Deltek

Deltek Maconomy®

Application Performance Monitoring (APM)

June 28, 2024

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This edition published June 2024.

Contents

Overview

Application Performance Monitoring (APM) for Maconomy is a logging framework designed to serve several different monitoring purposes:

1. Audit logging of all user logins
2. High-level activity logging of incoming requests
3. Detailed drill-down logging of long-running requests

Maconomy's APM framework consists of a core logging framework, and a number of loggers distributed throughout the Maconomy technology stack. When triggered during the execution of a request, these loggers create log events that measure the time spent of a specific aspect of the request execution.

The generated log entries are passed to the core logging framework, where they are filtered according to the framework configuration settings. The purpose of this filtering is to ensure that only log entries which are of interest are included in the final log output.

In addition to tracking the performance of the workspace and container contribution layers in the Coupling Service, the APM framework can also collect log events from a number of Maconomy Server-side loggers, from the Maconomy Application layer, all the way down to the database interface, and back up again.

This means that it is possible to quantify what fraction of a long-running request is spent on application processing and database queries. Combined with the high-level information provided by the workspace and/or container contributions involved in the request, this can provide a great starting point for a targeted effort to identify and improve on specific performance bottlenecks anywhere in the Maconomy technology stack.

Before You Begin

The Application Performance Monitor (APM) framework uses different modes. Before you begin, review the mode types below and consider which of these modes most closely matches your logging requirements.

Modes

Audit Logging of All User Logins

This is the most basic mode which produces the fewest log entries per time period. In this mode you are only interested in events that are marked as 'audit' by default. You can set a high minimum time for logging non-auditable events, and limit the list of included server log entries to include only 'SecurityLog'.

You can still use a 3rd party log analysis tool to collect the produced log entries, but a simpler manual approach would also be quite feasible.

High-Level Activity Logging of Incoming Requests

In this mode, you will mark a selection of high-level log entries as auditable to ensure that they are always logged. This can be combined with a high minimum time and a limited list of included server log entries, as described for Audit Logging above. In this mode you can collect reliable statistics on request execution time across all client interfaces, while still keeping the volume of log entries down.

While it will still be possible to analyze the log entries manually you will likely quickly see the benefits of integrating with an automated log analysis tool stack.

Detailed Drill-Down Logging of Long-Running Requests

Drill-down logging is achieved by configuring the minimum exit time of log entries to match the minimum time duration you wish to be able to perform drill-down analysis on. Each request being logged is represented as a tree of log entries where the root connection is the log entry describing the larger operation that the log entry is a part of.

The lower you set the minimum exit time, the deeper and wider the tree will become, with more branches describing the various sub tasks that the request consists of. More detailed drill-down logging will also result in a higher log volume, which will increase the I/O pressure and log file size on your server.

Finding the right balance between desired level of detail and acceptable I/O overhead might require a bit of experimentation. In general we will advise against setting the minimum exit time of log entries lower that 100 milliseconds.

Setup and Configuration

APM setup and configuration includes these parts:

- APM Setup
- Maconomy Setup
- Configuration

APM Setup

Log Modes

The APM framework can operate in two modes:

Direct: Log entries are written directly to the SLF4J logger. This is the most verbose logging mode and can be expected to produce large quantities of log entries on a production system.

This mode is not recommended (as the logging itself negatively impacts performance)**.**

Delayed: Log entries are queued with a configurable delay. If the total duration of a log event is smaller than the delay its entries will not be logged. This is the recommended mode.

To enable Direct Log Mode:

1. Open the **monitor.ini** file in /configuration/settings.
2. Update the following line:

```
log.monitor {
  minimum-time = 0
}
```

3. There is a significant performance penalty associated with logging in direct mode on the MaconomyServer process. To safeguard itself against this performance penalty it will enforce a default minimum time of 10 milliseconds.

   To override this safeguard and enable direct logging on the MaconomyServer process, update the following line as well:

```
log.monitor.server {
  all-entries = true
}
```

To enable Delayed Log Mode:

1. Open the **monitor.ini** file.
2. Update the following lines:

```
log.monitor {
  minimum-exit-time = a positive integer value
  minimum-time = a positive integer value (higher than minimum-exit-time)
```

```
}
log.monitor.server {
  all-entries = false
}
```

See Configuration for more information.

Note: Entries with log level ‘ERROR’ and entries marked with the ‘Audit’ flag are always logged.

Log Levels

All log events are associated with an SLF4J log level, which is applied to the log entries generated by the event. The monitor logger declared in the Logback configuration will compare this level with its own level to determine if the log entry should be included in the log output. The default log level is INFO, but a few log events set a lower level such as DEBUG. If the operation monitored by a log event trigger a serious or unexpected error, the level of all subsequent log entries from that log even are raised to ERROR.

A log entry can declare a number of key-value pairs containing extra information about the event being logged. Each of these key-value pairs are also assigned a log level. Less verbose values are generally assigned a high level (typically INFO), while more verbose values might be given a lower level (DEBUG or TRACE).

Change the log level of the Logback monitor logger to control both the number of log entries included in the log output, and to some extent the verbosity of each log entry.

These log levels are available

ERROR: Only monitor serious error conditions occurring while serving requests.

INFO: Used for high-level activity monitoring. This provides an overview of system activity while skipping some of the more verbose fields, such as key field information and search restriction on container requests.

DEBUG: Provides more detailed monitoring of system performance. This includes key field information and search restrictions for contain requests, which are helpful when investigating the runtime complexity of the different requests.

TRACE: All log events and all key-values. The additional information included at this level is rarely needed, and it is generally not a recommended log level to use in production.

Regardless of which log level you chose, best practice is to always configure the APM framework with a minimum time duration to reduce the volume of log entries included in the log output.

To configure a minimum time duration, see Configuration for more information.

Log Entries

Log events created by the APM framework follow a simple life cycle:

Enter: The log event is created and an ‘Enter’ message is issued. (optional)

Message: Additional ‘Message’ entries issued for an active log event. (optional)

Exit: The log event is closed, and an ‘Exit’ message is issued.

Each log entry produced by a log event consists of a collection of structured key-value pairs (Key=”Value”). This format, while somewhat verbose, is quite readable, but it is primarily designed to be easily consumable by external log monitoring frameworks such as Splunk or Logstash.

The information included in each log entry is designed with reporting in mind, and therefore contains several ID fields which group the entries on different dimensions, such as those belonging to the same request, or to the same user login session.

The APM framework can be configured with various time duration thresholds used to omit log entries which are considered too fast to be of interest. For this reason the Enter and Message entries of a log event might fall below the configured threshold and be omitted. If the entire event falls below the threshold the Exit entry will also be omitted, unless another condition dictates that it should be included anyway.

Each log entry is annotated with the following information:

Field	Description
Time	A timestamp for when the log entry was created (*not* when it was written).
LogDepth	The nesting level of the log entry.
Thread	The name of the thread that created the log entry.
Name	A standardized name for the logger.
Type	Either 'Enter', 'Message' or 'Exit'.
Level	An SLF4J log level for the log entry. If this level is lower than the log level of the Logback logger ('com.maconomy.util.logger.McMonitorLogger') the log entry will not be written.
Duration | Elapsed	The time duration (in milliseconds) since the log event was created. If the time measurement is not entirely contained within the scope of the log event then it will be reported as 'Elapsed' rather than 'Duration'.
*Values**	Additional key-value pairs. Each value is annotated with an SLF4J log level for controlling whether the value should be included in the log entry.
Error	Any error that was encountered while executing the logged action (optional). If the error is considered serious (internal/unexpected) the log entry will be elevated to ERROR level, otherwise it will remain at its current level.
Audit	A flag ('true/false') indicating whether the log entry contains audit-related information. (optional, 'false' if absent)
PrincipalName	A Maconomy User- or System Principal Name describing the available user and/or system information (e.g. macoprod.en_US\Administrator\Standard).
EntryID	A semi-unique ID identifying log lines generated by the same log entry.
RequestID	A unique ID identifying the request that the log entry belongs to. (optional)
ContextID	A unique ID identifying the wider context that the request appears in, such as the user's login session. (optional)
Connection	A description of the connection to the client. If present the description can be expected to adhere to the following format: [*obj*[@*id*]] l(*addr*:*port*)<->r(*addr*:*port*) -where: '*obj*' is the object providing the connection information '*id*' is the instance ID of '*obj*' (Java Identity Hash Code)

Field	Description
	'l(…)<->r(.'.)' describe the local<->remote connection endpoints '*a'dr*' is the endpoint IP address '*p'rt*' is the endpoint port number
Client	A string identifying the client that issued the request.

Configuration

The APM framework is configured in 'configuration/settings/monitor.ini' and ‘configuration/logback.xml’ on the Coupling Service. The default versions of these files contain a suggested setup with explanatory comments and sensible default values.

All settings are dynamically monitored, and can be changed ad hoc without having to restart the Coupling Service or any other Maconomy component.

monitor.ini

The table below describes all available settings in ‘monitor.ini’:

Setting	Description	Default
log.monitor.*		
enabled	Change to 'true' to enable monitor logging.	false
time-unit	The time unit for 'minimum-time'. Must be a unique prefix of 'milliseconds', 'seconds' or 'minutes'.	seconds
minimum-time	The minimum time duration for log entries to be printed. Setting this to '0' will put the framework in 'direct' mode.	0
minimum-exit-time	The minimum time duration for 'Exit' log entries to be printed. This can only be set to a value <= 'minimum-time'	*minimum-time*
time-format	The date/time format pattern to use for the time stamp on each log line.	yyyy-MM-dd HH:mm:ss.SSSZ
maximum-value-length	The maximum length of values including in log entries. Values exceeding this length will be truncated.	500
mdc-keys	A comma-separated list of keys for the Mapped Diagnostic Context (MDC) values to include in all log entries.	*n/a*
include, exclude	Comma-separated lists with string patterns for log entry names to explicitly include/exclude. • If only an include list is specified, then log entries will only be included if they match one of the include patterns. • If only an exclude list is specified, then log entries will be included unless they match one of the exclude patterns. • If both an include and an exclude list is specified, then the include list contains exemptions from the exclude list. I.e. if the log entry name matches one of the patterns in the exclude list, then it will only be included if it also matches one of the patterns in the include list.	*n/a*
audit	A comma-separated list of string patterns for log entry names that are auditable, and which should therefore be logged even if their duration is below the minimum time duration.	*n/a*

Setting	Description	Default
~~exit-log-depth~~	**Deprecated (use 'audit' instead)** Write exit messages for the first N log entry levels, even if they don't meet the minimum time duration criteria. (only in: 2.4.4)	1
log.monitor.server.*		
enabled	Explicitly enable/disable server-side monitor logging.	*log.monitor.enabled*
include	A comma-separated list of the server-side loggers to enable.	*n/a*
all-entries	Bypass the "delay queue" in each server process and send all server-side log entries to the Coupling Service. When this option is not used a default minimum delay is configured on each server process, to conserve network bandwidth and reduce log entry volume. At the time of writing this minimum delay is set to 10 milliseconds.	false
push	Use push messages to send server-side log entries to the Coupling Service	true

To configure APM Log Modes and Log Entries:

1. Open the **monitor.ini** file.
2. Update the following lines (using needed values):

```
minimum-time = 30
minimum-exit-time = 5
maximum-value-length = 200
```

Example APM Framework Configuration

In the example below, the APM framework is configured to provide:

- Top-level monitoring of all Workspace Client-, Web- and WebDaemon requests
- Drill-down logging of call trees exceeding 5 seconds
- Additional Enter/Message logging of requests exceeding 30 seconds

Note: The ‘server.include’ setting lists all the server-side loggers which are available for monitoring.

Example

```
log.monitor {
  enabled = true
  time-unit = seconds

  # The minimum time duration for log entries to be printed.
  minimum-time = 30
```

```
    # The minimum time duration for 'Exit' log entries to be printed.
    minimum-exit-time = 5

    # The maximum length of values including in log entries.
    maximum-value-length = 200

    # A comma-separated list of string patterns matching the names of log
    # entries that are auditable, and which should therefore be logged
    # even if their durations are below the minimum time duration.
    audit = Workspace:*, WebService:*, WebDaemon:*

    # A comma-separated list of the server-side loggers to enable.
    server.include = Database,SecurityLog,WebComm,MScript,MSL,RPC
  }
```

logback.xml

All log entries generated by the APM framework are produced by a single SLF4J logger in the Coupling Service. The standard Logback configuration file (‘) is used to direct these log entries to an output file (by default ‘).

To direct Log Entries to an Output File:

1. Open the **configuration/logback.xml** file.
2. Update the following lines (using needed values):

   ```
   log/coupling/maconomy-monitor.log
   ```

Example of Logback Configuration

Below is an example of a configuration. In this example, the threshold log level is set to INFO. Refer to the section Log Levels above to learn more about the implications of setting this level.

Example

```
<appender name="MONITOR" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>
    ${logback.output.directory}/maconomy-monitor.log
  </file>
  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <fileNamePattern>
      ${logback.output.directory}/maconomy-monitor-%d{yyyy-MM-dd}.log
    </fileNamePattern>
    <maxHistory>30</maxHistory>
```

```
    </rollingPolicy>
    <layout class="ch.qos.logback.classic.PatternLayout">
      <Pattern>%msg%n</Pattern>
    </layout>
  </appender>

  <logger name="com.maconomy.util.logger.McMonitorLogger" additivity="false">
    <level value="INFO" />
    <appender-ref ref="MONITOR" />
  </logger>
```

Data Extraction

This section provides details on log event types as well as sample queries to provide example of how to extract meaningful information from the Log events Deltek has used this information to build our own dashboards and reports implemented within a third party Splunk APM solution. This solution is available for customers to gain access to use out of the box, or for inspiration for building your own. Customers may want to invest in an alternative third party APM Tool.

Log Events

The following sections describe the different types of log events that are generated by the APM framework.

With a few exceptions, log events with names on the form 'MaconomyServer:*LoggerName*' are generated by the Maconomy Server processes, and require the logger name after the ':' to be included in the 'log.monitor.server.include' setting. (see Configuration above)

Two exceptions to this rule are the 'MaconomyServer:Request' and 'MaconomyServer:Callback' events, which are produced by the Coupling Service before calling remote functions and when receiving callbacks from the Maconomy Server process.

Log Event Types

This section provides an overview of all log event types included in the APM framework.

Authentication	Audit	LoginRules	ApplyAll
Auth:Login	•	•	•
Auth:Logout	•	•	
Auth:Context			

Service Provider Request	Message
ServiceProvider:GetLease	•

Workspace Client Requests	WorkspaceName	PaneRequest	PaneNames
Workspace:Spec	•	•	•
Workspace:PaneSpec	•	•	•
Workspace:Data	•	•	•

WebService / WebDaemon Requests	Method	Url	Query	Request	RemoteAddr	Received / Sent
WebService:Request	•	•	•			
WebDaemon:Request	•		•	•	•	•

Container Requests	ContainerName	Restriction	Parameters	NewValues	ActionName	Operation	SourceRestriction	TargetRestriction	RestoreData	ForeignKeyName	RestrictionData
Container:Open	•										
Container:Close	•										
Container:Specify	•										
Container:Lock (2.3.4 only)	•	•	•								
Container:Unlock (2.3.4 only)	•	•	•								
Container:Initialize	•	•	•	•							
Container:Create	•	•	•	•							

Container:Read	•	•	•								
Container:Update	•	•	•	•							
Container:Delete	•	•	•								
Container:Action	•	•	•		•						
Container:Print	•	•	•								
Container:Move	•		•			•	•	•			
Container:Restore	•		•						•		
Container:Restrict	•		•							•	•

Container Contributions	ContainerName	EventId	ContributionId	BundleId	PhaseId	ActionName
Container:Contribution	•	•	•	•	•	•

Remote Procedure Calls	Function	Parameters	PID
MaconomyServer:Request	•	•	•
MaconomyServer:Callback	•	•	•

Maconomy Server Requests	Audit	Context	PID
MaconomyServer:RPC		•	•
MaconomyServer:WebComm		•	•
MaconomyServer:MScript		•	•

MaconomyServer:MSL		•	•
MaconomyServer:Database		•	•
MaconomyServer:SecurityLog	•	•	•

Authentication

Authentication log entries are produced when user requests interact with the login modules configured in the Coupling Service (in configuration/maconomy.security.config), and when an authentication context is established in order to execute an authenticated request.

This includes all authentication related activity for the Workspace Client (WSC) and Maconomy RESTful web services, but not Maconomy legacy web products such as Portal, MaconomyWS and MScript. Authentication requests from these clients are only logged by the Security Log entries.

The following authentication related log events exist:

Name	Description	Since
Auth:Login	Issued when rules are being (re-)applied for a user.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
LoginRules	The login rules being applied	INFO
ApplyAll	Whether the login rules are applied unconditionally	INFO
Audit	Set to 'true' on the 'Exit' entry to classify it as audit-related (since 2.3.5)	*n/a*

Name	Description	Since
Auth:Logout	Issued when a user is logged out of all previously applied login rules.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
LoginRules	The login rules that were applied	INFO
Audit	Set to 'true' on the 'Exit' entry to classify it as audit-related (since 2.3.5)	*n/a*

Name	Description	Since
Auth:Context	Issued when an authentication context is established for an existing user session. These log events track the total execution time of each action the user performs, including the administrative overhead (e.g. acquire/release a server instance, reconnect/disconnect user, etc.)	2.3.4, 2.4.4
Key	**Value**	**Log Level**
	No additional key-values	

Service Provider Requests

The Service Provider Request event tracks the time spent obtaining a MaconomyServer service lease from the service pool:

Name	Description	Since
ServiceProvider:GetLease	Issued when a service lease is requested from a service pool. Usually these requests will be very fast and therefore filtered away, but if the service pool is running full or is otherwise unable to scale to meet demand then requests will be queued while waiting for a service to become available.	2.3.5, 2.4.4
Key	**Value**	**Log Level**
Message	The service being requested: 'Service: *service-description*'	INFO

Workspace Client Requests

The following log events are issued for Workspace Client Requests:

Name	Description	Since
Workspace:Spec	Issued when workspace specification request is performed.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
WorkspaceName	The name of the workspace	INFO
PaneRequest	A descriptive text for the pane request part of the spec request	INFO
PaneNames	A list with all pane names in the workspace request tree	INFO

Name	Description	Since
Workspace:PaneSpec	Issued when a workspace pane specification request is performed, i.e. a request to attach additional panes to a workspace - typically Foreign-Key search panes.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
WorkspaceName	The name of the workspace	INFO
PaneRequest	A descriptive text for the pane request part of the pane spec request	INFO
PaneNames	A list with all pane names in the workspace request tree	INFO

Name	Description	Since
Workspace:Data	Issued when a workspace data request is performed.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
WorkspaceName	The name of the workspace	INFO
PaneRequest	A descriptive text for the pane request part of the pane spec request	INFO
PaneNames	A list with all pane names in the workspace request tree	INFO

WebService Requests

The following top-level log entries are issued for the REST-full web service requests:

Name	Description	Since
WebService:Request	Issued when a web service request is performed.	2.3.5, 2.4.4
Key	**Value**	**Log Level**
Method	The HTTP method of the request (GET/POST/...)	INFO
Url	The complete URL of the request (protocol://server:port/path), but not including the query string	INFO
Query	The query string of the request	INFO

WebDaemon Requests

The following top-level log entries are issued when a WebDaemon request is performed:

Name	Description	Since
WebDaemon:Request	Issued when a WebDaemon request is performed (Java client, Analyzer or MScript). WebDaemon requests pass through the Coupling Service as a binary protocol, but they can trigger Maconomy Server generated log events.	2.3.5, 2.4.4
Key	**Value**	**Log Level**
RemoteAddr	The value of the X_FORWARDED_FOR or REMOTE_ADDR CGI environment variable passed to the CGI program (optional)	INFO
Method	The value of the REQUEST_METHOD CGI environment variable passed to the CGI program (optional)	INFO
Url	The invoked CGI request URL (without query parameters) OR the CGI executable path and script name (for command-line invocations)	INFO
Query	The query parameters passed to the CGI program, either via the QUERY_STRING CGI environment variable, or on the command-line (for command-line invocations). This field will only be present for requests where the CGI client has not supplied a request description ('Request').	INFO
Request	A short description of the request. If a request description is not supply by the CGI client the default value will be the value of the PATH_INFO CGI environment variable	INFO
Received	The number of bytes received from the CGI program	INFO
Sent	The number of bytes sent to the CGI program	INFO
~~PrincipalName~~	**Deprecated (replaced by an MDC provided 'PrincipalName'):** The system principal of the server pool that will execute the request (only in: 2.3.4)	INFO
~~Socket~~	**Deprecated:** The connected CGI socket (only in: 2.3.4)	INFO
~~Request~~	**Deprecated:** The CGI request descriptor (only in: 2.3.4)	DEBUG

Name	Description	Since
MaconomyServer: WebComm	Issued when a WebComm command is received by the Maconomy Server (Java client, Analyzer or MScript). Requires the 'WebComm' TimeLog logger to be enabled.	2.3.5, 2.4.4
Key	**Value**	**Log Level**
Context	The WebComm command string	INFO

Name	Description	Since
MaconomyServer: MScript	Issued when an MScript command is received by the server. Requires the 'MScript' TimeLog logger to be enabled.	2.3.5, 2.4.4
Key	**Value**	**Log Level**
Context	The name of the MScript command	INFO

Container Requests

The following container request-related log entries exist:

Name	Description	Since
Container:Open	Issued when a container is opened.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO

Name	Description	Since
Container:Close	Issued when a container is closed.	2.3.4, 2.4.4
Issued when a container is closed.		
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO

Name	Description	Since
Container:Specify	Issued when a container specification is requested.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO

Name	Description	Since
Container:Lock	Issued when a container entry is locked. (deprecated, only in 2.4)	2.3.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
Restriction	The current container restriction	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Unlock	Issued when a container entry is unlocked. (deprecated, only in 2.4)	2.3.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
Restriction	The current container restriction	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Initialize	Issued when a new container entry is initialized.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
Restriction	The current container restriction (optional)	DEBUG
NewValues	'null'	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Create	Issued when a new container entry is created.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
Restriction	The current container restriction	DEBUG
NewValues	The new values to create the container entry with	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Read	Issued when a container entry is read.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
Restriction	The current container restriction	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Update	Issued when a container entry is updated.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
Restriction	The current container restriction	DEBUG
NewValues	The new values to update the container entry with	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Delete	Issued when a container entry is deleted.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
Restriction	The current container restriction	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Action	Issued when a container action is executed.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
ActionName	The name of the action	INFO
Restriction	The current container restriction	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Print	Issued when a container print action is executed.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
Restriction	The current container restriction	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Move	Issued when a container table move operation is performed.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
Operation	The move operation being performed	INFO
SourceRestriction	The source row restriction	DEBUG
TargetRestriction	The target row restriction	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Restore	Issued when a container restore operation is performed.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
RestoreData	The restore data (optional)	DEBUG
Parameters	The current container parameters	DEBUG

Name	Description	Since
Container:Restrict	Issued when a restriction is acquired for the current container entry	2.3.4, 2.4.4
Key	**Value**	**Log Level**
ContainerName	The name of the container	INFO
ForeignKeyName	The name of the foreign key that provides the restriction	DEBUG
RestrictionData	The record data to base the restriction on	DEBUG
Parameters	The current container parameters	DEBUG

Container Contributions

The following log entry breaks down container request-related time into its container contribution pre/post phases:

Name	Description	Since
Container:Contribution	Issued when a container contribution event phase (pre/post) is executed. The execution of a container contribution event is split into 3 phases: 1. A 'pre' event phase performed by the container contribution (PhaseId="PRE") 2. A delegation to the underlying container contributions 3. A 'post' event phase performed by the container contribution (PhaseId="POST") The total time spent on an event in a container contribution is the sum of the 'pre' and 'post' phase durations logged for that container contribution. The time spent in underlying container contributions is not included. A container contribution is uniquely identified by its container name, contribution id and bundle id.	2.3.5, 2.4.4

Key	Value	Log Level
ContainerName	The name of the container	INFO
EventId	The ID of the container event being processes: OPEN, CLOSE, SPECIFY, TRANSACTION, INITIALIZE, CREATE, READ, UPDATE, DELETE, ACTION, PRINT, MOVE, RESTRICT	INFO
ContributionId	The contribution ID of the container contribution	INFO
BundleId	The bundle ID of the container contribution	INFO
PhaseId	The container contribution phase being executed: PRE, POST	INFO
ActionName	The name of the action being executed, if EventId="ACTION" (optional)	INFO

Remote Procedure Calls

The following log entries are issued when RPC functions are invoked:

Name	Description	Since
MaconomyServer: Request	Issued when a call to an RPC function is performed. Required log level: DEBUG	2.3.4, 2.4.4
Key	**Value**	**Log Level**
PID	The PID of the remote server process	DEBUG
Function	The RPC function being called	DEBUG
Parameters	The parameters to the RPC function	TRACE

Name	Description	Since
MaconomyServer: Callback	Issued when an RPC callback function invocation is received from the remote server. Required log level: DEBUG	2.3.4, 2.4.4
Key	**Value**	**Log Level**
PID	The PID of the remote server process	DEBUG
Function	The RPC callback function being invoked	DEBUG
Parameters	The parameters to the RPC callback function	TRACE

Name	Description	Since
MaconomyServer: RPC	Issued when an RPC function call is executed on the remote server.	2.3.5, 2.4.4
Key	**Value**	**Log Level**
Context	The RPC function being called. The 'Enter' event will have the format: '*functionName*(*parameters*)' The 'Exit' event will have the format: '*functionName*(*parameters*)->(*returnValues*)'	INFO
PID	The PID of the remote server process	INFO

MSL Requests

The following log entry is produced by the 'MSL' TimeLog logger.

Name	Description	Since
MaconomyServer: MSL	Issued when an MSL dialog script or sub-program script is invoked	2.3.5, 2.4.4
Key	**Value**	**Log Level**
Context	The name of the MSL script	INFO

Database Requests

The following log entries are produced by the 'Database' TimeLog logger.

Name	Description	Since
MaconomyServer: Database	Issued when a database operation is invoked	2.3.4, 2.4.4
Key	**Value**	**Log Level**
Context	*Op*: *Query*	INFO
-where 'Op' can be:		
Cursor:Declare	Cursor declare operation	
Cursor:Execute	Cursor execute operation	
Cursor:Fetch[{Count}]	Cursor fetch operation. This log entry can appear in two variations: As a log entry for the execution time of a single database fetch operation As an accumulating log entry with the elapsed time since the first record was fetched from the cursor, and the total number of records fetched The accumulating log entry is only generated if the cursor is fetched until exhaustion (i.e. the last fetch operation returns 'nil')	
Cursor:Update	Cursor update operation	
Cursor:Insert	Cursor insert operation	
Cursor:Delete	Cursor delete operation	
SqlRequest	A "raw" (non-cursor) SQL request operation	

Security Log

The following log entries are produced by the 'SecurityLog' TimeLog logger.

Name	Description	Since
MaconomyServer:SecurityLog	Issued when a security log event is created. All login attempts, whether successful or not, will create a security log event.	2.3.4, 2.4.4
Key	**Value**	**Log Level**
Context	*Message [User]*	INFO
Audit	The Audit flag is always set for this event	*n/a*

Callbacks

The following callback-related log entries exist:

Name	Description	Since
~~RequestContext:Callback~~	**Deprecated:** Replaced by Callback:Function Issued when a callback is dispatched out of the server execution context.	2.3.4 only
Value	**Description**	**Log Level**
Function	The callback function	INFO
Result	The resulting value returned from the callback handler (only on the 'Exit' log entry)	INFO

Name	Description	Since
Callback:Function	Issued when a callback is dispatched out of the server execution context. The time duration logged for a callback is time that the server process has been idle, waiting for a response from the client.	2.3.5, 2.4.4
Key	**Value**	**Log Level**
Function	The callback function	INFO
Result	The resulting value returned from the callback handler (optional)	INFO

Name	Description	Since
Callback:Progress	Issued when a progress callback is dispatched out of the server execution context. The progress callback log entry covers the entire life cycle of the progress callback, i.e. from 'ShowProgress' across all 'UpdateProgress' calls to the final call to 'RemoveProgress'. This allows the log entry to exceed an entry duration threshold even though the individual progress callbacks are fast. Each 'UpdateProgress' call is logged as a 'Message'-type log entry and will contain the reported "work completed" fraction. The time spent in a progress callback is reported as 'Elapsed' rather than as 'Duration', since this time does not count towards the total execution time of the request.	2.3.5, 2.4.4

Key	Value	Log Level
Message	The progress message	INFO
Fraction	The "work completed" fraction (only for 'Message'-type entries)	INFO
Result	The resulting value returned from the callback handler (only on the 'Message' and 'Exit' log entries, and for the 'Message' entries showing the return value of the previous progress callback)	INFO

Notifications

When actions are performed in the context of a notification recalculation, the threshold log level for log entries is raised to TRACE, to avoid filling the log output with highly repetitive universe queries.

This means that further log entries for notification recalculation-related operations will only be printed if the log level of the Logback logger is also set to TRACE.

Sample Queries

In this section we show some examples of how the monitor log entries can be queried. The queries are written in a SQL-like syntax, and should be fairly easy to translate to the concrete query syntax of any capable log reporting framework.

Note: For APM configurations with multiple shortnames, you can do a wildcard query on PrincipalName rather than filtering on both Url and PrincipalName.

In the example below, the table of log entries is referred to as 'LogEntries'.

Request Execution Time

To list all requests and their execution time simply group the requests by 'RequestId' and add up the 'Duration' of the top-level log entries in each request:

```
SELECT RequestId, COALESCE(SUM(Duration),0) FROM LogEntries
   WHERE Type = 'Exit' AND LogDepth = 0
   GROUP BY RequestId;
```

Notice the use of the 'COALESCE(…)' function to normalize NULL values. In this example it's a highly theoretical possibility, but as we begin to introduce various sub-queries the consistent use of this function becomes very important.

The execution times returned by the query above will include time spent waiting for user callbacks, which is typically not relevant from a performance perspective. To take this into account the query can be expanded to subtract the time spent in callbacks:

```
SELECT RequestId, TotalTime, CallbackTime, TotalTime-CallbackTime AS RequestTime FROM
   (SELECT
      RequestId,
      TotalTime = (SELECT COALESCE(SUM(Duration),0) FROM LogEntries
         WHERE Type = 'Exit' AND LogDepth = 0 AND RequestId = L.RequestId),
      CallbackTime = (SELECT COALESCE(SUM(Duration),0) FROM LogEntries
         WHERE Type = 'Exit' AND NAME LIKE 'Callback:%' AND RequestId = L.RequestId)
   FROM LogEntries L
   GROUP BY RequestId) Result;
```

The query can be expanded further to also report the time spent on database operations:

```
SELECT RequestId, TotalTime, CallbackTime, DatabaseTime, TotalTime-CallbackTime AS
RequestTime FROM
   (SELECT
      RequestId,
      TotalTime = (SELECT COALESCE(SUM(Duration),0) FROM LogEntries
         WHERE Type = 'Exit' AND LogDepth = 0 AND RequestId = L.RequestId),
      CallbackTime = (SELECT COALESCE(SUM(Duration),0) FROM LogEntries
         WHERE Type = 'Exit' AND NAME LIKE 'Callback:%' AND RequestId = L.RequestId),
      DatabaseTime = (SELECT COALESCE(SUM(Duration),0) FROM LogEntries
         WHERE Type = 'Exit' AND NAME LIKE '%:Database' AND RequestId = L.RequestId)
```

```
   FROM LogEntries L
   GROUP BY RequestId) Result;
```

Workspace Requests

The following query example shows how to extract information on all Workspace Client requests:

```
SELECT
   RequestId, WorkspaceName, PaneRequest, Duration
FROM LogEntries
WHERE Type = 'Exit' AND Name LIKE 'Workspace:%';
```

The only caveat in this query is that it only shows the time spent on the workspace-specific part of the request, which is not necessarily representative for the total execution time of the request. Luckily the query can easily be amended to also calculate the total execution time:

```
SELECT
   RequestId, WorkspaceName, PaneRequest, Duration,
   TotalTime = (SELECT COALESCE(SUM(Duration),0) FROM LogEntries
      WHERE RequestId = L.RequestId AND LogDepth = 0)
FROM LogEntries L
WHERE Type = 'Exit' AND Name LIKE 'Workspace:%';
```

WebService and WebDaemon Requests

The following query examples show how to extract information on WebService and WebDaemon requests. These queries follows the same pattern as for Workspace Requests discussed above, except that the additional inner select used to obtain the total execution time is not necessary, since these request types at the time of writing are always logged at top-level (`LogDepth = 0`):

```
SELECT
   RequestId, Method, Url, Query, Duration
FROM LogEntries
WHERE Type = 'Exit' AND Name LIKE 'WebService:%';
SELECT
   RequestId, Method, Url, Query, Request, RemoteAddr, Duration
FROM LogEntries
WHERE Type = 'Exit' AND Name LIKE 'WebDaemon:%';
```

Login Auditing

The following query examples show how to extract information about user logins.

If the ‘SecurityLog’ TimeLog logger is enabled am audit-level log event is issued for every login attempt:

```
SELECT
   Context, PID
FROM LogEntries
WHERE Type = 'Exit' AND Name LIKE '%:SecurityLog';
```

In addition, login and logout requests from all Maconomy 2.x clients (Workspace Client, REST-API) will trigger ‘Auth’ events in the Coupling Service:

```
SELECT
   LoginRules, PrincipalName
FROM LogEntries
WHERE Type = 'Exit' AND Name = 'Auth:Login' OR Name = 'Auth:Logout';
```

Active Request

To identify requests which have not yet completed, we must find requests that have a top-level ‘Enter’ entry without a matching ‘Exit’ entry. This can be as simple as selecting all ‘Enter’ entries with log depth zero and subtract all ‘Exit’ entries with log depth zero:

```
SELECT RequestId FROM LogEntries WHERE Type = 'Enter' AND LogDepth = 0
EXCEPT
SELECT RequestId FROM LogEntries WHERE Type = 'Exit' AND LogDepth = 0
```

Troubleshooting

This section provides details to help you troubleshoot issues and errors in the configuration and function of the APM logging framework for your Maconomy system.

Log Entries

Issue

Log entries are not written to maconomy-monitor.log

Recommendation

- Check that log.monitor.enabled is set to true in monitor.ini.
- Check that the MONITOR log appender configuration is present in logback.xml, and that it is configured to write to the 'maconomy-monitor.log' log file.
- Check that a logger is configured in logback.xml on the APM framework class (com.maconomy.util.logger.McMonitorLogger) referencing the MONITOR log appender. The log level of this logger should be either INFO, DEBUG or TRACE.
- Inspect the Coupling Service main log file (maconomy.log) for any errors or log entries from the APM framework class (McMonitorLogger). Whenever any of the monitor log related entries are changed in monitor.ini an INFO-level entry will be written with the new configuration.

Issue

Log entries are not written to maconomy-monitor.log (continued)

Recommendation

- Check that the minimum-time and minimum-exit-time settings in the log.monitor section are set to the desired values according to the time unit configured in time-unit.
- Try to add one or more of the top-level log entry types to the list of auditable events. This will cause these events to be logged even if they do not satisfy the configured minimum time requirements:

  ```
  log.monitor.audit = Workspace:*, WebService:*, WebDaemon:*
  ```

- Inspect the Coupling Service main log file (maconomy.log) for any errors or log entries from the APM framework class (McMonitorLogger).

Field Values

Issue

Not all the expected field values are included on the log entries

Recommendation

- Some of the APM loggers in the Coupling Service will only include certain field values at move verbose log levels. If the logger configuration for com.maconomy.util.logger.McMonitorLogger in

logback.xml is set to INFO or DEBUG you have the option to increase the verbosity to DEBUG, TRACE or ALL to get more fields included from these loggers.

- If the missing field values still do not show up consider if you are on a Maconomy version where the field in question is not yet available.
- Consult the Log Events section for information on the version availability and required log level of each field in the different log events.

Splunk

Issue

Mac_monitor_idx or jmx_idx or host_disk_idx indexes are empty after the Application has been deployed and configured on the Splunk Forwarder

Recommendation

After the deploy of the application to the forwarder, all the ERROR will be logged on splunkd.log file on the forwarder var/log/splunkd/ folder.

Appendix: Splunk

This section provides information on using the third-party tool, Splunk, to report on your APM logs, including:

- Prerequisites
- Installation
- Configuration

Note: Splunk is a third party APM Tool which requires licensing. The use of this tool is not covered by Deltek product support or maintenance.

For comprehensive Splunk documentation, refer to Splunk documentation and training.

https://www.splunk.com/en_us/training.html

Prerequisites

Forwarder Configuration

Before you install and configure Splunk, you must have:

- Splunk Universal Forwarder v8+
- Java 1.8+
- Python 3
- SPLUNK_HOME environment variable set to the SplunkUniversalForwader dir.
    - If you run the forwarder on a docker container you can create your own image or use andreadeltek/universalforwarder.
    - This particular image comes with Python and Java already installed.

Example

An example of a docker-compose to start a universal forwarder:

```
version: '3.7'

services:
  splunkuniversalforwarder:
    container_name: uf
    hostname: hostname
    image: andreadeltek/universalforwarder:uf8041_j11_p373
    environment:
      - "SPLUNK_START_ARGS= --accept-license"
      - "SPLUNK_PASSWORD=changeme"
    volumes:
      - ./Docker/conf/UF/host.yaml:/opt/splunkforwarder/host.yaml
```

```
      -
./Docker/conf/UF/localConfig/deploymentclient.conf:/opt/splunkforwarder/etc/system
/local/deploymentclient.conf
      - ./Docker/defaults/default.yml:/tmp/defaults/default.yml
# Inputs.conf and outputs.conf can be omitted if the files were previously set up
in the deployment server
      -
./Docker/conf/UF/mac_monitor/local/inputs.conf:/opt/splunkforwarder/etc/apps/mac_m
onitor/local/inputs.conf
      -
./Docker/conf/UF/mac_monitor/local/outputs.conf:/opt/splunkforwarder/etc/apps/mac_
monitor/local/outputs.conf
      - ./Docker/Maconomy/coupling_logs:/opt/splunkforwarder/share/data
      - ./Docker/logs/UF:/opt/splunkforwarder/var/log/splunk
    networks:
      - maconomy-network

networks:
  maconomy-network:
    name: maconomy-internal-network
```

Summary Indexes

To speed up searches and improve dashboard efficiency, the application defines and uses five different summary indexes. Each index is populated by a report set to run each night between midnight and 1 a.m.

- **si_container_perf** —populated with information related to container requests, the responsable report name is: "SI - Container performance"
- **si_daily_request_count** —holds information related to the number of request received in a day, the responsable report name is: "SI - Daily request count"
- **si_daily_session_count** —holds information related to session logged in a day, the responsable report name is: "SI - Daily session duration and count"
- **si_daily_request_values** — holds information related to each request logged in a day breaking down each duration for server, db and callback time, the responsable report name is: "SI - Daily single request values"
- **si_user_login** —collect the login information logged during the day, the responsable report name is: "SI - User login"

Fill Summary Index Gaps

If Splunk does not directly receive events from the Maconomy Server and the data is uploaded on demand or Splunk has already stored some events prior to application installation, follow the example below to fill the gaps in the summary indexes.

Example:

```
# Example to run all 5 summary indexes to fill gaps between 15th of June and the current day:
./splunk cmd python fill_summary_index.py -app mac_monitor -names "SI - User login","SI - Daily single request values","SI - Daily session duration and count","SI - Daily request count","SI - Container performance" -et 1592222400 -lt now -dedup true -j 2 -auth admin:password
```

Configure Splunk APM Application

To configure Splunk APM application configuration for Universal Forwarder, you must do the following:

- Configure Server.conf
- Configure Inputs.conf
- Configure Outputs.conf
- Configure host yaml

Server.conf

To configure server.conf:

1. Go to:

   /splunkforwarder/etc/system/local/server.conf

2. The python script is developed to work with Python 3, so you must set the Splunk Forwarder to use Python3 as interpreter. To do so, add the following line to the General stanza:

   ```
   python.version = force_python3
   ```

For more information about server.conf specification visit Splunk documentation at:

https://docs.splunk.com/Documentation/Splunk/latest/Admin/Serverconf

Inputs.conf

This file is used to specify which inputs you want to monitor and send to Splunk. By default, you must define three stanzas.

To configure inputs.conf:

1. Go to:

   /splunkforwarder/etc/apps/mac_monitor/local/inputs.conf

2. Define JVM stanza.

3. Define Mac Monitor stanza.
4. Disk I/O stanza (either Windows or Linux)

For more information about inputs.conf specification see Splunk documentation:
https://docs.splunk.com/Documentation/Splunk/7.2.6/Admin/Inputsconf

JVM stanza

```
[jmx://localhost_jvm]
# path to the config.xml file containing he properties we want to import from the
#jvm stats
config_file = config.xml
# the number of seconds between each read of the jvm properties
polling_frequency = 30
sourcetype =  jmx
index = jmx_idx
disabled = 0
```

Mac Monitor stanza

```
# the log file we want to monitor and collect events from
[monitor://$SPLUNK_HOME/share/data/maconomy-monitor.log]
# A comma-separated list of tcpout group names. Default: *
_TCP_ROUTING = *
# sourcetype to associate to all the events
sourcetype = mac_monitor
# the index name where events will be stored
index = mac_monitor_idx
```

Disk I/O stanza

If the Universal Forwarder is directly installed on Windows, use the following stanza.

Windows:

```
[perfmon://LocalPhysicalDisk]
# number in seconds between each read
interval = 0
# to also log zeros
showZeroValue = 1
# has to be PhysicalDisk
object = PhysicalDisk
```

```
# counters we are interested in importing to Splunk
counters = Disk Bytes/sec; % Disk Read Time; % Disk Write Time; % Disk Time
instances = *
disabled = 0
index = host_disk_idx
```

If the Universal Forwarder is installed on Linux or is running as Container in Docker, use the following stanza.

Linux:

```
[script://./bin/iostat.sh]
# number in seconds between each read
interval = 60
sourcetype = host_disk_io
source = hostname
index = host_disk_idx
disabled = 0
```

Outputs.conf

This file is used to specify how the forwarders send the data to Splunk instances.

To configure outputs.conf:

1. Go to:

   /splunkforwarder/etc/apps/mac_monitor/local/outputs.conf

2. Refer to the example below.

For more information about outputs.conf specification, see Splunk documentation:

https://docs.splunk.com/Documentation/Splunk/7.2.6/Admin/Outputsconf

```
[tcpout]
defaultGroup= splunk_instance

[tcpout:splunk_instance]
server=10.4.2.7,10.4.2.9
```

Host yaml

To connect the application to the JVM and pull information from the JVM, create a host.yaml file in the root of Splunkforwarder.

To connect to the JMX agent:

1. Create the file:

   /splunkforwarder/host.yaml

2. Inside the yaml file, specify three properties: host, description, port (JVM_HOST, JVM_DESCRIPTION and JVM_PORT).

```
# the container name of the Maconomy server when running in Docker, or the IP
# address of the host if Maconomy is running as a service on Windows
JVM_HOST: maconomy-core
# THe Hostname of the server the Universal Forwarder is running on
JVM_DESCRIPTION: Hostname
# the port to use to connect to the JVM
JMX_PORT: 9090
```

If this file is not defined, the application will try to connect to the JMX agent using the default values:

```
Host : localhost
Description: maconomy_server
Port: 9090
```

Run Splunk Universal Forwarder (UF) in Docker

Deploying the UF in a Docker container is recommended because the UF image comes with all of the software and settings the APM requires. This removes the need to manually install dependencies on the host machine.

There are two ways to run the UF. Both require pulling the UF image. You can run this either in an environment where Maconomy is running in a Docker container or where Maconomy is running as a service on Windows. Both methods are described below.

Notes:

- **DCO** — This section is intended for DCO use. This section includes an official Splunk Universal Forwarder image and that the custom image is based on the one with the addition of the software needed by the app to run (python and Java)
- **UF image delivered with software** —Additionally, the UF image is delivered with all the software the application requires to run and the tag describes the bundled UF, java and python versions.

  For example:

  Uf804_j11_p373

  means this image comes with:

  Universal Forwarder 8.0.4, Java 11 and Python 3.7.3
- **Open JDK** — All images use Open JDK.

Configure UF Runtime

To configure the UF Docker image,supply a default.yml file. This file is used to define a standard set of variables that control how Splunk is set up.

To generate the file automatically:

1. Generate the file automatically with:

   ```
   docker run --rm --ur splunk/universalforwarder:8.0.4.1 create-defaults
   > default.yml
   ```

2. To generate server.conf with the use of python3, you must modify default.yml. To do so, add the following section:

```
[...]
      splunk:
            conf:
            - key: server
            value:
                  directory: /opt/splunkforwarder/etc/system/local
                  content:
                        general:
                              python.version: force_python3
[...]
```

For more information on how create custom configuration for splunk docker image:

https://splunk.github.io/docker-splunk/ADVANCED.html#create-custom-config

To deploy in a Splunk UF:

1. Install Docker on the machine:

   https://hub.docker.com/editions/community/docker-ce-desktop-windows

2. Pull the UF image from Docker Hub:

   https://hub.docker.com/repository/docker/andreadeltek/universalforwarder

   Note:

   The image is delivered with all the software the application requires to run and the tag describes the bundled UF, java and Python versions. For example: uf8041_j11_p373 means this image comes with Universal Forwarder 8.0.4.1, Java 11 and Python 3.7.3. Note that all images use Open JDK.

3. **If you run Maconomy in a container**, follow the Dockerised Maconomy instructions.

   or

   If you run Maconomy as a service on Windows (in other words, not in a Docker container), follow the Maconomy Service on Windows instructions to install the UF.

Dockerized Maconomy

To install the UF if you run Maconomy in a container:

1. To connect to the Maconomy JMX running in the Docker container, edit a Maconomy configuration file. Add the following line to the file:

   `<shortname>/CouplingService/servicewrapper/conf/wrapper.equinox.conf:`

2. Add the line below to append to the wrapper.equinox.conf file.

```
wrapper.java.additional.10=-Djava.rmi.server.hostname=<Mac_Container>
```

3. Replace `<Mac_Container>` with the name of the Docker container you use to run Maconomy, such as *maconomy-core*.
4. Update the Host yaml and set the JVM_HOST to the Maconomy Docker container name.
5. Start the UF. You can use docker-compose to run the UF.

   In this case the UF container will run on the same network as the Maconomy and the database containers. Because they are all on the same network there is no need to map any ports.

   See the example below.

6. Once the UF starts, check splunkd.log for any issues.

Example: maconomy-network

In the following example, "maconomy-network" is the network used by the Maconomy and database containers.

```
version: '3.7'


services:
```

```
  splunkuniversalforwarder:
    container_name: uf
    hostname: andrea_osx
    image: andreadeltek/universalforwarder:uf804_j11_p373
    environment:
      - "SPLUNK_START_ARGS= --accept-license"
      - "SPLUNK_PASSWORD=changeme"
    volumes:
      - /Docker/conf/UF/host.yaml:/opt/splunkforwarder/host.yaml
- ./Docker/defaults/default.yml:/tmp/defaults/default.yml
      -
/Docker/conf/UF/localConfig/deploymentclient.conf:/opt/splunkforwarder/etc/system/
local/deploymentclient.conf
      - /Docker/Maconomy/coupling_logs:/opt/splunkforwarder/share/data
      - /Docker/logs/UF:/opt/splunkforwarder/var/log/splunk
      # The following can be omitted if you are using a deployment server with the
configuration already in the application.
      -
/Docker/conf/UF/mac_monitor/local/inputs.conf:/opt/splunkforwarder/etc/apps/mac_mo
nitor/local/inputs.conf
      -
/Docker/conf/UF/mac_monitor/local/outputs.conf:/opt/splunkforwarder/etc/apps/mac_m
onitor/local/outputs.conf
    networks:
      - maconomy-network

networks:
  maconomy-network:
    name: maconomy-internal-network
```

Maconomy as a Service on Windows

To install the UF if you run Maconomy as a service on Windows:

1. To allow the UF to connect to the JMX, edit the file at:
   `<shortname>/CouplingService/servicewrapper/comf/wrapper.equinox.conf`
2. Add the following:

```
wrapper.java.additional.10=-Djava.rmi.server.hostname=192.168.130.154
```

where

192.168.130.154

is the IP address of the machine.

3. Restart the wrapper to apply the changes.
4. Update the Host yaml to set the JVM_HOST to the IP address of the Windows host.
5. Start the UF. You can use docker-compose to start the UF. See the example below.
6. Once the UF starts, check splunkd.log for any issues.

Example: Docker-compose file to start UF with Maconomy running as a Windows Service.

```
version: '3.7'

services:
  splunkuniversalforwarder:
    container_name: uf
    hostname: andrea_win_uf
    image: andreadeltek/universalforwarder:uf804_j11_p373
    environment:
      - "SPLUNK_START_ARGS= --accept-license"
      - "SPLUNK_PASSWORD=changeme"
    volumes:
      - C:\SplunkForwarder\host.yaml:/opt/splunkforwarder/host.yaml
- C:\SplunkForwarder\defaults\default.yml:/tmp/defaults/default.yml
      -
C:\SplunkForwarder\conf\deploymentclient.conf:/opt/splunkforwarder/etc/system/loca
l/deploymentclient.conf
      -
C:\maconomy\w_19_0\CouplingService\log\coupling:/opt/splunkforwarder/share/data
      - C:\SplunkForwarder\logs:/opt/splunkforwarder/var/log/splunk
      # The following can be omitted if you are using a deployment server with the
configuration already in the application.
      -
C:\SplunkForwarder\conf\inputs.conf:/opt/splunkforwarder/etc/apps/mac_monitor/loca
l/inputs.conf
```

```
      -
C:\SplunkForwarder\conf\outputs.conf:/opt/splunkforwarder/etc/apps/mac_monitor/loc
al/outputs.conf
```

About Deltek

Better software means better projects. Deltek is the leading global provider of enterprise software and information solutions for project-based businesses. More than 23,000 organizations and millions of users in over 80 countries around the world rely on Deltek for superior levels of project intelligence, management and collaboration. Our industry-focused expertise powers project success by helping firms achieve performance that maximizes productivity and revenue. www.deltek.com