contents

1 Release Notes

1.1 Fabasoft app.telemetry 2024

1.1.1 Platform Support for Red Hat Enterprise Linux 9

All Fabasoft app.telemetry Services are supported also on Red Hat Enterprise Linux 9.

1.1.2 Reloading time ranges for large requests

Large requests may not be fully loaded when their number of telemetry events exceeds the configurable limit. The parts of the request which couldn't be loaded are now shown with a gray background in the request overview and you are able to reload specific time ranges within the request. To do this, you zoom into the time range you want to reload and click on the "Reload Time Range" button.

If loading the request is exceeding a specific time limit, only the part of the request which has been loaded until then is shown. In this case a button "Continue Loading" appears in the request overview, on which you can click to continue loading the whole request.

1.1.3 Explicit passing of thread context in the C/C++ SDK

In cases where you don't want the current thread context to be used, but explicitly define the current software telemetry context for your application, you can use now the transaction handling functions available when defining `APM_EXPLICIT_TRANSACTION` in a preprocessor directive.

You use the transaction handle returned by `APMTxCreateContext` or `APMTxAttachContext` to pass to other explicit transaction functions such as `APMTxEvent` or `APMTxReleaseContext`.

1.1.4 Quick filter for specific columns

If you set the "quickfilter" flag for a string column in the Log Definition, there is an additional filter option in the Software-Telemetry and Telemetry-Research view to enable you to directly filter for this column by setting the filtered value in the corresponding field.

1.1.5 Request Views for analyzing Fabasoft Folio requests

To help you analyze Fabasoft Folio requests, there are now four new views in the Request Details showing specific information:

- In the "VAPP" tab you find a tree structure corresponding to events related to the Virtual Application module as well as the expressions executed from within your specific component.
- In the "VAPP Flame Graph" tab also the events related to the Virtual Application module and expressions are shown, visualized as a flame graph.
- In the "Expressions" tab you get an overview of the executed expressions called per component and file.
- In the "Method Calls" tab you get an overview of the executed method calls per action and object class.

1.1.6 Set root event

You can now set an event with flag "Enter" as root event by clicking on the "Set Root" button in the call stack in the Request Details. This event is then selected as new root element instead of the very first event of the request.

As a result, the call stack only consists of events executed within the call stack of the root element and also statistics, errors, expressions and method calls are shown only for telemetry data included in the root element call stack. You can reset the root event by clicking on the "Unset Root" button in the toolbar of the Request Details.

1.2 Fabasoft app.telemetry 2023 UR1

1.2.1 Hostname in Activity Statistics Agent Dimension

To account for cloud native applications deployed in a cluster, the application hostname corresponding to the pod name in Kubernetes is used as "Agent" Dimension in the Activity Statistics instead of the app.telemetry agent, in order to be able to differentiate between different pods that are all connected to the same agent.

1.2.2 ImageProxy deactivated by default

The softwaretelemetryweb module provides the ImageProxy feature, which allows a web client to download images from other websites through the server. This is required to render screenshots using JavaScript, when images are loaded from websites other than the current web pages host, because the client will decline such requests. This server feature opens a backdoor called Server Side Request Forgery (see https://owasp.org/www-community/attacks/Server_Side_Request_Forgery). Although there are some filters implemented, that restrict the usable URLs, the feature opens a channel for SSRF attacks. Therefore, the ImageProxy is now deactivated by default and if enabled the URLs can be restricted using a configurable regular expression.

1.2.3 Extended SessionInfo

The SessionInfo contains now also the number of involuntary context switches and major page faults besides user time, kernel time and process memory on Linux systems. The difference of these counters with respect to the last SessionInfo is printed out next to the absolute value. In addition, the SessionInfo is printed out at least every one second, to ensure having system information also in the case when no telemetry events are occurring.

1.2.4 Session duration

When you start a session for an application in the “Start Software-Telemetry Session” dialog window, you can now specify a duration, after which your session is automatically stopped. The duration of a session can be maximum one hour.

1.2.5 Service View

The Service View is a new view in the Fabasoft app.telemetry Client which allows a service-oriented perspective on telemetry data instead of the request-oriented perspective in the Telemetry Data or Research view. When you are analyzing a request, you can find out what happened in parallel on the service during the time the request was executed, i.e. which other requests were processed at the same time as the request you are interested in.

To open the Service View from within a Log Pool, first select a request and then click on the "Open Service View" button found at the bottom of the "Details" tab. This will open a new window with the

Service View, where all requests are loaded that were processed at the same time as the request you selected. The requests are loaded for the service which was currently selected in the "Details" tab, if you select one specific telemetry event, then the time of this event will get drawn as a red line in the Service View to help find parallel events that happened at a specific point of time.

The loaded requests that were processed by the service during the selected time period are shown in the top area of the Service View, where for each thread of the service individual blocks are drawn which correspond to blocks of telemetry data sent by the service. A request can consist of a single block or also of several blocks depending on the duration of the request, the transmitted data size and if the request was processed by several threads. The color of a block indicates the module with the highest selftime within this request section.

The blocks marked with light gray correspond to the request you selected in the Log Pool. If you click on a block, also all other blocks belonging to the same request are highlighted, and the request details are shown in the bottom area of the Service View, analogously to the Telemetry Data or Research view. You can also select several requests at once by holding the Ctrl key and clicking on the respective blocks, then the requests are shown next to each other in the "Overview" tab.

Another possibility to open the Service View is from the "Applications" Tab of a selected request, where clicking on the "Open Service View" button will load all requests processed on the selected application for the time of the selected request.

You can also persist the loaded telemetry data for a service by clicking on the "Persist" button in the Service View. This saves the data in a "Service View session" file, analogously to the Software-Telemetry sessions, which you can upload and analyze later by selecting it from the "Service View Session Files" widget in the "Applications" view.

1.2.6 Reloading ranges of large requests

When the number of events within a request is exceeding the configured limit, it can't be fully loaded and therefore only a limited number of events could be analyzed until now. You now have the possibility to reload specific time ranges within such a large, not completely loaded request by zooming into the range you want to load in the request overview and then clicking on the "Reload Time Range" button. Then the parts of the request that are not within the specified time range are unloaded again in favor of the parts within the specified range to be loaded. This enables you to at least analyze specific points of interest within large requests even if the request as a whole can't be analyzed. This works analogously for the Service View.

1.3 Fabasoft app.telemetry 2023

1.3.1 .NET 6 Software Telemetry library

For .NET 6 a native implementation of the Software Telemetry .NET API is now available as "`softwaretelemetry`" NuGet package, which also supports Software Telemetry Counter. To use the native library, set the environment variable APM_DOTNET_TRANSPORT.

1.3.2 Improvements of SNMP Counter

Enumerated integer values returned by a SNMP Counter are now mapped to their respective label. In addition, also values returning an IP address can now be evaluated and SNMP indices are read from the MIB to map the available instances correctly.

1.3.3 Open Research View from Associated Requests

If you analyze a specific request and e.g. you want to find out what the specific user additionally did at this point of time, you now can go directly from the Associated Requests of that request to the Research View by clicking on the "Open Research" button. Two minutes before and after the start

time of the request is then used as filtered time range for the Research View. The value of a column of an Associated Request can also be set as filter for the Research View by a right-click on the value and selecting "Open Research View Filtered".

1.3.4 cURL SMTP Notification Channel for Microsoft Windows

Sending mails using the curl library is now also supported on Microsoft Windows in addition to Linux.

1.3.5 Change column length in database table

If the column length of a column defined in the Log Definition is increased for an already defined Log Pool, the respective column is also altered automatically in the database table to prevent errors when writing values to the database.

1.4 Fabasoft app.telemetry 2022 UR1

1.4.1 Export all Requests from Research View

To allow the export of many requests (up to 10000) within one zip file, it is now possible to export the result of a query in the research view. Click "Download" and select "All Requests" in the download dialog. It may take several minutes to export the data, so keep the request count low by adjusting the time range and the filter criteria. Make sure to clean up the downloaded session from the software-telemetry session files list in order to free the disk memory on the server.

1.4.2 Secure TCP Transport between Agent and Library

TCP Transport between the Fabasoft app.telemetry Agent and the Library is secured by using TLS both for the Software-Telemetry C/C++ and the Java library. To secure communication, you have to set the configuration parameter "TelemetryTLSPort" in the agent configuration (10008 by default) and also configure this port on the library side correspondingly by setting the environment variable APM_TRANSPORT=tls://<agent hostname>:<port>. You can still have the agent listen in parallel on the unsecure port (10002 by default) to receive telemetry data from libraries of older versions.

1.4.3 Support SNMPv3 with User Based Security

SNMP Version 1 and 2c, supported by app.telemetry over many years, use unencrypted UDP packets to query data from servers or network nodes. SNMPv3 supports authentication and encryption of SNMP traffic.

1.4.4 Force Service Checks on Service Group

Until now only single Service Checks could be forced via right-click and selecting "Force Service Check". This is now possible also for a Service Groupe, where you then force all Service Checks belonging to this group at once.

1.4.5 Research for Requests running at a defined point in time

To be able to track long running requests, which are running at a defined point in time, you go to the Research View, enter the specific time as end time of the search interval, select a suitable start date and click the new “Running” checkbox. The query will show you all requests from the selected interval which were running at the beginning of the second specified as end time.

1.4.6 Open Subrequests in a new Window from Detail View

While analyzing subrequests originated from an application which has the “Don't Resolve Subrequests” setting selected in the according Log Pool. By clicking on the new link in the Parameter column of the GetContext event, the request will open in a new browser window. These links are not available from sessions.

1.5 Fabasoft app.telemetry 2022

1.5.1 Platform Support for Microsoft Windows Server 2022

All Fabasoft app.telemetry Services are supported also on Microsoft Windows Server 2022.

1.5.2 Open request in separate browser window

To analyze specific requests in detail, you can open the “Request Details” view visible at the bottom in the Software-Telemetry and Software-Telemetry Research View in an own browser window. A new window is opened by selecting a request and clicking on “Open in New Window” in the left sidebar. Alternatively, you can copy the link referencing to the selected request by choosing “Copy Link” in the context menu of a request. This link can then be sent to other users, so that they can directly open the corresponding request for analysis.

You can copy the link also within the “Details” tab in the “Request Details” view, to refer directly to the currently visible node within the request, or in the “Associated Requests” tab to obtain the direct link to an associated request. The corresponding symbol is located in both cases at the bottom bar.

1.5.3 Stop resolving subrequests

If you don’t want to load full requests including all associated subrequests for a Log Pool, you can set the property “Don’t Resolve Subrequests” in the Log Pool configuration, which can alternatively be set in the corresponding Log Definition as “dontResolveSubrequests” flag in the root “APMLogAnalyzer” element.

When this property is set, request contexts are only resolved down to the level of the corresponding application of this Log Pool. Associated subrequests are still shown in the “Associated Requests” tab in the Request Details pane. They can be loaded in full detail by selecting the corresponding subrequest and clicking on “Copy Link”, then this specific subrequest can be opened in a new window.

1.5.4 Application properties filter for Log Pools

You can add an additional application filter in the Log Pool configuration to filter for application properties that have been registered in the instrumentation of your application or have been set as environment variables with the prefix “APM_PROPERTY_” in your system. The filter is entered in the form of an SQL like query and can be used to configure separate Log Pools for applications with the same registration but differing in other properties, e.g. to distinguish between different systems (production and test system) with their applications otherwise identical.

1.5.5 Screenshots as attachment in feedback notification

If you want to have the screenshot of a feedback sent as attachment in the notification mail, you may check the “Add Feedback Session Attachment” property in the configuration of your SMTP Notification Account.

1.5.6 Software Telemetry Counter Charts

Counters defined via Software Telemetry can now be directly shown in your dashboard using the “Software Telemetry Counter” data source type when creating a new chart. To specify the counters for the chart, a SQL like query string is entered to filter for specific counter attributes.

1.5.7 Hide columns in the Log Definition

To hide specific columns in Log Pools, Filtered Log Pools or Log Pool Views, set the *displaypriority* for the relevant column in the Log Definition to -1. This also applies for system columns such as “Agent”, if you want to hide internal hostnames from users having access to that Log Pool.

1.6 Fabasoft app.telemetry 2021 UR 1

1.6.1 Full-text search in Research view

If you are looking for a particular text in the telemetry points within a particular time interval, you can use the new full-text search field in the Research view to identify all the requests containing this specific search string.

1.6.2 Anonymization of personal data

Specific columns containing personal data (e.g. Loginname, user group) can be automatically anonymized by setting a corresponding “anonymous” flag in the Log Definition. Additionally, the Data Anonymization has to be activated in the Log Pool Properties, where also the number of days is set after which data is anonymized.

The entries for a column to be anonymized are encoded using a random GUID which changes every day. This GUID is written to the database instead of the real entry and the information which is used to decode the GUID into the real value again is stored only for the number of days specified for the anonymization. Therefore, for personal data older than the specified number of days only the GUID can be read, enabling to still distinguish between e.g. different users, but not showing the directly related personal information any more. For younger data the real values as described in the Log Definition are shown enabling you to still read the personal data directly for this specific time period.

If the anonymization is activated for an already existing Log Pool without former anonymization, older data won’t be anonymized and new data arriving after the activation will be stored in a new table with the corresponding anonymization. Personal data contained in the raw data files, in sessions or feedback dialogs is not anonymized.

The anonymized data encoded by the GUID is also considered for the statistics accordingly.

In addition to the anonymization of columns containing personal data, such columns can be hidden altogether from specific users or groups by entering such users and groups in the Log Pool Properties next to “Hide Personal Data from User/Group”.

1.6.3 Support of multiple telemetry services in C/C++ Client Applications

Applications like Fabasoft Native Client process data in the context of multiple services each using their own app.telemetry server. The correct assignment of telemetry data to the right app.telemetry server is now supported by using a common directory per service. Initialize and close your service directories as required and associate your registered applications with the directory context, so that you can send each applications telemetry data to the correct telemetry server also in case of later data recovery processing.

1.6.4 Automatic configuration of service checks for telemetry counter

When registering a telemetry counter, you can automatically configure a corresponding service check for this counter by specifying at least one of two new attributes in the registration:

- `APM_COUNTER_ATTRIBUTE_WARNING_LIMIT`: Warning limit for the service check.
- `APM_COUNTER_ATTRIBUTE_ERROR_LIMIT`: Error limit for the service check.

1.6.5 Namespace property within Kubernetes

A Kubernetes namespace provides the scope for Pods, Services and Deployments in a Cluster. The namespace is thus an important information where a specific container belongs to. As Fabasoft app.telemetry should know where services belong to, the Fabasoft app.telemetry library attaches this namespace information to the application registration as an application property (*apm:namespace*). There is currently no documented source available, where the namespace can be read from, but if there is a file named /run/secrets/kubernetes.io/serviceaccount/namespace it contains the namespace and the library reads that file. If the file is not available, the Kubernetes Downward API should be used to provide an environment variable named “APM_NAMESPACE” containing the namespace of the pod. The Fabasoft app.telemetry Library will pass the value to the *apm:namespace* application property.

1.7 Fabasoft app.telemetry 2021

1.7.1 Platform Support for RHEL/CentOS 8

All Fabasoft app.telemetry Services are supported also on RHEL/CentOS 8.

1.7.2 OpenMetrics

The Fabasoft app.telemetry Agent can provide an http endpoint for retrieving Software-Telemetry Based Counters of all connected applications in the OpenMetrics (Draft) format, which is supported e.g. by Prometheus. Enable this feature by providing apptelemetryagent command line options openMetricsBindAddress and openMetricsBindPort.

1.7.3 “gap” column in Software-Telemetry Details

The new “gap” column denotes the time between the previous and the current telemetry event, which helps in finding delays in not instrumented parts of the code.

1.8 Fabasoft app.telemetry 2020

1.8.1 Database cleanup using Partitioned Tables for PostgreSQL

Automatic cleanup of Log Pool tables and statistics is supported on a daily basis. Since deleting records is at least as costly as the insertion of new records, the Fabasoft app.telemetry 2020 supports the automatic usage of partitioned tables in PostgreSQL (as implemented since PostgreSQL 11). Therefore, every day a new partition table will be created to hold the records of that particular day. Cleanup is performed by simply dropping outdated partitions. There is no migration of old table layout performed, so switching from a non-partitioned table layout to a partitioned table layout requires manual migration of data if necessary. An app.telemetry database connection can either use partitioning for all tables or store all the data in the traditional table layout. But you can create an additional database connection to migrate the log pools by changing the database connection assigned and optionally migrating data.

1.8.2 Linux counters from the /sys filesystem

Linux systems provide status data and performance measures in files under the /sys folder. There are a number of such counters that are not accessible using SNMP, so Fabasoft app.telemetry agents can now access the files in the /sys folder to read the respective counters.

1.8.3 Additional Functions Available in Calculated Counters

Additional functions are available calculation counter values:

- log … Natural Logarithm
- log10 .. Logarithm base 10
- sqrt … Square root

1.8.4 Grafana Support for Software-Telemetry Based Counters

Grafana is a common visualization tool used heavily in container environments. Fabasoft app.telemetry now supports JSON REST queries, that reflect the Software-Telemetry counters in a format compatible with Prometheus data sources. So you can integrate Fabasoft app.telemetry Software-Telemetry counters in your Grafana dashboards. See the REST-API documentation on the product kit for more details.

1.8.5 Support of Dimension Filters in Activity Statistics

The Sunburst visualization of the Activity Statistics data operated on the unfiltered statistics data regardless of the filters set in the Stream, Time Line or Data Grid view. Now, filters are respected and can even be extended based on the selection within the Sunburst graph. In the Stream and Time Line visualization an additional Dimension “Agent” is supported.

1.8.6 Container Support

The Fabasoft app.telemetry rpms now support installation within RHEL/CentOS 7 containers. So a Fabasoft app.telemetry server can now be hosted in a container environment e.g. on a RedHat OpenShift Container Platform.

To improve application identification in a container environment, an application is now identified by a random GUID instead of Agent Id and Process Id, which is not unique in a container environment, where the main process of each container is started with pid 1. This change requires the extension of the block header format of the software telemetry data, which makes the data incompatible with older versions of Fabasoft app.telemetry. Therefore, an older Fabasoft app.telemetry Server cannot read requests exported from a version 2020 server. And during the upgrade process, an older Fabasoft app.telemetry Server will not be able to process telemetry data received from already upgraded Fabasoft app.telemetry Agents version 2020. So the preferred upgrade sequence implies to upgrade the Fabasoft app.telemetry Server before the agents.

1.9 Fabasoft app.telemetry 2019

1.9.1 Software-Telemetry Based Counter API

The addition of a counter API based on the Fabasoft app.telemetry Software-Telemetry data transport enables applications to provide insight into their operation with minimal configuration. The Software-Telemetry based counter API also sidesteps limitations of Windows Performance Counters and SNMP by allowing for much greater flexibility to identify counter instances. Check the SDK-Documentation for more information.

1.10 Fabasoft app.telemetry 2018 UR 3

1.10.1 Extended Application Registration Properties

Applications can register additional metadata about them that empowers operators to gain a better understanding of which specific services were involved in requests. A small set of predefined metadata is automatically provided by the Software-Telemetry API libraries, check the SDK Documentation for more information.

1.11 Fabasoft app.telemetry 2018 UR 2

1.11.1 Fabasoft app.telemetry Configuration service

The new Fabasoft app.telemetry Configuration service manages app.telemetry configuration data such as:

- The Fabasoft app.telemetry Infrastructure configuration (infra.xml, encryption.key and encryption.pem)
- User settings
- The License
- Request Categories
- …

The files that represent the listed configuration data changed their location, which may require changes to the configuration of backup software to ensure backups remain usable in the future. A running Fabasoft app.telemetry Configuration Service is very important for the correct operation of all Fabasoft app.telemetry services.

1.11.2 Loading of XML and JSON files from Infrastructure Scripts

The Fabasoft app.telemetry infrastructure scripting gained functions to load and parse Log-Definitions (including forms), and JSON files into JavaScript objects that are directly useable.

1.11.3 Dynamic Java Instrumentation (JVMTI) removed

Dynamic instrumentation of Java Applications using JVMTI has been removed.

1.11.4 Authentication support for mod_auth_openidc / Keycloak

Authentication using the Apache HTTPD module mod_auth_openidc with Keycloak is now supported. With appropriate configuration the Fabasoft app.telemetry Client can also provide autocomplete support for Keycloak roles to simplify the configuration of access permissions within Fabasoft app.telemetry.

1.12 Fabasoft app.telemetry 2018

1.12.1 Software-Telemetry Research View

The Software-Telemetry Research View implemented in Fabasoft app.telemetry 2018 supports the identification of requests matching complex filter criteria and longer timespans. In contrast to the standard Software-Telemetry Data View, which has been primarily designed for live request view, the Research View handles a dataset of (up to 10000) requests which is calculated on demand and can be sorted appropriately. The calculation of the dataset is done on request, asynchronously and

can be interrupted, so the user has full control of the calculation and no more client timeouts may occur during calculation due to long running queries. The compulsory need of specifying the period of time to search in, allows to influence the duration of the query directly and helps avoiding long running queries. The analysis of single requests is available as well in the Research View as in the Data View. Navigation to the Requests from the Request Statistics view and the Request Categories view will lead to the new Research view to support the analysis of larger datasets.

1.12.2 Apache HTTPD Configuration on CentOS 7

In case you have customized the Apache HTTPD configuration for app.telemetry in /etc/httpd/conf.d/apptelemetrywebserver.conf it may be necessary to review the Fabasoft app.telemetry Webserver configuration file in /etc/httpd/conf.d/apptelemetry.conf to reapply your customizations.

1.12.3 Services need to be manually started after the installation on CentOS 7

On CentOS 7 app.telemetry services will no longer be started immediately after an installation to improve support for installations in systems that are not fully running (such as for example during an automated kickstart installation). Since you need to run the /opt/app.telemetry/bin/serversetup.sh script after an installation of a Fabasoft app.telemetry Server you will not notice a difference there. Fabasoft app.telemetry Agent installations on the other hand require a manual start of the app.telemetry Agent (or a system reboot if you prefer that).

1.13 Fabasoft app.telemetry 2016

1.13.1 Agent Counters

In order to trigger status events in case of an invalid agent time status, additional counters are available from the “Server Statistics Counter” plugin under the agent object, which reflect the values of the agent view.

The “Time Drift (ms)” counter represents the absolute value of the time difference between host machine of the app.telemetry server and the host machine of the selected app.telemetry agent in milliseconds. In some environments (e.g. when using Kerberos authentication) it is essential to keep the time drift between machines within a small range (< 5 seconds). With the new counter you can set warning or error limits to get informed when the time difference between servers is out of the valid band.

The “RTT (ms)” counter represents the time in milliseconds it takes to send a simple request to the selected agent and receive the answer. The time depends on the quality of the network connection and the load of the systems involved.

1.13.2 SSL Connection information in nginx Module

In order to track the quality of the SSL encryption, the SSL Version and SSL Cipher property is logged for SSL connections.

In addition the remote port property is reported to allow identifying http connections based on the remote port.

1.13.3 Compare Counter Checks with String Values

Some counter (e.g. from SNMP sources) report status values as strings. These strings can be matched with regular expressions to generate a warning or error status.

In the Service Check configuration dialog you can either use the **Numeric Ranges** to specify a range of critical values as you could do this also in previous versions or you use the new **Text**

Match to specify, which text values should trigger an error. The **Pattern** is a regular expression to match the value of the counter.

In this example, the counter is reported as an error, if the value is no equal to `connected` or `ok`.

The following example will generate a critical status, if the value starts with `err`.

1.13.4 Deactivate and Activate Dashboards and Charts

To temporarily avoid the usage of dashboards or dashboard charts, they can now be deactivated by an administrator using the context menu in the configuration mode.

1.13.5 Filter Notifications by source and target status

To avoid notifications, when the status changes between OK and warning, notifications can now be filtered not only by target status but also by source status.

Thus, if only *Critical* is selected as *from* and *to* status filter, then notifications are only sent, when a selected check, service or service group changes to *critical* or if it has changed from *critical* to any other status.

1.13.6 Encryption of sensitive parameters in the Infrastructure

There are several password or passphrase parameters required for service checks, database connections. On order to safely store these parameters they are now encrypted using an RSA key so they are not readable in the infra.xml. The encryption keys are generated automatically by the app.telemetry server. Make sure to create a backup of the key pair encryption.(key|pem) located under /etc/app.telemetry/server/ or C:\ProgramData\Fabasoft app.telemetry\server\ to allow the server to decrypt the messages in case of restoring the infra.xml on another system.

1.13.7 Instrumentation extended in nginx module

SSL Version and SSL Cipher were added as parameters of the nginx softwaretelemetry module.

1.13.8 Additional customization options in feedback forms

Some additional options have been implemented to allow further customization of your feedback dialogs. Choose your own fonts, define the border radius and shadow of the form and the buttons and hide the copyright text to adapt the feedback dialog to your website design.

2 Feature Details

This chapter contains technical and informational descriptions for features of Fabasoft app.telemetry not described in the “Installation Guide” or somewhere else.

2.1 Virtual Host Detection (VMware ESX)

With Fabasoft app.telemetry it is possible to get information about the location of a virtualized system which uses VMware ESX-Server virtualization technology.

Note from "*vSphere Basic System Administration Guide*": (ESX/ESXi 4)

> ESX/ESXi includes an SNMP agent embedded in hostd that can both send traps and receive polling requests such as GET requests. This agent is referred to as the embedded SNMP agent.
>
> Versions of ESX prior to ESX 4.0 included a Net-SNMP-based agent. You can continue to use this Net-SNMP-based agent in ESX 4.0 with MIBs supplied by your hardware vendor and other third-party management applications. However, to use the VMware MIB files, you must use the embedded SNMP agent.
>
> By default, the embedded SNMP agent is disabled. To enable it, you must configure it using the vSphere CLI command vicfg-snmp.

Note: VMware does not officially support SNMP (GET requests) with ESXi products but since ESXi 4 it is possible to obtain the required SNMP counters.

The Fabasoft app.telemetry virtual host detection feature requires the VMware SNMP counters provided by the embedded VMware SNMP agent.

Enable the VMware SNMP support in one of the following ways:

- using the embedded VMware SNMP agent only:
  - use the vSphere remote CLI interface and turn on the embedded VMware SNMP (using the default port and setting a SNMP community)
  - for details see the VMware guide "*vSphere Basic System Administration*"
- or using the embedded VMware SNMP agent in combination with the Net-SNMP agent
  - for details about this configuration method see the VMware technical note "*Configuring the Net-SNMP Agent on ESX Hosts*"

To turn on SNMP support for VMware ESXi 4 use the vSphere remote CLI interface and turn on the embedded VMware SNMP agent which provides the required SNMP counters for VMware products

Test SNMP Configuration for VMware ESX/ESXi 4.0:

To test the correct SNMP configuration of VMware vSphere (ESX 4.0) / ESXi 4.0 servers check if the following SNMP counters are available from a remote system (especially from the app.telemetry proxy agent used for the VMware ESX server VM host detection):

- .1.3.6.1.4.1.6876: VMware root MIB
- .1.3.6.1.4.1.6876.1: VMware product information MIB (for product name and version)
- .1.3.6.1.4.1.6876.2: VMware VM information MIB (for VM detection)

Check the existence of the SNMP counters on the VMware ESX server from your proxy agent system by means of:

```
snmpwalk -v 1 -c <your_community> <ESX_server_IP> .1.3.6.1.4.1.6876
```

```
Example: snmpwalk -v 1 -c public 10.20.30.40 .1.3.6.1.4.1.6876
```

2.2 Special Notification Channels (Command Line)

The Fabasoft app.telemetry notification system is based on a notification channel defining the way how to send a notification and several notification accounts which are notified of status changes. The following notification channel types are supported:

- **SMTP** (E-Mail server)
  - Microsoft Windows: directly via SMTP
  - Linux: indirectly via local `sendmail` process (sendmail is usually provided by postfix)
- **Command Line**: using any defined command line to send a notification

To set up the notification system correctly you have to create a notification channel first and then create sub elements of type notification account inside the notification channel.

Command Line Notification Channel:

Create a notification channel, choose "*Command Line Notification*" and define the notification command line to be executed on any status change. The command line consists of the absolute path of the command or script (on the Fabasoft app.telemetry server) and additional parameters passed to the command or script. Parameters with spaces must be quoted ("). The following variables can be used to pass concrete values to the notification command:

- **%FILE** … will be replaced with the temporary filename of the notification template filled with the current values of the current notification.
  - Example: "/tmp/app.telemetry/128920567310000000_1247583131572241.tmp"
    Note: On Linux a systemd feature called PrivateTmp is used which means that /tmp as seen by this script is not the same as /tmp visible to other services or users.
- **%TO** … will be replaced with the "TO"-address of every configured notification account.
  - Example: "0664123456789"
- **%SUBJECT** … will be replaced with the current notification subject value.
  - Example: "*Service Check \"crond availability check\" changed to ok*"
- **%AGENTHOSTNAME** … will be replaced with the hostname of the agent where the status change came from (if possible).

An example of such a command line looks like:

Example: Command Line Notification

```
/path/to/script.sh %FILE %TO %SUBJECT %AGENTHOSTNAME
```

… this will result in the following call:

```
/path/to/script.sh
  "/tmp/app.telemetry/128920567310000000_1247583131572241.tmp"
  "0664123456789"
  "Service Check \"crond availability check\" changed to ok"
  "examplehostname"
```

Set Timing Options for Notifications:

For some situations or in special installations you may need to tune some timing options for the notification system:

By default notification status is processed every 10 seconds starting 60 seconds after the app.telemetry Server service started. You may modify these settings by adding attributes to the respective *NotificationChannel* element in the `infra.xml`. To modify the notification interval, add an attribute `scheduleSeconds` with a value between 1 and 600 in seconds as the interval. To modify

the time between the service start and the first notification, add an attribute `delayOnStartSeconds` with a number of seconds to wait between 0 and 3600. These two parameters cannot be changed at runtime

Example: Notification Channel Configuration

```
<NotificationChannel id="100123" name="Mailserver" status="0" type="smtp"
delayOnStartSeconds="300" scheduleSeconds="20">

    <Param key="Authenticate" value="Anonymous"/>
    <Param key="SendEmailAddress" value="notifications@domain.com"/>
    <Param key="Server" value="10.20.30.40"/>
    <Param key="ServerPort" value="25"/>

</NotificationChannel>
```

This configuration will delay the notification processing for 5 minutes after server start (instead of 1 minute default) and process the notifications every 20 seconds (instead of default every 10 seconds).

2.3 Notification Templates

Fabasoft app.telemetry allows customization of notification templates. The default notification template files are located inside the template sub directory of the installation directory:

- Microsoft Windows: `%PROGRAMFILES%\Fabasoft app.telemetry\templates\`
  - To customize such a template copy it into the `%PROGRAMDATA%\Fabasoft app.telemetry\` directory and apply your customizations there, changes to the files in the installation directory will be overwritten on every update.
- Linux: `/opt/app.telemetry/templates/`
  - To customize such a template copy it into the `/etc/app.telemetry/` directory and apply your customizations there, changes to the files in the installation directory will be overwritten on every update.

Those template files support some substitution variables that will be replaced with the current value for that notification. These variables are written in the templates with following escape-syntax: "`<%VARIABLENAME%>`" (e.g.: `<%TIMESTAMP%>`).

Status-Change Notification Templates:

The following substitution variables exist for status-change notification templates:

- `NOTIFICATIONCLASS`: element type responsible for notification change (Service Group, Service or Service Check)
- `NAME`: name of element responsible for notification change
- `STATUS`: current status of element type responsible for notification change
- `PREVIOUS_STATUS`: previous status before this status change (status changed from `PREVIOUS_STATUS` -> to `STATUS`)
- `LOCALTIMESTAMP`: timestamp when the status change happened (in local time - time zone set on app.telemetry server)
- `TIMESTAMP`: timestamp when the status change happened (in UTC)
- `SUBNODES`: hierarchical structure of nodes affected with this status change ... for details see example below

Example: Status Change Notification Template (`commandlinetemplate.txt`)

```
<%NOTIFICATIONCLASS%> "<%NAME%>" changed to <%STATUS%>

Fabasoft app.telemetry notification Message

The status of <%NOTIFICATIONCLASS%> "<%NAME%>" changed from
<%PREVIOUS_STATUS%> to <%STATUS%>.

Date: <%LOCALTIMESTAMP%>, <%TIMESTAMP%>

<%SUBNODES%>

Reason

<%GROUP%>  Service group "<%NAME%>" reported status <%STATUS%>

<%/GROUP%><%SERVICE%>  Service "<%NAME%>" on agent <%HOSTNAME%> reported
status <%STATUS%>

<%/SERVICE%><%CHECK%>  Service Check "<%NAME%>" reported <%VALUE%>
<%MESSAGE%>

<%/CHECK%><%SERVICEPOSTFIX%>  --
```

```
<%/SERVICEPOSTFIX%><%GROUPPOSTFIX%>

---

<%/GROUPPOSTFIX%><%/SUBNODES%>
```

Escalation/Feedback Notification Templates:

The following substitution variables exist for status-change notification templates:

- `SUBJECT`: "Feedback Notification for Application *<LOGPOOL>* from *<USER>*" ... contains the user (name/email) of the user who submitted the feedback and the application/Log Pool where the feedback belongs to

Since product version 2014 Fall Release you can also customize the notification e-mail subject by means of replacing the predefined *<SUBJECT>*-tag with a custom template value consisting of raw text in combination with any desired other property value.

Here is an example based on feedback forms having a field with name “Message”:
`<title>Feedback via form <%FORMNAME%> from user <%FROM%> with message: <%PROPERTY%>Message<%/PROPERTY%></title>`

- `MESSAGE`: the message entered by the user with his feedback
- `FROM`: "name <email>" using feedback field "*name*" and field "*email*" to fill this values
- `URL`: the web page URL the feedback was sent from
- `FILTER`: session filter that was used for the feedback session
- `LOCALTIMESTAMP`: timestamp when feedback was submitted (in local time - time zone set on app.telemetry server)
- `TIMESTAMP`: timestamp when feedback was submitted (in UTC)
- `LOGPOOLNAME`: name of the Log Pool/application where the feedback belongs to
- `LOGPOOLID`: internal ID of the Log Pool/application where the feedback belongs to
- `SESSIONID`: session ID of this feedback
- `APPLICATIONNAME`: appname of registered application which triggered the feedback session
- `APPLICATIONID`: appid of registered application which triggered the feedback session
- `APPLICATIONTIER`: apptiername of registered application which triggered the feedback session
- `FORMNAME`: name of feedback form which triggered the notification
- `SESSION_PROPERTY_LIST`: a list of all additional feedback property fields
- `PROPERTY`: to obtain the value of any desired session property using *<%PROPERTY%>property-name<%/PROPERTY%>*
- `ADD_FILES_AS_ATTACHMENTS`: adds all in the feedback containing files (screenshot, systeminfo) as e-mail attachment to the mail notification. The location of this tag in the template is not relevant as it is not inlined but added as attachment.

Example: Escalation/Feedback Notification Template (`escalationmailtemplate.html`)

```
<html>
  <head>
    <meta name="generator" content="Fabasoft app.telemetry" />

    <title><%SUBJECT%></title>

    <style type="text/css">
      BODY { font-family: Arial; font-size: 10pt; background-color:
#F5F5F5; }
```

```
          h1 { font-size: 13pt }
          h2 { font-size: 11pt; margin-bottom: 0.3em; }
      div, a { margin-left : 10px; }
    </style>

  </head>

  <body>

    <h1>Fabasoft app.telemetry feedback notification</h1>

    <h2>Message</h2>
    <div><%MESSAGE%></div>

    <h2>Date</h2>
    <div><%LOCALTIMESTAMP%></div>
    <div><%TIMESTAMP%></div>

    <h2>Application</h2>
    <div><b>Logpool: </b><%LOGPOOLNAME%></div>
    <div><b>Application Tags: </b><%APPLICATIONNAME%> - <%APPLICATIONID%>
- <%APPLICATIONTIER%></div>

    <h2>Sent by</h2>
    <div><%FROM%></div>

    <h2>Sent from</h2>
    <div><b>URL: </b><%URL%></div>
    <div><b>Filter: </b><%FILTER%></div>

    <h2>Feedback Infos</h2>
    <div><%SESSION_PROPERTY_LIST%></div>

    <h2>Open/View Feedback Session</h2>
    <div>Click the link below to view more details of this feedback in
your Fabasoft app.telemetry client:</div>
    <a href="<%CLIENTURL%>?feedback-
view&amp;formid=<%FORMLOGPOOLID%>&amp;sessionid=<%SESSIONID%>">Open
Feedback Session</a><br/>

  </body>
</html>

<%ADD_FILES_AS_ATTACHMENTS%>
```

2.4 Fabasoft Folio Object Address Resolution

Fabasoft Folio object addresses (a.k.a. *COO-addresses*) are exact but quite meaningless in respect to the character of the object they represent. It is mainly for the sake of optimization that the Fabasoft app.telemetry instrumentation of Fabasoft Folio uses the 64-bit integer representation of the addresses to pass object identity information. Whereas the conversion to the "COO-Address" format has been coded into Fabasoft app.telemetry, a more user friendly way of presenting Fabasoft Folio objects is still available.

Mapping of addresses to Names and References:

Providing an XML file containing a mapping from object address to names or references Fabasoft app.telemetry can represent Fabasoft Folio addresses in human readable format to help users to interpret recorded request information more easily.

Generating the mapping file:

In order to generate the mapping file the "*Integration for app.telemetry Software-Telemetry*" Software Component provides the XSL Transformation file `FSCAPPTELEMETRY@1.1001:GenerateLogAnalyzerData`. Calling this XSL Transformation by a script or by a Fabasoft Expression you receive an XML file containing the addresses and names of the following object classes:

- Software Product
- Software Component
- Component Object (and all derived object classes)
- User
- Group
- Domain

To generate the mapping file start a command line (`cmd.exe` (Microsoft Windows) or `bash` (Linux)) on a Fabasoft Folio Server of your domain with a Fabasoft Folio service user account having permissions to read all objects, set the `HOST` and `PORT` variable to point to the Fabasoft Folio backend service and execute the following command (call the `fsceval` command in one line):

On Linux systems the default service user account is `fscsrv` and the default port of the Fabasoft Folio backend service is *18070*. The `fsceval` binary is located under `/opt/fabasoft/bin/` but should already be available via the PATH-variable without an absolute path.

Run `fsceval` (on Linux) to generate address resolution mapping file.

```
su - fscsrv

HOST=localhost PORT=18070 fsceval -eval
"coouser.COOXML@1.1:XSLTransformObject(coouser,
FSCAPPTELEMETRY@1.1001:GenerateLogAnalyzerData, 'fscdata.xml')"
```

On Microsoft Windows systems you should be logged in with an administrative account (of Fabasoft Folio). Setting the `HOST` (default: *localhost*) and `PORT` (default: *18070*) environment variables is optional and not required for a default installation.

Run `fsceval` (on Microsoft Windows) to generate address resolution mapping file.

```
fsceval.exe -eval "coouser.COOXML@1.1:XSLTransformObject(coouser,
FSCAPPTELEMETRY@1.1001:GenerateLogAnalyzerData, 'fscdata.xml')"
```

Note: In earlier Fabasoft Folio or Fabasoft eGov-Suite installations the component name was `FSCAPPLSTRUDL@1.1001` instead of `FSCAPPTELEMETRY@1.1001`.

The result is a generated fscdata.xml for your domain:

Syntax of `fscdata.xml` mapping files

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Objects>
    <Object id="COO.1.1.1.2500" name="Fabasoft Folio/Base"
reference="ComponentBase@1.1"/>
    <Object id="COO.1.1.1.9285" name="Fabasoft Folio/Folio"
reference="ComponentFolio@1.1"/>
    ...
</Objects>
```

In earlier versions of the XSL-transformation script shipped with the app.telemetry Software Component some more additional properties (`<Attributes>`-sublist and `<Methods>`-sublist below the `<Object>`-tags) have been generated which are not used for the address resolution and name mapping and can be skipped. The only required entries in the mapping file for name- and reference-resolution are the `<Object …/>`-tags. To remove those not needed old sublist elements you can use grep to exclude all `<Attributes>`-sublist and `<Methods>`-sublist entries:

Exclude not used attributes from mapping file (optional)

```
grep -v "<Attribute" fscdata.xml | grep -v "</Attribute" | grep -v
"<Method"  | grep -v "</Method" > fscdata-small.xml
```

Set up Fabasoft Folio address resolution for Fabasoft app.telemetry:

The Fabasoft app.telemetry web browser client receives the formatted values from the Fabasoft app.telemetry web service, which is therefore responsible for the formatting of the addresses. This implies that the mapping file has to be stored in the configuration folder on the web service under the following path:

- Linux: `/etc/app.telemetry/fscdata.xml`
  (ensure that the apptelemetryworker user can access/read the file)
- Microsoft Windows: `%PROGRAMDATA%\Fabasoft app.telemetry\fscdata.xml`
  (ensure that the app.telemetry Worker user can access/read the file)

Restart the Fabasoft app.telemetry Worker service to read the new content of `fscdata.xml`.

Since Version 19.1 you can upload a new fscdata.xml file in the Fabasoft app.telemetry web browser client using the “Upload Mapping” action in the Application view. The existing fscdata.xml will be replaced and immediately applied without a restart of the Fabasoft app.telemetry Worker service.

Since Version 19.1 the upload of the fscdata.xml is also supported using an HTTP POST request.

Upload fscdata.xml using curl

```
curl -u username:password --header "content-type: application/xml" --url
http://localhost/apptelemetry/server/UploadMapping --data-binary
@fscdata.xml
```

The decision whether the COO-address is mapped to the objects name or reference is defined in the log definition of the corresponding log pool by the column “format” with the values:

- `format="fsc:address"` … pretty-print the COO-address
- `format="fsc:reference"` … print the reference value of the object if found in mapping file (fall-back: COO-address)
- `format="fsc:name"` … print the name value of the object if found in mapping file (fall-back: COO-address)

2.4.1 Merging of Multiple Mapping Files

In special situations one Fabasoft app.telemetry server may be used to monitor multiple Fabasoft Folio domains (e.g. test domain and production domain). Currently the app.telemetry server only supports one global address resolution mapping file (as described in the main chapter).

The solution to get Fabasoft Folio object addresses of different domains resolved together is to merge the separate mapping files (generated for each Fabasoft Folio domain) into one single mapping file.

1. Generate the separate mapping files for each Fabasoft Folio domain
   1.1. Resulting in several `fscdata.xml` files (e.g.: `fscdata1.xml, fscdata2.xml, fscdata3.xml, fscdata4.xml`)
2. Remove the basic XML-file header (line 1) and surrounding `<Objects>`-list XML-tags (line 2 and last line)
   2.1. First file (`fscdata1.xml`): remove the end tag `</Objects>` from last line of the first file
   2.2. File 2 … <n-1> (`fscdata2.xml, fscdata3.xml`): remove XML-header and `<Objects>`-start-tag and </Objects>-end-tag
   2.3. File n (`fscdata4.xml`): from the last file only remove the XML-header and `<Objects>`-start-tag
3. Concatenate all files in the same order: `”cat fscdata1.xml fscdata2.xml fscdata3.xml fscdata4.xml > fscdata.xml”`

Or just copy all plain `<Object>`-entries without any surrounding container-tags into 1 single file with the syntax shown in the following example:

Syntax of `fscdata.xml` mapping files

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Objects>
    <Object id="COO.1.1.1.2500" name="Fabasoft Folio/Base"
reference="ComponentBase@1.1"/>
    <Object id="COO.1.1.1.9285" name="Fabasoft Folio/Folio"
reference="ComponentFolio@1.1"/>
    ...
</Objects>
```

Note: Be careful with the file encoding – ensure to edit and save the file with valid encoding (UTF-8).

Duplicate `<Object>`-mapping definitions may occur in the merged file but doesn’t matter. The first `<Object>`-definition for an id (*COO-address*) is used to resolve the entry.

2.4.2 Upload Mapping Files with Command Line

Since Version 2022 UR2 the command line supports uploading or merging the fscdata.xml file with the following command:

Upload Mapping file using Command Line

```
apptelemetry mapping upload fscdata.xml
```

or

```
apptelemetry mapping merge fscdata.xml
```

When merging mappings, the entries will replace existing entries and the merged fscdata.xml file will be updated on the server.

2.4.3 Download fscdata.xml from Folio Webservice

With 2022 November Release Folio provides a web service URL to download the fscdata.xml form a Folio webservice:

Download fscdata.xml from Folio Web Service

```
curl -u username:password -o fscdata.xml \
https://folio.mycompany.com/folio/apmgeneratemapping/fscdata
```

Incremental update can be a acquired passing a valid timestamp as an optional `changedat` parameter as in:

Download incremental fscdata.xml from Folio Web Service

```
curl -u username:password -o fscdata.xml \
https://folio.mycompany.com/folio/apmgeneratemapping/fscdata?changedat=2022-10-22
```

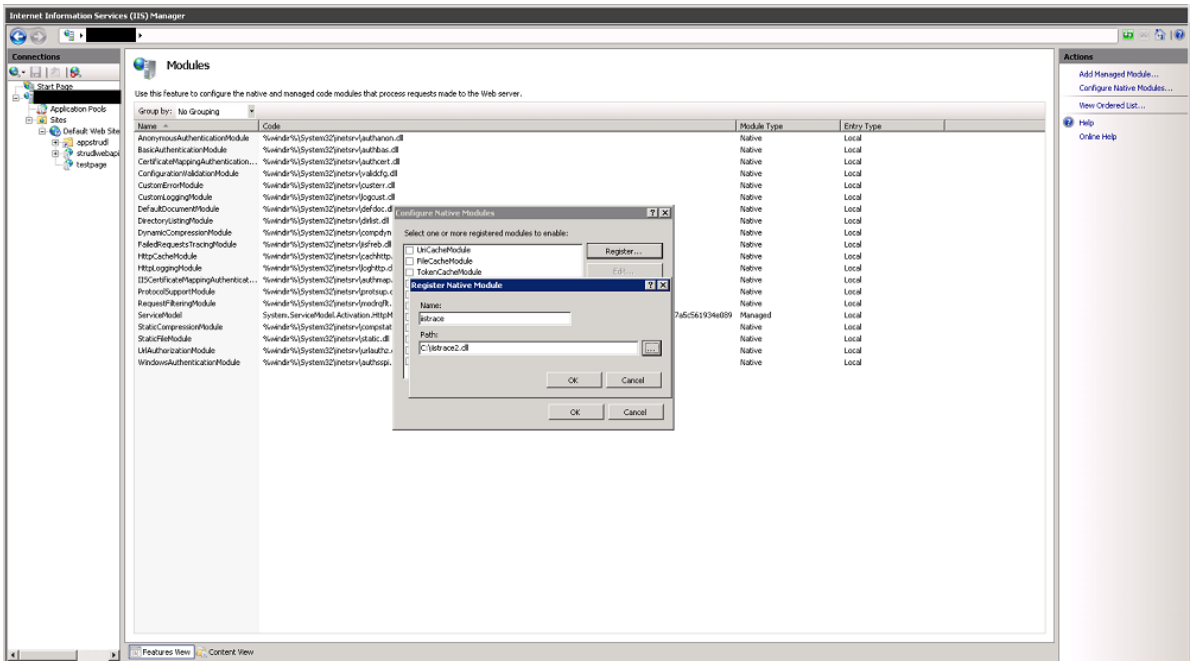
2.5 Configure Software-Telemetry Module for Microsoft IIS

With the native Software-Telemetry module for Internet Information Services (IIS) Fabasoft app.telemetry can log each HTTP-request with some important parameters to a separate Software-Telemetry log pool or the data is shown as extension module in the requests of an involved log pool of another web application. The module shows the start-, and end-time of each request, the time needed for authorization and execution and additional request parameters

Configuration:

To install the module on Microsoft IIS7 web server follow these steps:

1. Get the module files from the app.telemetry installation media from the directory "`Telemetry-Modules\WINDOWS_Xxx`"
2. Place the `telemetryiis7.dll` in any folder on your file system and give it the needed permission settings (the configured user of the web applications application pool must have read and execute access to the file).
3. Open IIS Manager and move to the root node of your web server
4. Select and enter the "*Modules*" feature.
5. Click on "*Configure native module*" inside the action pane on the right edge of IIS manager.
6. Click on "*Register*" in the new configuration dialog.
7. Fill in a custom name and the path of the `.dll` file into the required fields and commit by clicking OK.
8. Disable the module on the root level.



Enable the module for your web application:

1. Navigate to your web application in IIS manager and enter the "*Modules*" feature.
2. Click on "*Configure native module*" inside the action pane on the right edge of IIS manager.
3. Enable the module in the configuration dialog.

Note: You can enable the module only for your web application or for the complete web site, but be careful not to enable the module on two configuration layers because this will lead to a duplicate error.

Enable context transitions for Browser-Telemetry:

The IIS-Software-Telemetry module supports End-2-End Software-Telemetry by providing a context in a session-cookie. Use the following steps to enable this feature:

1. Configure the IIS schema
   1.1. stop IIS (iisreset /stop)
   1.2. copy `apmmodule_schema.xml` from the "`Telemetry-Modules\WINDOWS_Xxx`" directory to `%windir%\system32\inetsrv\config\schema`
   1.3. edit `%windir%\system32\inetsrv\config\applicationHost.config` to include the "`apmModule`" section under the "`system.webServer`" sectionGroup:
   ```
   <sectionGroup name="system.webServer">
         <section name="apmModule" overrideModeDefault="Allow" />
   …
   ```
   1.4. start IIS (iisreset /start)
2. Use Internet Information Service (IIS) Manager to configure the Fabasoft app.telemetry IIS module
   2.1. select the respective web site or virtual directory where you want to enable context transitions (you should enable this feature on virtual directories providing instrumented html files)
   2.2. double-click "*Configuration Editor*" from the "*Features View*"
   2.3. select "`system.webServer/apmModule`" in the "*Section*" property
   2.4. change the value of "*provideContext*" to "*True*"
   2.5. click "*Apply*"

2.6 Generic Log Definition Columns

Since Fabasoft app.telemetry 2012 Spring Release log pools and log definition columns have been extended to be more powerful and flexible than before.

Log definition columns can be defined as generic/dynamic columns based on other columns obtaining their value by means of evaluating a calculation formula.

Possible calculation types are:

- **categorize** value
- **split** value (regular expression)

Categorize Value

To categorize an existing log definition column value, decide which parent/base column you want to split up into reasonable value categories. This column is defined in the new column definition as `parent`-attribute containing the name of the chosen existing column. You can also choose internal columns like the duration column.

The next step is to define the split points how to separate the value into the categories using the `calculation`-attribute. If you define 3 split points, you will get 4 categories: below the 1st split point, between 1st and 2nd, between 2nd and 3rd and above the 3 split point.

Then you can define textual labels for the categories using the format-attribute containing the keyword "`enum:`" followed by the number of the category, a colon (`:`) and the label text separated with a semi-colon (`;`) from the next category.

Last but not least set the flags of the new defined column including the flag for a `CALC_VALUE_CATEGORY = 0x10000` (decimal=65536). If you want to define the column to be a dimension-column you also have to include that flag (0x0100 / decimal=256).

Syntax for Categorize Value

```
calculation="x1;x2;x3"
format="enum:1:0-x1;2:x1-x2;3:x2-x3;4:&gt;x3"
parent="name of parent column"
name="Category Column Label"
flags="65792"
```

Split Value (Regular Expression)

To split an existing log definition column value into sub parts, decide which parent/base column you want to split up. This column is defined in the new column definition as `parent`-attribute containing the name of the chosen existing column.

The next step is to define the regular expression splitting up the existing string into a new value using the `calculation`-attribute:

- Enclose the desired text part in the regex with grouping braces "`( … )`"
- Define the desired group match number with the `paramid`-attribute (first/only 1 matching group ... `paramid="1"`).

Last but not least set the flags of the new defined column including the flag for a `CALC_REGEXP = 0x20000` (decimal=131072). If you want to define the column to be a dimension-column you also have to include that flag (0x0100 / decimal=256).

Example: Web Timing Dynamic Columns

```
<APMLogAnalyzerEntry  name="Response Time Category"  parent="duration"
  calculation="10000000;40000000;100000000;300000000;600000000"
  format="enum:1:0-1s;2:1-4s;3:4-10s;4:10-30s;5:30-60s;6:&gt;60s"
  flags="65792"  paramid="0"  eventid="0"  module=""
  columndatatype="1" displaypriority="0" displaywidth="50"
tablename="browser" tabletype="1" type="1"/>


<APMLogAnalyzerEntry  name="Protocol"  parent="Page URL (referer)"
  calculation="([\w]+):"  paramid="1"  flags="131328"  eventid="0"
  module=""  columndatatype="3" columnlength="100"
  displaypriority="0" displaywidth="50"  tablename="browser"
tabletype="1" type="3"/>


<APMLogAnalyzerEntry  name="URL Path"  parent="Page URL (referer)"
  calculation="[\w]+://[^/]*([^\?]+)"  paramid="1"  flags="131328"
  eventid="0"  module=""  columndatatype="3" columnlength="200"
  displaypriority="0" displaywidth="120"  tablename="browser"
tabletype="1" type="3"/>
```

2.7 Log Statistics (Daily Statistics)

Since Fabasoft app.telemetry 2012 Spring Release a new database-based statistic feature is available which allows you calculate defined statistics on the available data at a defined time.

For example you can calculate and summarize all the requests from the last day every night and generate significant statistic charts.

Feature Details:

The new log statistics are based on the following components:

- An application *log pool* defining which data is recorded.
- A *database* selected in the log definition storing the basic data (full base data set) and the statistic tables.
- A *log statistic* defining which data is calculated at which time in which way.
- A *chart* based on that log statistic representing the summarized/calculated data.

The *Log Statistic* object defines the following key facts:

- A *time resolution* (in minutes) defining the interval when the statistic is calculated.
- A *time schedule* defining the delayed offset (in minutes) after the interval when the statistic is calculated
- A database table extension (*tableext*) that is appended to the new created statistic database tables (belonging to the database assigned in the log pool). This value must be unique for the statistics selected for a single logpool.
- A list of dimension fields. The statistic result may be grouped by the given dimensions (see "*group by*" in the chart).
- Optional *database filter* defining which requests should not be included for the statistic calculation (the filter has to be defined in SQL syntax, e.g. "`duration < 600000000`")

The Top-X Logpool Statistic Chart defines the following key facts:

- The *log pool* where the base data is taken from
- The *log statistics* where the calculated chart data is taken from
- A *time range* defining how many time intervals will be calculated
- *group by* field. Select the dimension from the Log Statistic dimension field by which to group the values by.
- Optional *database filter* defining which requests should not be included for that chart (the filter has to be defined in SQL syntax and may only cover dimensions included in the Log Statistic)
- A list of displayed *measures* defining which measures will be shown in the chart
- In Chart Data Mode you can specify the time column format (“date”, “time”, “datetime”) if the chart type is “Table”, use “summary” to get a single summary record.

Configuration Details

The basic configuration can be done via the app.telemetry client (GUI) interface on the edit view.

1. Check if your application log pool has a database assigned
2. Create a new log statistics object and define all required fields
3. Create a new chart - select "*Top-X Logpool Statistics*" as data source - and define all required fields

2.8 TCP Transport for Agent-Library Communications

Since Fabasoft app.telemetry 2012 Summer Release a new transport channel for app.telemetry agent/library communication is available.

Before this feature was introduced telemetry data could only be sent to the app.telemetry agent via a native library using shared-memory communication which limits application instrumentation on the supported platforms of the app.telemetry agent.

In order to extend the support for other platforms (for application instrumentation) we have introduced the TCP transport channel which can be used to transport the telemetry data from any Java platform (also with other hardware architecture).

Feature Details:

TCP transport channel is available for following app.telemetry libraries:

- Software-Telemetry C/C++ library (on supported app.telemetry Agent platforms)
- Software-Telemetry Java library (on any Java 8 or higher platform)

Configuration Details app.telemetry Agent

First of all you have to enable the TCP transport channel for any app.telemetry Agent in your infrastructure (default the agent does not listen for any TCP data).

Define the network port the agent should listen on for telemetry data in the app.telemetry agent configuration:

- Linux: `/etc/app.telemetry/agent.conf`: "`TelemetryPort 10002`"
- Microsoft Windows: Registry Key: `\HKLM\SOFTWARE\Fabasoft app.telemetry\Agent\TelemetryPort = 10002` (DWORD - decimal)

Restart the app.telemetry agent daemon/service.

Configuration Details for C/C++ Library

In order to tell the native C/C++ Software-Telemetry library to communicate via TCP transport (instead of shared memory) with an app.telemetry agent, start the instrumented application with the following environment variable:

- `APM_TRANSPORT=tcp://<agent-IP>:<port>`
  - for example: `APM_TRANSPORT=tcp://localhost:10002`

Configuration Details for Java Library

In order to tell the Java Software-Telemetry library to communicate via TCP transport (instead of communicating with the native library on the local system) with an app.telemetry agent, start the instrumented application with the following configuration parameters:

- either as Java system property at the startup command line of your instrumented application:
  - `-Dcom.apptelemetry.apm.transport=tcp://<agent-IP>:<port>`
- or as environment variable:
  - `APM_TRANSPORT=tcp://<agent-IP>:<port>`
- or as app.telemetry config property set via the app.telemetry Config tool:
  - `java -jar softwaretelemetry.jar`
  - to get basic help how to use this config tool call it with the param "`help`"
    - `java -jar softwaretelemetry.jar help`
  - to get more extensive help about the possible configuration parameters call it with the param "`info`"

    - `java -jar softwaretelemetry.jar info`
  - setting the transport as config property:
    - `java -jar softwaretelemetry.jar set configs/<myconfig> transport tcp://<agent-IP>:<port>`
  - and start the application with the config name with the Java property:
    - `-Dcom.apptelemetry.apm.config=<myconfig>`
  - or if using JVMTI start the application with the `javaagent` and the config name:
    - `-javaagent:softwaretelemetry.jar=config=<myconfig>`

Optionally you can define a log file for debugging purpose as Java system property:

- `-Dcom.apptelemetry.apm.logfile=<logfile-name.log>`
- `-Dcom.apptelemetry.apm.loglevel=debug|trace|default`
  - `debug` will cause many debug log messages
  - `trace` will result in a huge amount of trace messages where almost everything is logged (for developer)
  - any other *loglevel* value will result in normal logging (info, warning, error)

2.9 File System Counter Checks for Fabasoft Folio Internal Data

Fabasoft app.telemetry counter checks are a very powerful possibility to monitor arbitrary values of different (foreign) systems. One of those (foreign) systems (from the app.telemetry point of view) is Fabasoft Folio and the internal data structures.

With shell scripts you have still the possibility to obtain some internal data from Fabasoft Folio and with app.telemetry counter checks you can monitor those values obtained by some scripts writing the results into text files.

Some of the internal data of Fabasoft Folio can be obtained by executing the utility program `fsceval` on a Fabasoft Folio backend server (running as Fabasoft Folio service user).

1. Write your Fabasoft Folio Kernel expression printing out the desired value at the end of the script into a text file `<getvaluescript.exp>`.
2. Test your expression script with the tool `fsceval` running on a Fabasoft Folio backend server on behalf of a Folio service user (fscsrv): `fsceval -nologo -file <getvaluescript.exp>`.
   You may have to specify the Fabasoft Folio host (`HOST=localhost`) and the port of the backend service (`PORT=18070`).
   The script execution should print the value defined in the expression file.
3. "Grep" for the resulting value in the output and store the plain value into a text-file stored in the app.telemetry status-file folder readable for the app.telemetry agent service user:
   `fsceval -nologo -file <getvaluescript.exp> | grep value | awk '{print substr($3, 2, length($3) - 2)}' > /var/opt/app.telemetry/status/<fsc-value.txt>`.
4. Create a new Fabasoft app.telemetry service check of type "Counter check using file system" and select the status file defined in your scripts. You should also harmonize the update interval of your file counter (`cron` job) with the check interval.

Note: Running these scripts as cron job may require some special environment handling:

- Ensure that the tool `fsceval` is in the `PATH` of the cron job script or call it with full path (`/opt/fabasoft/bin/fsceval`).
- Ensure that the LD_LIBRARY_PATH is set correctly
  `LD_LIBRARY_PATH=/opt/app.telemetry/lib64:/opt/fabasoft/share/eval:/opt/fabasoft/share/eval/INSTALLDIR/Domain_1_1:/opt/fabasoft/share/eval/INSTALLDIR/Domain_1_1001`
- The expression file <`getvaluescript.exp`> has also to be passed with absolute path.

This may sound a little bit complex but the following two examples will help you understand and use this powerful feature:

2.9.1 Check Remaining Object Addresses in a Fabasoft Folio COO-Store

To monitor the count of free object addresses in a Fabasoft Folio COO-Store you may use the following expression and scripts:

1. Write the expression and save it to a file (objinfo.exp).

`objinfo.exp`: Expression for free addresses in COO-Store

```
// Check all stores and write free addresses to a file per store


// specify target path here
@writenumbertopath = "/var/opt/app.telemetry/status/";
```

```
@svcs = coort.SearchLocalObjects3(cootx, "COOService");
@objremaining = 0;
for (@i = 0; @i < count(@svcs); @i++) {
  @Storelist = @svcs[@i].coosrvinfo.cooinfmaxobjids;
  for (@j = 0; @j < count(@Storelist); @j++) {
    @Storeagg = @Storelist[@j];
    if (@Storeagg.cooinfmaxcoost.objclass == COO.1.1.1.440) {
      @objremaining = @Storeagg.cooinfavailobjids;
        @cont = coort.CreateContent();
        @cont.SetContent(cootx, 1, 65001, @objremaining);
        @cont.GetFile(@writenumbertopath + "freeids_" +
@Storeagg.cooinfmaxcoost.objname + ".txt");
    }
  }
}
```

2. Write a script to get and update the value, save it as shell script and test it (running as Folio service user on a backend server).

Test expression using `fsceval`

```
su - fscsrv
HOST=localhost
PORT=18070
fsceval -nologo -file objinfo.exp
```

3. Create new app.telemetry counter checks for each COO-Store to monitor the value from the status files and define the update interval and the warning/critical limits (for example: set a warning level for below 1000000 and a critical limit for below 100000) to be notified when the COO-Store is low on free object addresses.

2.9.2 Check Remaining Days until Fabasoft Folio License Expires

In order to get notified before your Fabasoft Folio license expires just follow this example.

1. Write an expression like the following to get the days until your license will expire and save it to a file (`fsclicense.exp`).

`fsclicense.exp`: Expression to check days until license expires

```
@lics = coort.GetCurrentDomain().COOSWCLM@1.1:domainlicenses;
@expiryday = 1000;
for (@i = 0; @i < count(@lics); @i++) {
  @lic = @lics[@i];
  if (@lic.COOSWCLM@1.1:keyexpirydate) {
    @exp = (@lic.COOSWCLM@1.1:keyexpirydate -
```

```
coort.GetCurrentDateTime(coouser)) / 3600 / 24;

    if (@exp < @expiryday) {
      @expiryday = @exp;
    }
  }
}

@expiryday;
```

Warning: In some situations you may not rely on the accuracy of the `COOSWCLM@1.1:domainlicenses` property of your current domain.

2. Write a script to get and update the value, save it as shell script and test it (running as Folio service user on a backend server).

Test expression using `fsceval`

```
su - fscsrv
HOST=localhost
PORT=18070

fsceval -nologo -file fsclicense.exp
```

3. Extend your script by means of “grepping” for the desired value in the output and storing the result into an app.telemetry status file and configure a `cron`-job to call this update script periodically.

Update shell script (`update-license-expiration.sh`)

```
#!/bin/bash

HOST=localhost
PORT=18070

LD_LIBRARY_PATH=/opt/app.telemetry/lib64:/opt/fabasoft/share/eval:/opt/fa
basoft/share/eval/INSTALLDIR/Domain_1_1:/opt/fabasoft/share/eval/INSTALLD
IR/Domain_1_1001

export HOST PORT LD_LIBRARY_PATH

/opt/fabasoft/bin/fsceval -nologo -file /home/fscsrv/fsclicense.coo
  | grep value | awk '{print substr($3, 2, length($3) - 2)}'
  > /var/opt/app.telemetry/status/fsc-lic-expiry.txt
```

4. Create a new app.telemetry counter check to monitor the value from the status file and define the update interval and the warning/critical limits (e.g. warning below 30 days and critical below 5 days).

2.10 Data Retention Strategies for app.telemetry

Fabasoft app.telemetry provides different strategies for managing data retention.

Larger amount of data is stored by the app.telemetry Server continuously by the following services:

- **Software-Telemetry request data** … is stored in a database table for each log pool (1 record/request). Cleanup rules can be configured in the log pool configuration dialog.
- **Software-Telemetry request detail data** … is stored as rawdata on the file system in "daily"-directories (`…/server/telemetry/<yyyy-mm-dd>`). This kind of data is consuming much more disk space than the request records on the database.
  Note: Do not delete the directory of the current day (without a full restart of all server processes). Instead you should use the automatic cleanup rules from the "Server Properties" dialog.
- **Precalculated log pool statistics** … are stored beside the request database tables for each log pool.
- **Software-Telemetry sessions** … are stored on the file system containing request data and request detail data (rawdata). They are created on demand by a user or when a feedback is sent.
  - **Started/Stopped Server sessions** are read from rawdata so they are available until rawdata of the respective time is being deleted. To keep the data permanently make sure to download the session as a zip file.
  - **Feedbacks** will be automatically extracted from rawdata into zip files so that the feedback are available even after the rawdata have been cleaned up. Session zip files are deleted when the corresponding feedback is permanently deleted from the inbox.
  - **Uploaded sessions** are stored as complete ZIP-archives containing all required data on the worker and are already independent from the rawdata directories.
- **Counter check data** can also be stored in a database table (1 record/check).
- **Recorded status change records** can be configured to be automatically deleted after x days via the global "Server Properties" (since version 14.2 configurable – before it was defined to be 1 month).
- **SLA-relevant availability data** information will never be deleted from the defined database.
- **Activity statistic data** … since version 2015 all telemetry requests are analyzed and statistical data (module and event activity statistics) is stored on the file system (`…/server/telemetry/activitystats/`). Cleanup strategies can be configured in the global "Server Properties" dialog.

In order to handle the increasing amount of data and prevent the disk from running out of free space you can configure automatic retention time periods within the app.telemetry client. For more details read the sub chapter "Cleanup Strategies".

Note: Before starting to delete any data you should export Software-Telemetry server sessions and feedbacks by a special automatic app.telemetry server task (configuration setting) in order to access the request details of such sessions later on.

2.10.1 Automatic Export of Feedback Sessions

In most situations feedbacks should be available for a much longer time period than standard request detail data which will be held for post-problem analysis for some time. Therefore you can split off the detail data required for the feedbacks from the normal rawdata directories.

This feature will be automatically enabled after updating to version 2014 Spring Release or later unless it is explicitly disabled in the server configuration file. If you want to change this setting or the session export path stop the app.telemetry server daemon then open the configuration file of the

app.telemetry server (`/etc/app.telemetry/server.conf`) and setup and activate the configuration parameter "`SoftwareTelemetrySessionPath`":

```
# The SoftwareTelemetrySessionPath property defines the target location
# for extracting reported telemetry sessions from the raw data files. (optional)
# The default path is: /var/opt/app.telemetry/server/sessions
# To disable automatic session extraction uncomment the line below (set to empty)
SoftwareTelemetrySessionPath /var/opt/app.telemetry/server/session
```

After this configuration is activated you can start the app.telemetry Server daemon again.

After a while the server will start processing all available existing telemetry sessions and export them to the configured directory. This process may take some time depending on your infrastructure and on the number and size of reported sessions/feedbacks. You can watch the progress of that action by the increasing directory content size on the file system. This process is an ongoing process that will also export new incoming feedbacks a couple of minutes after they have been fully completed.

2.10.2 Cleanup Strategies

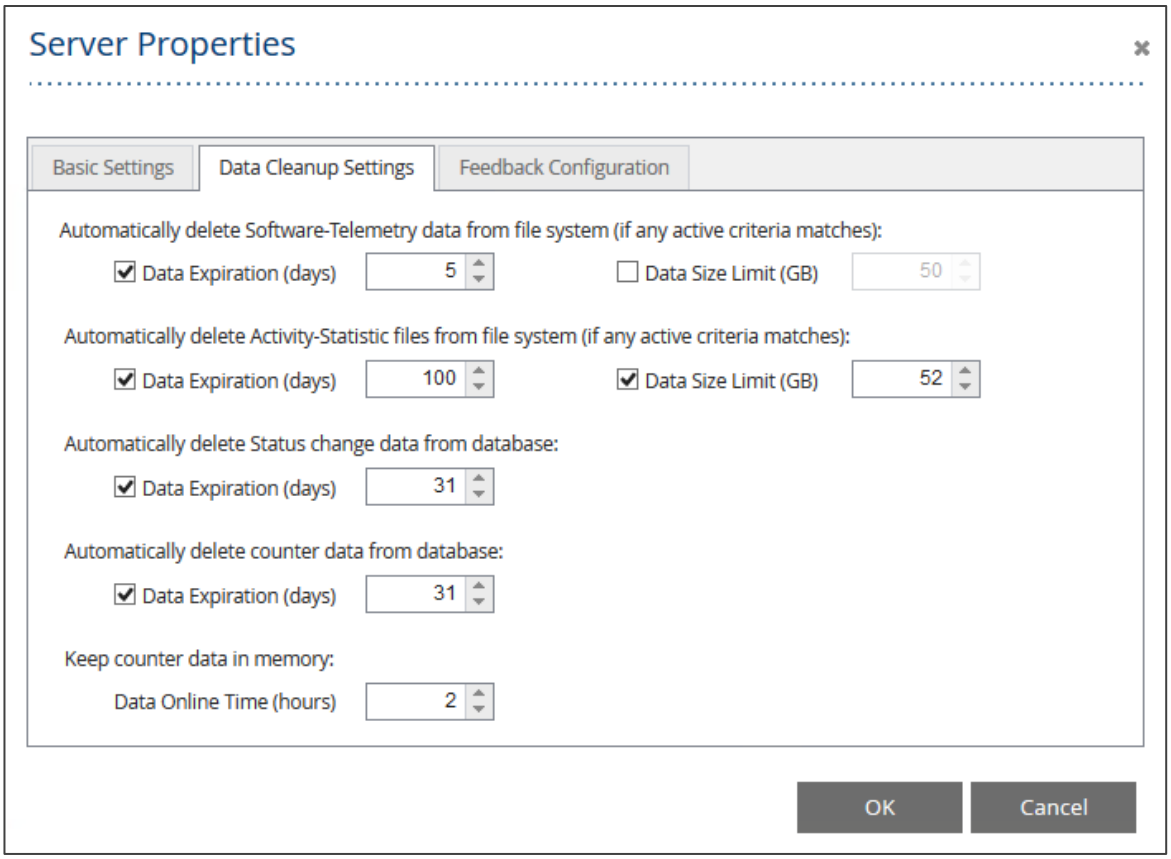
Within the Fabasoft app.telemetry client you can define different automatic data deletion rules in order to keep essential data for a defined time period but prevent filling up the disk with old data not required any more.

The most cleanup settings can be configured within the global "Server Properties" dialog (in edit view at the top of the infrastructure tree).

You can either **limit** the retention of the data **by time** as number of days. If you activate this cleanup rule, any data that is older than the defined time range will be deleted a non-deterministic time span later (please be patient after applying the changes and give the server some time to process the cleanup).

The other possibility to limit the amount of data (only available for file system based data) is to set a **data size limit** in gigabytes (GB). But be careful this limit is only an estimated size limit and can vary a little bit.

You can define a single type of limit for every kind of data or even both limits, which mean if your data match one of the two criteria the cleanup will be triggered to reduce the amount of data to fit the all criteria again.

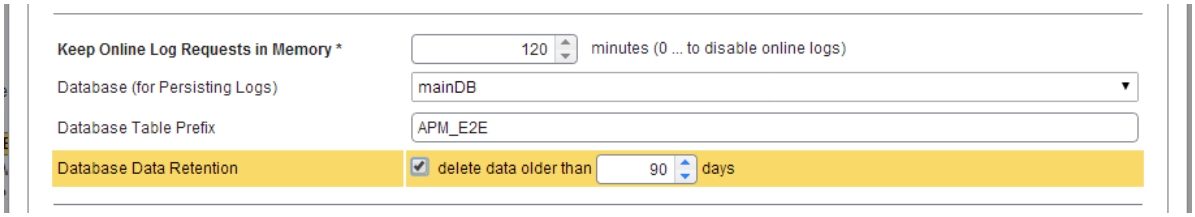


The data retention for the **Software-Telemetry request detail data** (rawdata on filesystem) is used to reduce the big-sized data of old requests. It is your choice how long you want to keep request detail data for a detailed problem cause analyze (request overview, request details, request statistics, request grades) and depends on the amount of available disk space.

Warning: you should have exported the reported telemetry sessions/feedbacks as described in the last chapter otherwise those session details will not be accessible!

For long-term analysis of your applications you could still use the **activity statics** if you keep those data for a longer period than the request detail data.

Additionally you can configure a **database retention** time range for all **log pools** as parameter on every log pool configuration dialog:



If you activate the database data retention for a given time range (in days) the following data will be automatically deleted from database tables belonging to that log pool:

- Software-Telemetry request data
- Precalculated statistics

The **Status change data** is required to show a table containing the history of all status changes of every service/counter-check. This history data is available for the time range defined in the cleanup settings of the “Server Properties”.

Service checks with a defined SLA-definition are persisted independently and unlimited in a database defined within the SLA-definition.

Counter checks with a defined database for persisting the counter results are also stored on the defined database. On the “Data Cleanup Settings” page of the “Server Properties” you may specify a **“Data Expiration (days)”** value to cleanup all counter values that are older than the given number of days (this option is available with Fabasoft app.telemetry 2015 Rollup 2).

2.11 Analyze Requests using Rules

Analyzing performance issues in requests – especially in distributed applications like Fabasoft Folio – is a time consuming task requiring a lot of application specific knowledge. To simplify this process, a set of rules is being created to identify common issues.

There are common rules applying to any Fabasoft app.telemetry instrumented application and rules specifically written for dedicated Products like Fabasoft Folio. In those rules common problems are identified by the telemetry data included in the request. Designing those rules is an evolving process where analyzing steps originally processed manually are being formalized and automated.

In order to use these rules for analyzing telemetry requests, open a request on the telemetry view and select the new “Grades” tab in the bottom analyzer area.

2.11.1 Analyze Entirely Completed Requests

This rule applies to any request generated by Fabasoft app.telemetry instrumented applications and simply checks, if all processing threads, which occur in this requests were correctly terminated.

There are several reasons why a request could be detected not to be finished:

- A request may be opened while still being processed. You may detect what the request is currently processing by looking at the end of the unfinished threads.
- A request may be too large to be processed. Try to reduce the amount of telemetry points per request by lowering the instrumentation level of the log pool or session.
- A process may have stopped working. The problem may be near to the last telemetry point of the unfinished thread.
- The instrumented application may not handle a failure (e.g. exception) correctly and so no ReleaseContext has been recorded.

2.11.2 Reduce RPCs to Fabasoft COO Service

The Fabasoft Web service communicates with the Fabasoft Backend Services using Remote Procedure Calls (RPCs). Each call requires at least one network roundtrip and the allocation of a backend service thread. Issuing too many calls will result in a delay mainly caused by the network latency. Replacing many small RPCs by fewer larger ones will save roundtrip time and management overhead on client and server side.

The grade of the rule will reflect the potential benefit of an improvement based on the fraction of time consumed by RPC requests in proportion to the total request time.

Thus the main info provided is the count and the duration of all RPCs executed. In addition the duration is split in the communication time and the execution time base on the time difference between the requests on the Fabasoft Kernel side and the execution on the Fabasoft COO Service side.

Especially when a high communication time is indicated, the COO Service RPCs are worth being further analyzed. Assuming that the Web Server and Backend Server are located in reliable and fast network infrastructure high communication time results most likely from a high number of RPCs. Each RPC takes at least half a millisecond overhead for the network to transfer the request, the COO Service to schedule the request to an available worker thread and to transfer the result back to the Web Service. So a high number of RPC requests directly lead to bad performance without having a bottleneck in any single application tier.

In the details section an RPC statistic based on RPC type is being provided indicating how the different RPC types contribute to the total RPC time and count.

The most common problem in this area is the so called **Typewriter**, which can be determined by a high “Request Count” in the “`COOSTAttrSel`” RPC, which is the RPC requesting object information

from the COO Service. The typical source for that situation is a loop iterating over a list of objects without previously loading the required attributes of all these objects in a single call. So any access to an object will require the Kernel to load the object one by one. While this will produce the correct result, it will lead to multiple RPC requests and therefore to bad performance. To optimize the Typewriter scenario requires a call to `coort.LoadAllAttributes/LoadSpecificAttributes` providing the list of all objects being iterated.

The list of the "Top 10 events" issuing RPCs" may help you identifying the method containing the loop. This can be identified by the last Event before the count drops to a low value. Clicking on the Event will lead you to the detail view showing the instrumentation points recorded while processing this method.

2.11.3 Improve Object Attribute Reads

This rule analysis how many object attributes are read and how much time is consumed to do this. The instrumentation points required for this analysis are only recorded in *Debug* mode. In less detailed recording levels this rule will show up as "N/A".

As the `GetAttribute` variants are the usual way to access Fabasoft Folio objects, it is not an error to do so, but if accessing information takes a reasonable fraction of the processing time, it is still a good starting point for further investigations. So use the list of "Top events accessing object attributes" to identify the method, in the context of which many attributes are being accessed.

You may optimize data access either by caching or by calling `GetAttribute` once instead of iterating an attribute using `GetAttributeValue`. Also a call to `HasAttributeValue` with a subsequent call to `GetAttributeValue` can often be replaced by a single `GetAttributeValue` saving at least the overhead of an access check.

When looking at the "Attribute Access Statistics" you can determine the duration for a specific type of data access. Most interesting here is the fraction between the "Duration" and the "Self Time" where a high "Self Time" indicates, that the duration mainly results from the count of data accesses, whereas a low "Self Time" compared to the duration indicates cache misses either on the object itself or on objects required for the access check. Cache misses will result in RPCs fetching object data from the COO Service.

Click on the Call name to go to the "Selected Statistics Events" tab, sort by duration and try to solve the performance problem, when attribute accesses lead to COO Service requests.

2.11.4 Optimize HTTP Requests

The Fabasoft Client communicates with the Fabasoft Web services using http requests. Each call requires at least one network roundtrip and the allocation of a web service thread. Issuing too many calls will result in a delay mainly caused by the network latency.

The "Optimize HTTP Request" rule helps you determining why a request on the Fabasoft Web Browser Client is slow.

- The number of HTTP requests should be low. Each http request requires a network roundtrip which can be very expensive when the device is connected via mobile data connections.
- In addition browsers limit the number of concurrent connections to a server so otherwise parallel requests will be serialized limiting the number of parallel requests.
- The "Client Request Time" indicates the time required to send the request until the response has been fully received by the browser. The "Client Processing Time" represents the time needed to process the result and to make the required changes to the page being displayed. A large "Client Processing Time" indicates a high rate of changes to the DOM model of the page, which is most likely the result of large lists being rendered. Older browsers (e.g. Internet Explorer 7 or 8) have significantly slower engines which results in higher "Client Processing Time".

- The “Network Time” is calculated from the difference between the “Client Request Time” and the “Server Processing Time” and depends mainly on the connection speed, connection latency and the amount of data being transferred. The size is being indicated here by the “Request Size” and “Response Size”.

Based on that analysis you can focus on the part of the request, which has most influence on the request time.

2.11.5 Optimize Database Statements

Fabasoft COO-Services read and write object data from/to a relational database. Reading data is required in case of queries and when objects are currently not in the COO-Service cache. Writing data occurs every time objects are being created, changed or deleted. Object lock information is also persisted on the database.

The way how to optimize queries depends on the type of database statement:

Reading Objects (`COOSTAttrSel`):

- Reading objects only occurs when objects are not in the COO-Service cache. So try to keep all active objects in the COO Service cache.
- Minimize the working set of objects being accessed by the user.
- Preload objects together before iterating a list of objects to reduce the count of database queries.

Queries (`COOSTQuery`): Processing queries is executed in several phases:

- Query for the object ids matching the properties given.
- Load all properties of all matching objects which are not in cache.
- With this information the kernel applies the security model and filters out those objects that may not be found by the user.

2.12 How to Configure Certificate Authentication on Linux

The default authentication method for Fabasoft app.telemetry web browser client users is "Basic Authentication". Since 2011 Winter Release you may use https with client certificates as an alternative login method. The following guide explains how to configure “Certificate Authentication” for Fabasoft app.telemetry using an Apache webserver on a Linux system.

Prerequisites:

- Server Certificate (pem and key file)
- CA certificate(s) of the client certificates

Configuration:

- Install mod_ssl (`mod_ssl-<version>.<os>.<arch>.rpm`)
- Copy the server certificate (e.g. to `/etc/pki/tls/certs/`)
- Copy the server certificate key file (e.g. to `/etc/pki/tls/private/`)
- Use `chmod/chown` to provide access to the certificate files to the apache user
- Configure Apache SSL and access restrictions (e.g. `/etc/httpd/conf.d/ssl.conf`)
  - Set the "`ServerName`" to the name used in the server certificate
  - Set the "`SSLCertificateFile`" to /etc/pki/tls/certs/<your certificate>.pem
  - Set the "`SSLCertificateKeyFile`" to /etc/pki/tls/private/<your certificate>.key
  - Set the "`SSLVerifyClient`" to `require`
  - Set the "`SSLVerifyDepth`" to a value of your chioce (e.g. 10)
  - Add the `SSLOptions +StdEnvVars`
- Import the client CA certificate
  - `openssl x509 -in cacert.cer -text >> /etc/pki/tls/certs/ca-bundle.crt`
- Add the common names (cn) of the users to the groups provided in `/etc/app.telemetry/htgroup`
- Restart the Apache web server (you have to provide the passphrase for the server certificate key file when restarting the web service).

Details / Hints:

- To extract the pem/key files from a pkcs12 certificate you may use the following commands:
  - `openssl pkcs12 -clcerts -nokeys -in vmcentos.pfx -out server.pem`
  - `openssl pkcs12 -nocerts -in vmcentos.pfx -out server.key`

2.13 Use Case Analysis

Implementation:

The object context required by the use case analysis has to be provided by implementing a wrapper to the GetObjectContext method. This method is called several times per request on the object a view or use case is operated upon. The wrapper should set the return parameters `ctxobj`, `ctxtype`, `context` and `category`, where `ctxobj` is the „container“ object (e.g. teamroom or file) of the current object, `ctxtype` is the class or type information (e.g. objectclass or file plan entry). The `context` is a string representing hierarchical context information (e.g. the file plan hierarchy, the syntax is currently unspecified and unparsed). The `category` should be set to the main document category.

Prerequisites:

- Fabasoft Folio/eGov Suite Version 21.1.0 or later (fscvapp)

- Implementation of a postwrapper for FSCVAPP@1.1001:GetObjectContext
    - Fabasoft Cloud Teamroom Implementation in 21.1.0
    - Fabasoft eGov-Suite Implementation to be done
- Fabasoft app.telemetry Server 21.1.

Configuration:

- Use Extended Log Definition (UseCase)

3 Known Issues

This chapter lists general known issues by Fabasoft app.telemetry encompassed with the recommended solution.

3.1 Connection to Agent Refused

Issue:

The status of a Fabasoft app.telemetry agent is displayed as "*not connected*" in the agent view of the app.telemetry client.

The error message text is similar to the following:

```
Connection refused because of multiple active connections
```

This situation is caused by multiple connections to a single Fabasoft app.telemetry agent. Multiple connections to a single Fabasoft app.telemetry agent might occur because one of the following configuration issues:

5. More than one Fabasoft app.telemetry server connect to the same Fabasoft app.telemetry agent.
6. One Fabasoft app.telemetry agent is configured multiple times within one Fabasoft app.telemetry server installation.

Solution:

To resolve the situation, ensure that following criteria apply:

1. Each Fabasoft app.telemetry agent is dedicated to one specific Fabasoft app.telemetry server.
2. All agents configured in the Fabasoft app.telemetry client connect to dedicated Fabasoft app.telemetry agents (e.g. no duplicate "*network address*" properties).

3.2 Turn on Crash Dumps on Linux

Issue:

The Fabasoft app.telemetry Linux services support as any other Linux process to produces a crash dump. With the default configuration this feature is disabled but can be enabled with a simple configuration file.

Solution:

To turn on crash dumps for app.telemetry Linux services (Agent, Server) follow the described steps below:

1. Open and edit the *sysconfig* configuration file for the `apptelemetryagentd` or the `apptelemetryserverd` in order to change the setting for the specified process. The files are self-explaining. Just uncomment or change the appropriate lines and restart the processes by means of using the init script.
   - `/etc/sysconfig/apptelemetryagentd`
   - `/etc/sysconfig/apptelemetryserverd`
2. 'Enable core files' by setting the core file limit to unlimited or any other reasonable value.
   - for Red Hat-based Linux systems: activate/uncomment the following line:
   - `DAEMON_COREFILE_LIMIT='unlimited'`
3. Set the core file location (optional):
   3.1. default (if not set explicitly) the system will write the core files two one of the following locations (depending on the startup kind of the process - during system boot or manually)
   - Processes started during system boot will be dumped to the root file system folder "`/`" (`/core.1234`)
   - Processes started manually by a user using the init script will be dumped to the current folder
   - In some cases the dump file may be placed next to the binary file (`/opt/app.telemetry/bin`)

   3.2. You can explicitly set the dump file location via the kernel `core_pattern` setting
   - Create the desired target directory and make it world writeable:
     ```
     mkdir /tmp/dumps
     chmod 777 /tmp/dumps
     ```
   - set the core file pattern in the kernel variable (file)
     ```
     echo "/tmp/dumps/core" > /proc/sys/kernel/core_pattern
     ```
     the default pattern is "`core`" which will place a file starting with the name "`core`" in the default location described above.
4. Restart the processes by means of using the service utility
   - `service apptelemetryagentd restart`
   - `service apptelemetryserverd restart`

Finding core files somewhere on your Linux system:

If you don't know where your core files are placed and you know there must be a core file somewhere on your system, you could try finding the core file with the following find statement:

```
find / -mount -type f -regex ".*/core\.[0-9]+$"
```

3.3 SNMP Counter not Available

Issue:

It may occur that some Fabasoft app.telemetry checks return the message "*SNMP-Counter (...) not available*". This message indicates that the desired SNMP counter could not be checked by the Fabasoft app.telemetry agent.

The numeric value in brackets in the message (for example: `.1.3.6.1.4.1.2021.4.15.0`) is the SNMP OID of the defined counter check that is queried to get the resulting value. If there are more than one of such messages listed, the counter value is calculated based on a formula consisting of several SNMP values.

The case that SNMP-Counters are not available can have different problem causes:

1. no SNMP daemon installed on target system (package net-snmp is missing)
2. SNMP daemon not running on target system
3. SNMP configuration invalid to get that counter
   - 3.1. security restrictions: SNMP community string mismatch
   - 3.2. security restrictions: SNMP security view does not allow accessing that counter
4. app.telemetry counter check configuration invalid for that target system – counter or instance does not exist

Note: many app.telemetry counter checks for unavailable SNMP counters with a short update interval may decrease the app.telemetry agent performance, so check your counter configuration carefully!

Solution:

To resolve this problem follow the solution tips for the appropriate problem item (item number from issue list above) and test the configuration as described below.

1. If no SNMP daemon is installed, install the software-package "net-snmp-5.x.<arch>.rpm"
2. If SNMP daemon is not running, start the daemon using "`service snmpd start`". Additionally you should set the SNMP daemon to autostart at system startup "`chkconfig --level 345 snmpd on`"
3. Check the SNMP configuration / security restrictions ... (`/etc/snmp/snmpd.conf`)
   - 3.1. Configure SNMP authentication: configure SNMP community string on both sides. The SNMP community string set in the `snmpd.conf` configuration (`rocommunity <string>`) has to match the defined community string in the app.telemetry agent configuration (app.telemetry web client > edit mode > target agent > SNMP community
   - 3.2. Configure SNMP view restrictions: you should include all required SNMP counters in your SNMP configuration
     - 3.2.1. The simplest configuration is to disable all access control configuration lines in the configuration file and only set the SNMP community string with the following configuration line: "`rocommunity <string>`"
     - 3.2.2. Another possibility is to include all counters and do not restrict any counters by means of using the following configuration line: "`view systemview included .1`"
     - 3.2.3. The most secure configuration is to include only the required SNMP subtrees for your desired counters, but then you need to find out all required SNMP OIDs.

To test the SNMP configuration and the availability of the SNMP counters you need to have the additional SNMP software package "`net-snmp-utils`" installed on one of your Linux systems that could connect to the problematic target system (for example you could install these utilities on the app.telemetry server).

All you have to do is run the following commands with your concrete values set:

Linux Shell – Test SNMP Example

```
snmpwalk -v 1 -c <COMMUNITY-STRING> <TARGET-AGENT-IP> 1.3.6.1.2.1.25.1
snmpwalk -v 1 -c <COMMUNITY-STRING> <TARGET-AGENT-IP> 1.3.6.1.4.1.2021

# example ...
snmpwalk -v 1 -c public 10.20.30.40 1.3.6.1.2.1.25.1
snmpwalk -v 1 -c public 10.20.30.40 1.3.6.1.4.1.2021
```

3.4 app.telemetry Client not Working after Installation

Issue:

After installation of Fabasoft app.telemetry on a new Linux system and trying to access the Fabasoft app.telemetry client with your web browser the client may show an info dialog with the message that something is not working properly.

The following reasons could prevent the Fabasoft app.telemetry client from working properly:

1. Fabasoft app.telemetry server is not running
2. SELinux is running in “enforcing” mode and the app.telemetry SELinux policies have not been installed or applied correctly.
3. The app.telemetry connection/listening ports of server/webserver are misconfigured

Solution:

The solutions for the reasons above (same numbered item) are listed below:

1. Start the Fabasoft app.telemetry server service/daemon and check if it is set to automatic startup
2. SELinux Troubleshooting
   2.1. If you don’t really need SELinux, just turn it off or to permissive mode
      - Check SELinux state: `sestatus`
      - Change to permissive mode: `setenforce 0`
      - Change SELinux configuration mode for startup (persistent) – modify `/etc/selinux/config`

   2.2. Check the setup output of the Fabasoft app.telemetry RPM installation and see if any Warnings/Errors occurred during setup /var/log/app.telemetry/<server|agent|webapi>-rpm.log
      - On installation errors you might have not installed the required RPM packages for SELinux policy file management (`policycoreutils-python`) – install the missing packages and reinstall the app.telemetry RPM files

   2.3. Maybe you have to restart the app.telemetry daemons to get the SELinux policies be applied

   2.4. Check the SELinux audit log and grep for denied log entries
   `cat /var/log/audit/audit.log | grep “=AVC”`
3. Check if the app.telemetry server and agent are running and listening on their configured ports by means of using the `ss -tnl` tool (apptelemetryserver default port 10000, apptelemetryagent default port 10001)

3.5 Security Warning/Restriction for End-2-End Instrumentation

Issue:

The Fabasoft app.telemetry end-2-end instrumentation for websites via the JavaScript SDK is based on data communication POST-requests between the web site and the Fabasoft app.telemetry web service (WebAPI).

Web browsers have security limitations for POST-requests invoked from JavaScript for different locations (different protocol, target, port) so you may receive warning messages or even all the data transfer with the WebAPI is blocked and no end-2-end instrumentation is possible.

The following table shows the different access possibilities for the example base page:
`http://store.company.com/dir/page.html`

URL	Result	Reason
`http://store.company.com/dir2/other.html`	Success	
`http://store.company.com/dir/inner/another.html`	Success	
`https://store.company.com/secure.html`	Failure	Different protocol
`http://store.company.com:81/dir/etc.html`	Failure	Different port
`http://news.company.com/dir/other.html`	Failure	Different host

Source: https://developer.mozilla.org/en/Same_origin_policy_for_JavaScript

So if you have any troubles with denied POST-requests to the configured app.telemetry WebAPI check and compare the used URLs and follow the hints listed below.

Solution:

In order to use Fabasoft app.telemetry end-2-end instrumentation (via JavaScript SDK) ensure that the instrumented web site is allowed to send the data (POST-requests) to the configured WebAPI url.

1. The simplest solution is to install the app.telemetry WebAPI on the same host (also same port and protocol) that also provides your instrumented web page.
2. Using a Load-Balancer to route the requests correctly via a sub directory.
3. Server-side routing of the requests using a local WebAPI url inside your instrumented web page and some special server-side logic capturing and forwarding all telemetry requests to the WebAPI.
4. Using Apache mod_alias, mod_proxy or mod_rewrite to route the requests.
5. Using an iframe with script content from the target domain and specific client-side code to handle the data.
6. Using a web browser with support for cross-origin resource sharing (https://dvcs.w3.org/hg/cors/raw-file/tip/Overview.html - e.g.: Mozilla Firefox 3.5). This may also require some code modifications.
7. Using „Signed Scripts for Mozilla Firefox“ (http://www-archive.mozilla.org/projects/security/components/signed-scripts.html).

3.6 Feedback Dialog shows two Screenshot Properties

Issue:

When using the Fabasoft app.telemetry feedback dialog (report dialog) via the app.telemetry API and your application is running in a Fabasoft application context with available Fabasoft plugin there are different screenshots provided:

- A native screenshot provided by the Fabasoft plugin with a native screenshot preview opened in a desktop application (this type is also called the legacy native screenshot)
- An HTML5 screenshot (which may also fetch the screenshot data from the Fabasoft plugin if available) providing a full-featured screenshot inline preview with mark/crop/blackout features.

If you can see those 2 different screenshot attachments in your feedback dialog you are still using a legacy configuration when calling the `createDialog`-API-call.

Solution:

In order to disable the legacy native screenshot you have to set the `screenshot` property object to "`enabled:false`" when passing to the `createDialog`-API-call.

3.7 Bar Chart Text Overlapping

Issue:

In some situations your dashboard charts may look improperly (e.g. long bar chart label texts overlap).

Depending on your web browser window size, your dashboard settings (number of columns to display) and your chart settings (height of chart) the chart content may fit into the available space or not.

Solution:

Beside the basic features of reducing the number of dashboard columns, increasing the chart height or reducing the label length (of your service checks) some new improvements may help you solve your problems.

With Fabasoft app.telemetry 2012 Spring Release the chart capabilities have been improved the following way:

- Chart legend labels are limited in their maximal length (depending on the chart width) - anyway you should display the legend only if you need it (for bar charts this information is most times redundant)
- Gauge chart scaling improved
- Bar Charts axis labels improved:
  - Horizontal bar chart: long axis label texts now wrap around (depending on the chart size)
  - Vertical bar chart: the axis label texts can now be rotated to display them diagonal or even vertical to prevent overlapping texts ... for this purpose you can define the axis label rotation angle in the chart properties (edit view) with a value between -90° and 90° (negative angle values (e.g.: -30) are displayed left of the bar and positive values (e.g.: 45) are displayed right of the bar, a value of 0 displays the labels horizontal as before)