Bundesverwaltungsamt

IsyFact

Logging-Konzept

Version 3.4
07. Oktober 2015

IsyFact

„Logging-Konzept“
des Bundesverwaltungsamts ist lizenziert unter einer
Creative Commons Namensnennung 4.0 International Lizenz.

Die Lizenzbestimmungen können unter folgender URL heruntergeladen
werden: http://creativecommons.org/licenses/by/4.0

Ansprechpartner:

Referat Z II 2
Bundesverwaltungsamt
E-Mail: isyfact@bva.bund.de
Internet: www.isyfact.de

Dokumentinformationen

Dokumenten-ID:	Konzept_Logging.docx

Inhaltsverzeichnis

1. Einleitung 6

2. Aufbau und Zweck des Dokuments 7

3. Anforderungen an das Logging 9

4. Vorgaben zur Logerstellung 12

4.1. Log-Inhalte 12

4.1.1 Entitäten 12

4.1.2 Log-Level, Log-Kategorie und Ereignisschlüssel 15

4.1.3 Korrelations-ID 15

4.1.4 Nachricht 15

4.2. Logausgabe 16

4.2.1 Logformat 16

4.2.2 Logdateien, Log-Rotation und Komprimierung 17

4.3. Logging-Framework 17

4.3.1 Logging-Fassade 17

4.3.2 Log-Aufbereitung 19

4.3.3 Hilfsklassen 20

4.3.4 Logging externer Bibliotheken 21

5. Aufbau der Log-Infrastruktur 22

5.1. Shipper (Logstash) 24

5.1.1 Log-Quellen (Input) 24

5.1.2 Filter 25

5.1.3 Log-Ziele (Output) 25

5.2. Puffer (Redis) 25

5.3. Indexer (Logstash) 25

5.3.1 Filter 26

6. Log-Auswertung 29

6.1. Allgemeine Hinweise 29

6.1.1 Kein Log-Level FATAL in den Logs 29

6.2. Allgemeine Inhalte jedes Logeintrags 30

6.3. Register Factory-konforme Anwendungen 30

6.4. Register Factory-konforme Anwendungen (vor Logging-Konzept Version 3.0) 31

6.5. Tomcat access-log 31

6.6. Wrapper-Log 32

6.7. Apache access-log und error-log 32

6.8. Mailserver-log 33

6.9. Suchverfahren-Log 33

6.10. logstash-log 37

7. Anhang 38

7.1. Glossar 38

7.2. Konfigurationsvorlagen 40

7.2.1 Logstash (Shipper und Indexer) 40

8. Quellenverzeichnis 45

9. Abbildungsverzeichnis 46

1. Einleitung

Dieses Dokument beschreibt die konzeptionellen Grundlagen des Loggings in der Register Factory. Es adressiert die einheitliche Erstellung von Logs in den entwickelten Anwendungen sowie die weitere Verarbeitung der erstellten Logs (auch der von Drittsystemen) durch die Log-Infrastruktur. Es bildet damit die Grundlage der technisch und fachlich einheitlichen Umsetzung des Loggings in der gesamten Anwendungslandschaft und die davon abhängige effiziente Auswertung der Logeinträge.

Die daraus abgeleitete Umsetzung des Loggings im Rahmen einer Anwendungsentwicklung und die möglichen Auswertungen der Logs sind in den Nutzungsvorgaben *Logging* (siehe [NutzungsvorgabenLogging]) beschrieben. Zu beachten ist, dass die Nutzungsvorgaben an einen größeren Leserkreis gerichtet sind und verschiedene Aspekte sowohl für das Logging-Konzept als auch für die Nutzungsvorgaben relevant sind. Um Redundanzen zu vermeiden, werden diese Aspekte alle in den Nutzungsvorgaben beschrieben und im vorliegenden Dokument darauf verwiesen.

2. Aufbau und Zweck des Dokuments

Ziel des Dokuments ist die Definition eines ganzheitlichen Standards für das Logging, von Erstellung (Inhalt und Ausgabe) der Logs durch die Systeme, über Verarbeitung der Logs durch die Infrastruktur, bis hin zur anwendungslandschaftsübergreifenden Auswertung der Logeinträge. Hierbei werden die Anforderungen von Betrieb, Test, Softwareentwicklungsreferat und Softwarelieferanten berücksichtigt.

Das Dokument ist entsprechend dieser Zielsetzungen in die folgenden Kapitel untergliedert:

Im Kapitel **Anforderungen an das Logging** werden die Zielsetzungen und Anforderungen an das Logging definiert, die die Grundlage für die nachfolgenden Kapitel bilden.

Im Kapitel **Vorgaben zur Logerstellung** werden die Vorgaben zur Erstellung der Logs innerhalb der Anwendungen gemäß der Register Factory definiert.

Im Kapitel **Aufbau der Log-Infrastruktur** wird die Log-Infrastruktur beschrieben, die zur Verteilung, Verarbeitung, Auswertung und Archivierung der erstellen Logdateien eingesetzt wird.

Im Kapitel **Log-Auswertung** werden die Informationen beschrieben, die letztendlich im Auswertungstool zur Verfügung stehen und damit die Grundlagen zur Durchführung von Analysen darstellen.

Im Anhang **Glossar** werden die Begriffe und Abkürzungen definiert, die spezifisch für das Themenfeld „Logging“ sind.

Abgrenzung:

Logging ist ein querschnittliches Thema einer Anwendungslandschaft mit Schnittstellen zu zahlreichen weiteren Themen. Folgende Abgrenzungen werden dabei getroffen:

- In diesem Dokument werden die Grundlagen geschaffen, um eine effiziente Auswertung der Logs zu ermöglichen. Die eigentliche Auswertung mit Hilfe von Analysetools ist nicht Teil des Dokuments und wird in [NutzungsvorgabenLogserver] beschrieben.

- Konfiguration von Drittsystemen: Die Konfiguration des Loggings von Systemen, die nicht mit der Register Factory entwickelt wurden (bspw. Apache Webserver, Apache Tomcat, Mailserver), wird in den Konzepten der einzelnen Systeme adressiert. Das Logging-Konzept definiert jedoch, welche Informationen durch die einzelnen Systeme bereitgestellt werden (Kapitel 6) und wie diese durch die Log-Infrastruktur verarbeitet und somit in die anwendungslandschaftsübergreifende Analyse der Logs integriert werden können (Abschnitt 5.1).

- Monitoring, Logging, Protokollierung: Unter „Logging“ wird in diesem Dokument „Logging zu technischen Zwecken verstanden“. Anforderungen bezüglich der Protokollierung (das Mitschreiben von Informationen zu durchgeführten Geschäftsvorfällen auf Grund fachlicher Anforderungen), werden im Konzept *Protokollierung* [ProtokollierungKonzept]) behandelt.

  Das Monitoring zur betrieblichen Überwachung der Anwendungslandschaft und frühzeitigen Eskalation von Problemen ist im Konzept *Überwachung und Konfiguration* [ÜberwachungKonfigKonzept] beschrieben.

  In folgender Tabelle werden die drei verschiedenen Disziplinen detailliert voneinander abgegrenzt:

	Logging	**Monitoring/Überwachung**	**Protokollierung**
erfasst werden	Anwendungszustände, Fehlermeldungen	Anwendungszustände, Fehlermeldungen	Aktionen innerhalb der Anwendung
wo erfasst	Log-Server Registerportal	zentrales Monitoring System, z.B. Nagios	Datenbank Protokollrecherche
Auswertung erfolgt	kontinuierlich	kontinuierlich	bei Verdacht auf security incident
Auswerteart	automatisch - inklusive Korrelation von Ereignissen und ggf. Meldung an Überwachungs-System anlassbezogen - Prüfung einzelner Logdateien bei Störung oder Verdacht auf security incident	automatisch - inklusive Alarmierungsfunktionen	nur manuell
Einschränkung der Auswertung auf	Administratoren	Administratoren	dedizierte Fachadministratoren der Protokollrecherche
Zielsetzung	zeitnahe Feststellung von Störungen und potentiellen Angriffen	Überwachung des Betriebszustands der Anwendungen plus Alarmierung bei Störungen	Nachvollziehbarkeit, Vorfallsaufklärung

3. Anforderungen an das Logging

Im Folgenden werden die Ziele und Anforderungen an das Logging definiert. Diese bilden die Grundlage für die Ausgestaltung des Loggings in den darauffolgenden Kapiteln.

Anforderungen an die Log-Inhalte und deren Auswertungsmöglichkeiten

- **Erkennen, Behandeln und Beheben von Fehlern:**

    1. **Fehlererkennung**: Sämtliche technischen Ausnahmen müssen erkannt werden können und in den Logs hinterlegt werden.

    2. **Fehlereskalation**: Für jeden Fehler muss klar ersichtlich sein, wie schwerwiegend er ist, durch welche IT-Komponente er ausgelöst wurde und durch wen (welche Nutzergruppe – bspw. Betrieb, Entwicklungsabteilung etc.) er zu behandeln ist.

    3. **Fehleranalyse**: Es müssen Informationen zur Ermittlung der Fehlerursache – im jeweils benötigten Detaillierungsgrad – verfügbar sein.

- **Analyse und Nachvollziehen von Systemereignissen und des Systemzustands:**

    1. **Systemereignisse und –zustand (Beschreibung der Systemereignisse)**: Wichtige Systemereignisse und Zustände müssen nachvollzogen werden können.

    2. **Systemengpässe / Laufzeiten**: Systemengpässe müssen erkannt werden können. Dazu müssen Laufzeiten relevanter Operationen zur Verfügung gestellt werden.

    3. **Erkennen von Anomalien**: Anomalien sowohl im Nutzungs- als auch im Systemverhalten müssen erkannt werden können. Dies beinhaltet auch die Anforderung der Erkennung von kritischen Aktivitäten, die durch Anomalien aufgedeckt werden können (bspw. Logins außerhalb der Geschäftszeiten).

    4. **Ermitteln von technischen Nutzungsstatistiken**: Es müssen technische Nutzungsstatistiken (bspw. Anzahl der Aufrufe einer bestimmten Schnittstelle oder Komponente) ermittelt werden können.

- **Weitere systemspezifische Auswertungen**: Für jedes System können weitere systemspezifische Anforderungen an die Auswertbarkeit der Logs definiert werden, das Logging muss hierfür entsprechende Erweiterungsmöglichkeiten vorsehen.

Anforderungen an die Logausgabe und das Logging-Framework

- **Technische Vereinheitlichung des Loggings**: Das Logging muss in allen Systemen, die gemäß der Register Factory entwickelt werden, technisch einheitlich umgesetzt werden. Dies umfasst die folgenden Teilaspekte:

    - **Einheitliche Strukturierung der Ausgaben**: Ausgaben werden in einer fest vorgegebenen strukturierten Form erstellt, um die einheitliche Verarbeitung und Auswertung der Einträge zu gewährleisten.

    - **Einheitliche Logging-Konfiguration**: Die Logger in allen Anwendungen werden mit den gleichen technischen Mitteln konfiguriert.

    - **Anpassung der Logging-Konfiguration zur Laufzeit (ohne Neustart der Anwendung)**: Anpassung der Konfiguration im Testsystem zur Laufzeit der Anwendung und Anpassung der Log-Level in der Produktionsumgebung bei Bedarf.

    - **Keine Abhängigkeit zwischen Anwendungen**: Jede Anwendung besitzt eigene Logdateien und eigene Mechanismen zum Schreiben der Logs. Es gibt keine gemeinsam genutzten Logdateien und keine zentrale Anwendung zum *Erstellen* der Logs.

    - **Eindeutige Zuordnung von Logdateien**: Logdateien müssen eindeutig einer Anwendung zugeordnet werden können.

    - **Es wird nur ein Logging-Framework verwendet**: Alle Anwendungen müssen das gleiche Logging-Framework verwenden (isy-logging auf Basis von logback).

    - **Minimale Auswirkungen auf Performanz:** Das Logging darf die Performanz der Anwendung nur minimal beeinträchtigen.

<u>Anforderungen an die Log-Infrastruktur</u>

- **Aufbau einer zuverlässigen und effizienten Log-Infrastruktur**

    - **Einfache Backupmöglichkeit:** Es muss ein einfaches Backup der Logs durch den Betrieb sichergestellt werden.

    - **Zentraler Auswertungsserver:** Zur einfachen Auswertung der Logeinträge wird ein zentraler Auswertungsserver eingesetzt.

    - **(Nahezu) Echtzeit-Auswertung**: Die Auswertung der Logs (im Auswertungsserver) muss nahezu in Echtzeit erfolgen können.

- **Datenschutz**: Falls fachliche Daten in den Logs enthalten sind, so müssen diese gemäß ihres Schutzbedarfs behandelt werden.

- **Robustheit**: Ein Ausfall der Log-Infrastruktur darf keine Auswirkungen auf Anwendungen haben. Es dürfen keine Logs verloren gehen.

- **Automatisierte Auswertung**: Die Log-Infrastruktur muss Logs automatisiert auswerten und daraus Aktionen ableiten können, z.B. zur Alarmierung.

Anforderungen an die Umsetzung

- **Komplexität**

  - **Möglichst geringer Aufwand**: Der Aufwand zur Umsetzung des Loggings im Rahmen der Anwendungsentwicklung muss möglichst gering gehalten werden.

4. Vorgaben zur Logerstellung

In diesem Abschnitt werden sämtliche Aspekte zur Erstellung von Logeinträgen – dem eigentlichen „Logging“ – in den Anwendungen betrachtet und einheitliche Vorgaben definiert. Dabei findet eine Standardisierung auf Ebene der Inhalte der Logs (Abschnitt 4.1), der Ausgabe der Logs (Abschnitt 4.2) und des eingesetzten Logging-Frameworks (Abschnitt 4.3) statt.

4.1. Log-Inhalte

Die Inhalte der Logeinträge sind standardisiert, um eine effiziente Auswertung der Logs zu ermöglichen. Ein Logeintrag ist dabei wie eine fachliche Entität des Loggings, mit klar definierten Attributen zu verstehen – analog zu einer persistenten Entität des fachlichen Datenmodells einer Anwendung.

4.1.1 Entitäten

Um zu verdeutlichen, dass es sich bei Logeinträgen um Entitäten – und nicht einfach nur einen „Freitext“ – handelt, wird der Inhalt eines Logeintrags in Abbildung 1 in Form von Entitäten dargestellt.

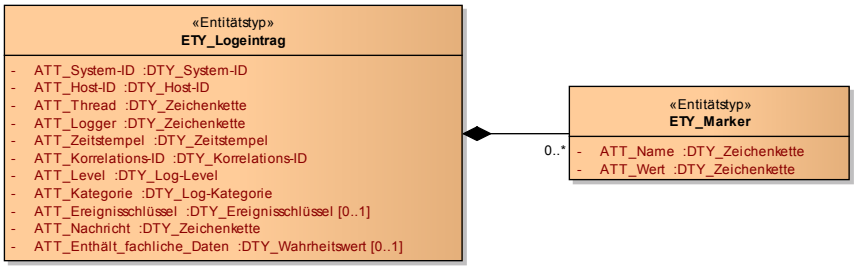

Abbildung 1: ETY_Logeintrag und ETY_Marker

Jeder Logeintrag muss gewisse Standardinformationen enthalten. Diese sind in der Entität ETY_Logeintrag zusammengefasst. Je nach Zweck des Eintrags müssen jedoch noch weitere auswertbare Informationen (bspw. die Verarbeitungszeit eines Aufrufs) hinterlegt werden. Dies geschieht in Form von Markern (ETY_Marker), die eine generische Ergänzung des Logeintrags um Name/Wert-Paare ermöglichen.

Die Inhalte der Entitäten werden im Folgenden beschrieben.

4.1.1.1 ETY_Logeintrag

Kurzbeschreibung	Dieser Entitätstyp enthält Informationen eines geloggten Ereignisses.

Feld	**Datentyp**	**Bedeutung**
ATT_System-ID	DTY_System-ID	Eindeutige Identifikation des Systems, durch das der Logeintrag erstellt

Feld	Datentyp	Bedeutung
		wurde.
ATT_Host-ID	DTY_Host-ID	Eindeutige Identifikation des Hosts, auf dem das System betrieben wird, durch das der Logeintrag erstellt wurde.
ATT_Thread	DTY_Zeichenkette	Name des Threads (bspw. „`main`“).
ATT_Logger	DTY_Zeichenkette	Name des Loggers (absoluter Pfad der Klasse).
ATT_Zeitstempel	DTY_Zeitstempel	Zeitpunkt der Erstellung des Logeintrags.
ATT_Korrelations-ID<1..n>	DTY_Korrelations-ID	Korrelations-ID des Aufrufs (siehe Abschnitt 4.1.3)
ATT_Level	DTY_Log-Level	Log-Level, welches dem Logeintrag zugeordnet wird (siehe Abschnitt 4.1.2).
ATT_Kategorie	DTY_Log-Kategorie	Kategorisierung des Logeintrags in Bezug auf dessen Zweck. (siehe Abschnitt 4.1.2).
ATT_Ereignisschlüssel	DTY_ Ereignisschlüssel	Eindeutige Identifikation des Ereignisses und dem damit verbundenen Zweck der Erstellung des Logeintrags (siehe Abschnitt 4.1.2).
ATT_Nachricht	DTY_Zeichenkette	Nachricht, welche das Ereignis des Logeintrags *menschenlesbar* beschreibt (siehe Abschnitt 4.1.4).
ATT_Enthaelt_fachliche_Daten	DTY_Wahrheitswert	Wahrheitswert, der angibt, ob der Logeintrag datenschutzrelevante Daten (vgl. Abschnitt 4.1.4.1) enthält.

Tabelle 1: Inhalte eines Logeintrags

4.1.1.2 ETY_Marker

Kurzbeschreibung	Dieser Entitätstyp ermöglich es, dem Logeintrag weitere Attribute in Form von Name/Wert-Paaren hinzuzufügen.

Feld	Datentyp	Bedeutung
ATT_Name	DTY_Zeichenkette	Bezeichnung des Markers.
ATT_Wert	DTY_Zeichenkette	Wert des Markers

Tabelle 2: Inhalte eines Markers

4.1.1.3 Datentypen

Datentyp	Basistyp	Bedeutung	Wertebereich
DTY_System-ID	Zeichenkette	Eindeutige Identifikation eines Systems.	Bspw.: `XYZRG`, `XYZGA`, `QKSVZ`
DTY_Host-ID	Zeichenkette	Eindeutige Identifikation eines Hosts/Servers.	Bspw.: `appserver01`
DTY_Zeitstempel	Datum	Zeitpunkt.	ISO8601-Format inklusive Zeitzone: `yyyy-MM-dd HH:mm:ss,SSSTZD` Bspw.: `2007-09-05 16:40:36,464Z`
DTY_ Korrelations-ID	Zeichenkette	Eindeutige Identifikation eines Aufrufs über Anwendungsgrenzen hinweg. (siehe Abschnitt 4.1.3)	Mit Semikolon separierte Liste von UIDs: Bspw: `c15638a2-4c38-4d18-b887-5ebd2a1c427d; f60143b3-3408-4501-9947-240ec1c48667;`
DTY_Log-Level	Enum	Kategorie eines Logeintrags in Bezug auf Wichtigkeit (siehe Abschnitt 4.1.2).	`FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACE`
DTY_Log-Kategorie	Enum	Kategorie eines Logeintrags in Bezug auf Zweck (siehe Abschnitt 4.1.2).	`FEHLERANZEIGE`, `PROFILING`, `JOURNAL`, `METRIK`, `SICHERHEIT`, `FEHLERANALYSE`
DTY_Ereignisschlüssel	Zeichenkette	Eindeutige Identifikation des Zwecks eines Logeintrags (siehe Abschnitt	Bspw.: `LISYLOO01001`

Datentyp	Basistyp	Bedeutung	Wertebereich
		4.1.2).	
DTY_ Zeichenkette	Zeichenkette	Frei wählbare Zeichenketten.	UTF-8 Zeichenkette
DTY_ Wahrheitswert	Wahrheitswert	Ein Wahrheitswert kann genau zwei Zustände annehmen: „true“ oder „false“.	`true`, `false`
DTY_Ganzzahl	Ganzzahl	Frei wählbare Ganzzahl.	Bspw.: `1`, `2`, `3`, `4`

Tabelle 3: Datentypen

4.1.2 Log-Level, Log-Kategorie und Ereignisschlüssel

Log-Level, Log-Kategorie und Ereignisschlüssel werden in den Nutzungsvorgaben *Logging* (siehe [NutzungsvorgabenLogging]) beschrieben.

4.1.3 Korrelations-ID

Die Korrelations-ID ist immer mitzuloggen, damit die Logeinträge einzelnen Aufrufen zugeordnet und über die Komponenten der Anwendungslandschaft verfolgt werden können. Die Korrelations-ID besteht aus einer Liste von Unique-IDs (siehe [ServicekommunikationKonzept]). Beim Schreiben des Logeintrags werden die einzelnen IDs als Array in den Logeintrag übernommen. Die Korrelations-IDs werden dabei in der Reihenfolge ausgegeben, in der sie erzeugt wurden (die erste ID des Logeintrags ist die, die als erstes erzeugt wurde, die letzte ist die des aktuellen Systemaufrufs).

Eine exemplarische Korrelations-ID ist demnach:

```
{"c15638a2-4c38-4d18-b887-5ebd2a1c427d", "f60143b3-3408-4501-9947-240ec1c48667", "c893d44f-3b8e-446e-a360-06a520440e64"}
```

4.1.4 Nachricht

Die Nachricht enthält eine menschenlesbare Beschreibung des eingetretenen Logereignisses. Diese ist dann relevant, wenn die Logeinträge unverarbeitet durch einen (menschlichen) Anwender ausgewertet werden, was insbesondere bei der Fehleranalyse der Fall ist.

Die Nachrichten sind so weit wie möglich zu vereinheitlichen. Dazu werden in den Nutzungsvorgaben *Logging* (siehe [NutzungsvorgabenLogging]) klare Vorgaben definiert, welche Nachrichten in welcher Situation zu erstellen sind.

4.1.4.1 Datenschutz

Die Nachrichten dürfen keine Daten enthalten, die datenschutzrechtlich relevant oder sicherheitsrelevant sind (d.h. Personendaten, Passwörter, etc.).

Ausnahme: Sollten die datenschutzrechtlich- oder sicherheitsrelevanten Daten zur Analyse zwingend notwendig sein – bspw. wenn fachlich fehlerhafte Daten zu technischen Fehlern führen – dürfen diese Daten im dafür notwendigen Maße mitgeschrieben werden. Die entsprechenden Logeinträge müssen durch das Setzen des Flags ATT_Enthält_fachliche_Daten markiert werden, damit sie durch die Loginfrastruktur gesondert behandelt werden können.

4.2. Logausgabe

In diesem Abschnitt werden sämtliche Aspekte zur Ausgabe der Logeinträge betrachtet und für die Register Factory standardisiert. Dies umfasst die Definition des Formats der einzelnen Logeinträge (Abschnitt 4.2.1), die Ablage der Einträge in Logdateien (Abschnitt 4.2.2).

4.2.1 Logformat

Die Logeinträge werden im JSON-Format ausgegeben. Dies hat den Vorteil, dass die Einträge dadurch sehr einfach maschinell verarbeitet werden können und der Umfang der erzeugten Datenmengen (bspw. im Vergleich zu XML) reduziert wird. Die „Menschenlesbarkeit“ der erstellten Einträge wird durch die Verwendung von JSON leicht eingeschränkt, dies ist aber akzeptiert, da eine Aufbereitung der Logeinträge durch die Log-Infrastruktur stattfindet (siehe Kapitel 5).

Jedes Attribut eines Logeintrags wird in einem entsprechenden JSON Name/Wert-Paar abgelegt. Attributnamen werden dabei komplett in Kleinbuchstaben und ohne Sonderzeichen geschrieben. Im Folgenden wird ein exemplarischer Logeintrag dargestellt, ergänzt um Leerzeichen und Zeilenumbrüche um die Lesbarkeit zu erhöhen:

```
{
  "systemid" : "Systemxyz",
  "hostid" : "appserver01",
  "thread" : "main",
  "logger" : "x.y.HelloWorldZ",
  "zeitstempel" : "2014-03-04T12:27:27.943",
  "korrelationsid" : {"XY1", "XY8", "XY9"},
  "level" : "INFO",
  "kategorie" : "PROFILING",
  "logschluessel" : "LISYLO01001",
  "nachricht" : "Aufruf des Nachbarsystems XYZ in 10 ms
beendet.",
  # Zusätzliche Marker:
  "dauer" : "10"
}
```

Zu beachten ist, dass durch das Logging-Framework sichergestellt wird, dass nur fest definierte Marker in den Logeintrag aufgenommen werden. Die Vergabe „beliebiger“ Marker ist nicht möglich, so dass ein

Namenskonflikt zwischen Marker und den anderen Attributen des Eintrags ausgeschlossen ist.

Ein tatsächlicher Logeintrag (ohne zusätzliche Leerzeichen und Zeilenumbrüche) sieht demnach wie folgt aus:

```
{"systemid":"Systemxyz","hostid":"appserver01","thread":"main
","logger":"x.y.HelloWorldZ","zeitstempel":"2014-03-
04T12:27:27.943","korrelationsid":{"XY1","XY8","XY9"},"level"
:"INFO","kategorie":"PROFILING","logschluessel":"LISYLO01001"
,"nachricht":"Aufruf des Nachbarsystems XYZ in 10 ms
beendet.","dauer" :"10"}
```

4.2.2 Logdateien, Log-Rotation und Komprimierung

Die Vorgaben zu Logdateien sowie deren Rotation und Komprimierung sind in den Nutzungsvorgaben *Logging* [NutzungsvorgabenLogging] beschrieben.

4.3. Logging-Framework

Als Logging-Framework wird in der Register Factory *logback*[1] eingesetzt. Durch die Querschnittsbibliothek *isy-logging* wird eine Fassade bereitgestellt, die eine einheitliche Nutzung von *logback* sicherstellt. Als Grundsätzliche Vorgabe gilt, dass logback durch eine Anwendung niemals direkt, sondern ausschließlich über isy-logging aufgerufen wird (Ausnahme sind hierbei externe Bibliotheken – siehe Abschnitt 4.3.4).

Die Nutzung und Konfiguration von isy-logging sind in den Nutzungsvorgaben *Logging* [NutzungsvorgabenLogging] beschrieben.

Im Folgenden werden die grundlegenden Aspekte der Implementierung von `isy-logging` beschrieben.

4.3.1 Logging-Fassade

`isy-logging` stellt eine Fassade zum Zugriff auf logback bzw. auf dessen Schnittstellen, die durch das allgemeine Framework SLF4J definiert werden, bereit. Wichtig ist, dass durch isy-logging selbst SLF4J nicht implementiert wird. Dies wird aus zwei Gründen nicht gemacht:

- Durch die Bereitstellung einer proprietären Schnittstelle können die spezifischen Anforderungen an das Logging besser umgesetzt werden (Verwendung spezifischer Attribute an den Schnittstellen).

- Es soll nur eine einzige SLF4J-Implementierung eingesetzt werden. Die Verwendung von zwei verschiedenen SLF4J-Implementierungen in einer Anwendung ist zwar möglich, führt jedoch zu einer komplexeren und fehleranfälligeren Konfiguration. Zudem ist durch die gewählte Architektur der Austausch von logback mit einer anderen SLF4J Implementierung auch weiterhin sehr einfach möglich. Es könnte sogar auf eine komplett neue

[1] http://logback.qos.ch/

Logging-Schnittstelle (hinter der Fassade) gewechselt werden, ohne dass der Anwendungscode angepasst werden muss.

In Abbildung 2 ist eine Übersicht der Implementierung der Fassade dargestellt.

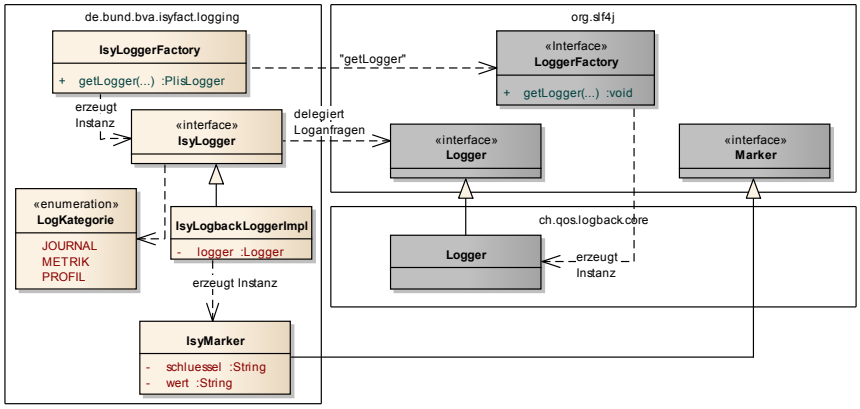

Abbildung 2: Logging-Fassade

Die Klassen der Fassade sind in folgender Tabelle beschrieben:

Klasse	Beschreibung
IsyLoggerFactory	Factory zum Erstellen einer Instanz der Klasse `IsyLoggerImpl`. Sie nutzt das Interface `LoggerFactory`, um eine Instanz der Klasse `Logger` von *logback* zu erzeugen, die durch `IsyLoggerImpl` gewrapped wird.
IsyLogbackLoggerImpl	Logger-Fassade, um Logeinträge zu erstellen. Die Klasse besitzt eine Referenz auf eine Instanz des eigentlichen Loggers (Klasse `Logger`), der durch *logback* bereitgestellt wird. Die bereitgestellten Methoden werden durch das Interface `IsyLogger` definiert, welches in Abschnitt 4.3.1.1 beschrieben ist.
IsyMarker	Implementierung des Interface `Marker`, welches durch SLF4J definiert wird. Die Klasse wird durch den `IsyLogger` verwendet, um einzelne Attribute (Name/Wert-Paare) an den Logger zu übergeben.

4.3.1.1 IsyLogger

Das Interface `IsyLogger` stellt Methoden zum Erstellen von Logeinträgen bereit. Es ist in Abbildung 3 dargestellt.

Abbildung 3: IsyLogger

Das Interface wird in den Nutzungsvorgaben [NutzungsvorgabenLogging] detailliert beschrieben.

Implementiert wird das Interface durch die Klasse `IsyLoggerImpl`. Eingehende Aufrufe werden dabei an den SLF4J-Logger delegiert. Die zusätzlich definierten Parameter (Object…) werden in Form von Markern (jeder Marker beschreibt dabei ein Name/Wert-Paar) an den Logger übergeben. Grundsätzlich ermöglicht es SLF4J jeweils nur einen Marker zu übergeben. Marker können jedoch aneinander gehängt werden (Methode `add` am Marker), so dass ein Marker über diesen Weg weitere Marker enthalten kann.

4.3.2 Log-Aufbereitung

Zur Aufbereitung der Logeinträge im JSON-Layout wird die Klasse `IsyJsonLayout` bereitgestellt. Diese erweitert die Klasse `JsonLayout` von *logback* um die Auswertung der `IsyMarker`. Wird ein entsprechender Marker übergeben, so schreibt die Klasse diesen und alle enthaltenen Marker als Name/Wert-Paare (JSON-Attribute) in das Log.

In Abbildung 4 ist das Zusammenspiel der verschiedenen Klassen zur Aufbereitung der Logeinträge dargestellt.

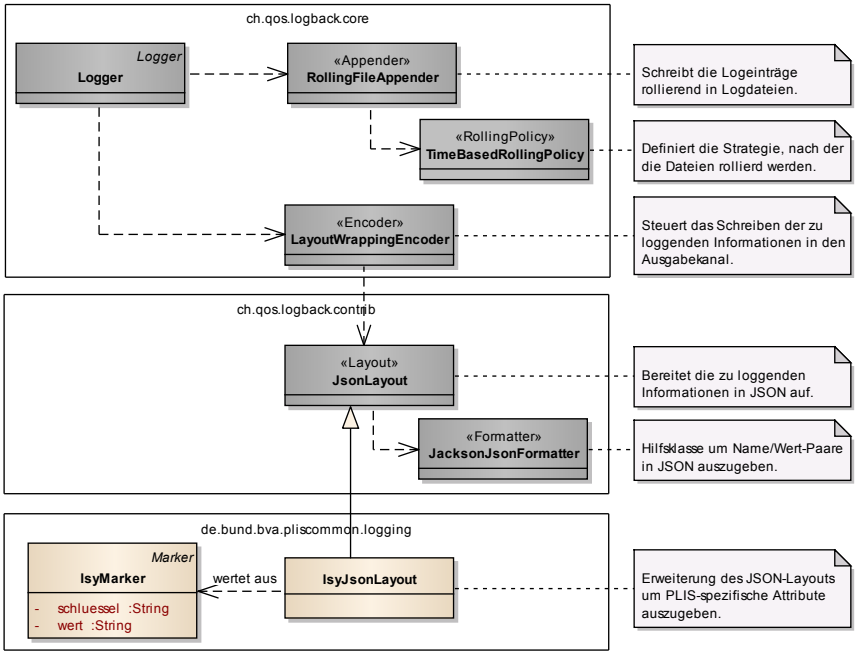

Abbildung 4: Übersicht der Aufbereitung von Logeinträgen

4.3.3 Hilfsklassen

`isy-logging` stellt zwei Hilfsklassen zur Erstellung von Logeinträgen bereit, um die Umsetzung der in den Nutzungsvorgaben [NutzungsvorgabenLogging] definierten Szenarien an die Logerstellung zu vereinfachen. Diese sind in Abbildung 5 dargestellt.

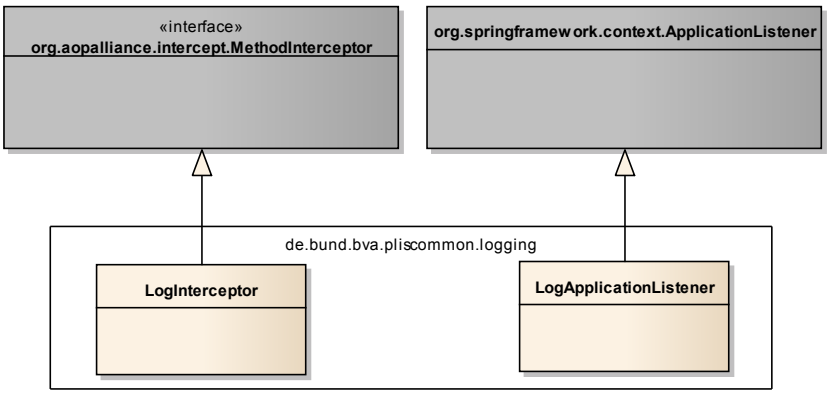

Abbildung 5: Hilfsklassen

Die Klasse `LogInterceptor` dient dabei als Hilfsklasse, um verschiedene Informationen zu Methodenaufrufen zu loggen und wird per Spring-AOP konfiguriert.

Die Klasse `LogApplicationListener` dient dem Loggen von wichtigen Systemereignissen und wird als Spring-Bean konfiguriert.

Die Informationen zur Konfiguration und Verwendung der beiden Klassen sind in den Nutzungsvorgaben [NutzungsvorgabenLogging] beschrieben und werden an dieser Stelle nicht wiederholt.

4.3.4 Logging externer Bibliotheken

Logback wird als einzige Implementierung von SLF4J eingesetzt (`IsyLoggerFactory` implementiert das Interface `LoggerFactory` von SLF4J nicht). Externe Bibliotheken, die SLF4J oder logback nutzen und in die Anwendung eingebunden werden, nutzen dadurch direkt die `LoggerFactory` und damit auch den `Logger`, der durch *logback* bereitgestellt wird. Dies ist ohne Einschränkungen möglich, da die Vereinheitlichung der Logeinträge durch das `IsyJsonLayout` durchgeführt wird, welches auch in diesem Fall zum Tragen kommt. Für Frameworks, die weder logback noch SLF4J nutzen, werden durch SLF4J entsprechende „Bridging Modules“ bereitgestellt, durch die die Aufrufe des Logging-Frameworks auf SLF4J (bzw. logback) umgeleitet werden können. Diese sind zu verwenden, so dass die Erstellung der Logeinträge ausschließlich durch logback erfolgt. Die Nutzung der Bridges ist in den Nutzungsvorgaben *Logging* beschrieben (siehe [NutzungsvorgabenLogging]).

5. Aufbau der Log-Infrastruktur

Die Log-Infrastruktur beschreibt die IT-Systeme und Mechanismen, die zur Verteilung, Verarbeitung, Auswertung und Archivierung der erstellten Logdateien eingesetzt werden und deren Zusammenspiel untereinander. In Abbildung 6 ist eine Übersicht der logischen Komponenten der Log-Infrastruktur dargestellt. Dabei kommen ausschließlich freie IT-Systeme zum Einsatz, die auf dem ELK-Stack (Elasticsearch, Logstash und Kibana[2]) basieren.

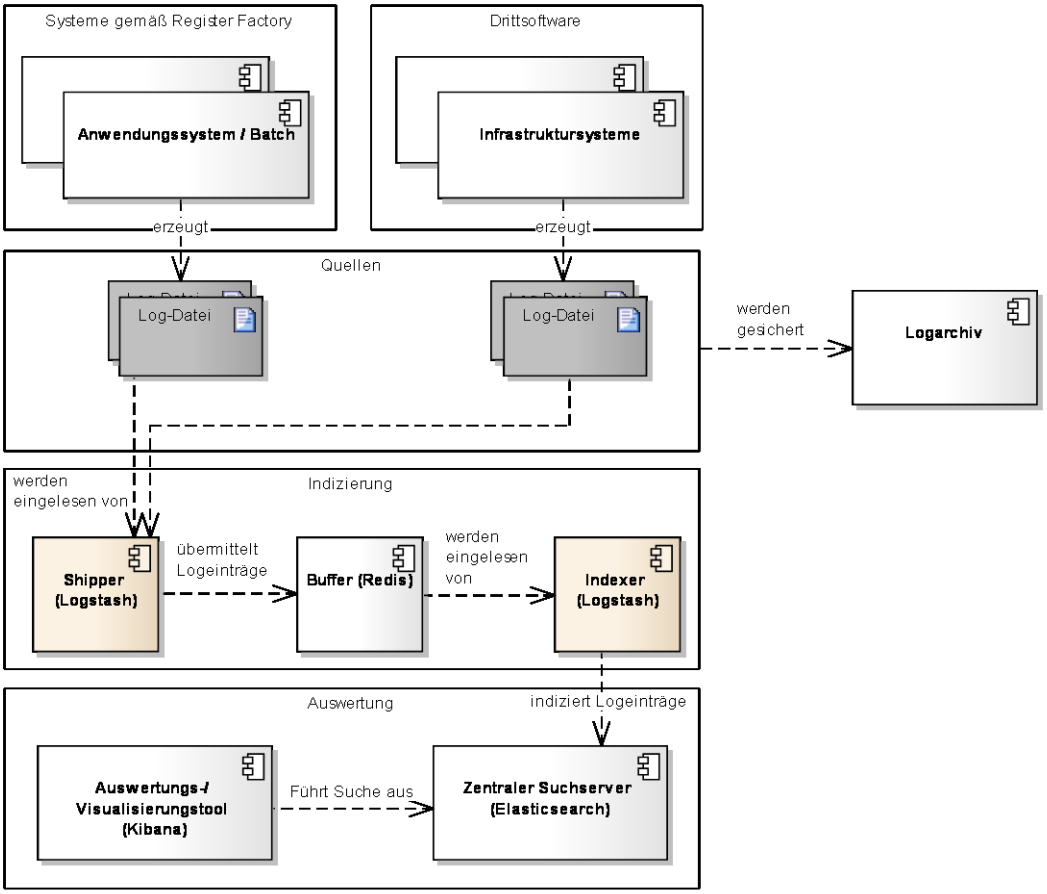

Abbildung 6: Logische Komponenten der Log-Infrastruktur

Die Kernaspekte der Loginfrastruktur sind:

- **Quellen**: Die Anwendungen schreiben ihre Logeinträge in Logdateien. Die Dateien können dabei als Schnittstelle von der Anwendung zur Log-Infrastruktur gesehen werden – jeder Logeintrag, der in eine Datei geschrieben wurde, wird von da an durch die Infrastruktur verwaltet.

  Anmerkung: Das Loggen in Dateien hat gegenüber dem direkten Loggen über eine Netzwerkschnittstelle den Vorteil, dass es robuster gegen Ausfälle der Netzwerkverbindung bzw. des

[2] http://www.elasticsearch.org/overview/

empfangenden Systems ist. Zudem erlaubt es eine sehr einfache Sicherung (Archivierung) der Logs auf Dateiebene.

- **Indizierung**: Eine Kernanforderung an die Infrastruktur ist es, die verschiedenen Log-Quellen einheitlich, zuverlässig und „nahe Echtzeit“ im zentralen Suchserver zu indizieren. Zu diesem Zweck wird eine zuverlässige Transport- und Konvertierungsinfrastruktur aufgebaut. Sie besteht aus den folgenden Komponenten:

    - **Shipper**: Aufgabe des Shippers ist es, die verschiedenen Log-Quellen an die Infrastruktur anzubinden. Hierzu werden Logstash-Instanzen eingesetzt, die es sehr einfach ermöglichen, verschiedene Quellen und Ziele miteinander zu verbinden. Logstash wird so konfiguriert, dass es die Einträge der Log-Quellen kontinuierlich einliest und an den nachgelagerten Puffer übergibt.

    - **Puffer**: Im Puffer werden alle zu indizierenden Einträge zwischengespeichert, bevor sie sequentiell durch den nachgelagerten Indexer weiterverarbeitet werden. Die Verwendung des Puffers ermöglicht es, Lastspitzen abzufangen. Als Puffer kommt Redis[3] zum Einsatz.

    - **Indexer**: Aufgabe des Indexers ist es, die Logeinträge einheitlich im Suchserver zu indizieren. Dazu werden die Logeinträge sequentiell aus dem Puffer eingelesen und im Suchserver (Elasticsearch) indiziert. Hierzu kommt ebenfalls eine Logstash-Instanz zum Einsatz. Logstash bietet auch hierfür die Möglichkeit, die beiden Systeme (Redis und Elasticsearch) deklarativ miteinander zu verbinden.

- **Suchserver und Analysetool:** Zur Analyse der Logeinträge wird ein zentraler Suchserver verwendet, auf dem mit Hilfe eines Analysetools Auswertungen durchgeführt werden können:

    - **Suchserver**: Als Suchserver kommt Elasticsearch zum Einsatz – ein sehr verbreiteter, hoch skalierbarer und etablierter Suchserver auf Basis von Apache Lucene.

    - **Auswertungstool**: Zur Auswertung der in Elasticsearch gespeicherten Logeinträge wird Kibana verwendet. Kibana stellt eine grafische Benutzeroberfläche zum einfachen Formulieren auch komplexer Auswertungen auf dem Datenbestand zur Verfügung.

Die Archivierung der Logs erfolgt auf Ebene der Log-Quellen (Dateien). Durch das Schreiben zeitlich rollierender Logdateien durch die Anwendungen (vergleiche Abschnitt 4.2.2) ist es möglich, die Logs von Anwendungen in festen Zeitintervallen – bspw. stündlich oder täglich – auf einen Archivserver zu verschieben. Die rollierten Logs werden dabei

[3] http://redis.io

bereits durch die Anwendungen komprimiert. Logdateien von Batches können nach Abschluss des jeweiligen Batchlaufes verschoben werden, müssen bei Bedarf aber noch durch die Umgebung komprimiert werden. Das Verschieben ist Aufgabe der Betriebsumgebung.

Im Folgenden wird die Konfiguration der einzelnen Komponenten beschrieben.

5.1. Shipper (Logstash)

Eine Logstash-Konfigurationsdatei besteht aus drei Bereichen:

- **Input**: Zur Definition der einzulesenden Log-Quellen
- **Filter**: Zur Definition von Filtern, die auf die Logeinträge angewendet werden
- **Output**: Zur Definition der Ausgabekanäle der Logdateien

```
input {
  ...
}

filter {
  ...
}

output {
  ...
}
```

5.1.1 Log-Quellen (Input)

Als Log-Quellen werden ausschließlich Dateien verwendet, die in gleicher Weise eingelesen werden. Die Konfiguration der Quellen erfolgt immer nach folgendem Schema:

```
# Zum Einlesen von einzeiligen Logs
file {
  path => "<Pfad>"
  codec => "plain" {
    charset => "UTF-8"
  }
  type => "<Typ>"
}

# Zum Einlesen von Logs im JSON-Format
file {
  path => "<Pfad>"
  codec => json {
    charset => "UTF-8"
  }
  type => "<Typ>"
}
```

Die Platzhalter bedeuten dabei:

- **Pfad**: Absoluter Pfad der einzulesenden Datei.
- **Typ**: Der angegebene Typ wird als JSON-Attribut an alle Logeinträge dieser Quelle angehängt. Er wird primär dazu verwendet, um nachfolgende Filterungen zu steuern.

Die Konfigurationen zum Einlesen der Logdateien der verschiedenen Systemtypen sind in Abschnitt 7.2.1 dargestellt.

5.1.2 Filter

Im Shipper werden keine Filter definiert (dies erfolgt im Indexer), um das Einlesen und Übertragen der Logeinträge in den Puffer möglichst effizient umzusetzen.

5.1.3 Log-Ziele (Output)

Es wird ein einzelner Ausgabekanal definiert, um die Logeinträge zur weiteren Verarbeitung an den Puffer (Redis) zu leiten.

```
output {
  # Weiterleiten der Protokolleinträge in den Puffer
  redis {
    host => "<host>"
    port => "<port>"
    data_type => "list"
    key => "log"
  }
}
```

Die Platzhalter bedeuten dabei:

- **host**: IP-Adresse des Servers, unter der die Redis-Instanz erreichbar ist.
- **port**: Port, unter dem die Redis-Instanz erreichbar ist. (Standard: 6379)

5.2. Puffer (Redis)

Für den Einsatz von Redis als Puffer sind keine speziellen Konfigurationen notwendig. Der Grund, warum Redis als Puffer eingesetzt wird (und nicht Beispielsweise RabbitMQ), liegt in der hohen Performanz und Einfachheit der Lösung.

5.3. Indexer (Logstash)

Als Indexer wird ebenfalls *logstash* eingesetzt, dessen grundlegende Funktionsweise bereits in Abschnitt 5.1 beschrieben wurde und an dieser Stelle nicht wiederholt wird.

Die Konfiguration des Indexers umfasst folgende Elemente:

- Einen Eingabekanal zum Lesen der Einträge aus Redis
- Mehrere Filter zum Parsen und Vereinheitlichen der Logeinträge
- Zwei Ausgabekanäle zum getrennten Indizieren der Einträge in Elasticsearch. Dabei werden Logeinträge, die Fachdaten enthalten, in einen separaten Index geschrieben, der nur durch berechtigte Personen durchsucht werden kann.

```
input {
  redis {
    host => "<host_redis>"
    port => "<port_redis>"
    data_type => "list"
    key => "log"
  }
}

filter {
  ... siehe Abschnitt 5.3.1 ...
}

output {
  if [fachdaten] == "true" {
    elasticsearch {
      host => "<host_elasticsearch>"
      port => "<port_elasticsearch>"
      index => "log_fachdaten-%{+YYYY.MM.dd}"
    }
  } else {
    elasticsearch {
      host => "<host_elasticsearch>"
      port => "<port_elasticsearch>"
      index => "log-%{+YYYY.MM.dd}"
    }
  }
}
```

Die Platzhalter sind dabei selbsterklärend und analog zu den vorgehenden Abschnitten. Die verwendeten Filter werden im nachfolgenden Abschnitt beschrieben.

5.3.1 Filter

Filter werden zum Parsen und Vereinheitlichen der Logeinträge verwendet. Es kommen dabei die folgenden Filterarten zum Einsatz:

5.3.1.1 grok

Grok-Filter werden verwendet, um Attribute der Logeinträge an Hand regulärer Ausdrücke zu parsen und die enthaltenen Felder in separate Attribute zu schreiben.

```
grok {
    match => ["<Attributname>","<RegExp>"]
}
```

Die Parameter bedeuten dabei:

- `Attributname`: Name des Attributs, dessen Wert geparsed werden soll.

- `RegExp`: Regulärer Ausdruck (im Format von Grok[4]) zum Parsen des Werts.

Folgende grok-Filter werden definiert:

- Es wird ein allgemeiner grok-Filter für alle Logeinträge definiert, der den Dateinamen der Eingabedatei ermittelt (standardmäßig liegt nur der komplette Pfad vor).

```
grok {
  match =>
   ["path","%{GREEDYDATA}/%{GREEDYDATA:dateiname}.log"]
}
```

- Es wird ein weiterer allgemeiner grok-Filter definiert, der die im Dateinamen enthaltenen Informationen (`hostid` und `systemid`) aus dem Dateinamen liest:

```
grok {
  match =>

["dateiname","%{GREEDYDATA:hostid}_%{GREEDYDATA:systemid
}.log"]
}
```

- Für jedes System, welches im Format „plain“ (einfacher Text) logged, wird ein spezifischer grok-Filter definiert, der die enthaltenen Informationen in klar definierte JSON-Attribute aufteilt. Die Konfigurationen für die verschiedenen Systemtypen sind in Abschnitt 7.2.1 definiert.

5.3.1.2 date

Date-Filter werden verwendet, um den Zeitpunkt des Logeintrags zu parsen und in das Logstash-Event als `@timetamp` zu übernehmen.

```
date {
    match => ["<Attributname>","<Format>"]
    timezone => "<Zeitzone>"
}
```

Die Parameter bedeuten dabei:

- `Attributname`: Name des Attributs, in dem der Zeitstempel abgelegt ist.

[4] http://logstash.net/docs/1.4.2/filters/grok

- `Format`: Format des zu parsenden Zeitstempels (in Joda-Time[5]).
- `Zeitzone`: Zeitzone des Zeitstempels (in Joda-Time[6]).

Folgende date-Filter werden definiert:

- Für jede Systemart muss ein spezifischer date-Filter definiert werden, der den Zeitpunkt des Logeintrags in das Logstash-Event übernimmt. Diese sind ebenfalls in Abschnitt 7.2.1 definiert.

5.3.1.3 mutate

mutate-Filter können dazu verwendet werden, Attribute zu setzen oder zu manipulieren. Bei allen Log-Quellen, die nicht Register Factory-konform sind, wird ein mutate-Filter verwendet, um das Attribut `@timestamp` (siehe Abschnitt 5.3.1.2) einheitlich in das Feld `zeitstempel` zu schreiben:

```
mutate {
    add_field => [ "zeitstempel", "%{@timestamp}" ]
}
```

Dies ist sinnvoll, da der Zeitstempel dadurch unter einem einheitlichen Namen im ISO-Format abgelegt ist, und das Attribut `@timestamp` eher ein internes Attribut zur Steuerung von logstash ist, welches theoretisch später durch logstash überschrieben werden könnte.

[5] http://joda-time.sourceforge.net/apidocs/org/joda/time/format/DateTimeFormat.html
[6] http://joda-time.sourceforge.net/timezones.html

6. Log-Auswertung

In diesem Kapitel werden die Informationen beschrieben, die letztendlich im Auswertungstool zur Verfügung stehen, und damit die Grundlage zur Durchführung von Analysen darstellen. In den einzelnen Unterabschnitten werden die verschiedenen Logdateien aufgeführt, die an die Log-Infrastruktur anzubinden sind. Wichtig ist, dass die Logeinträge der Systeme der Log-Infrastruktur selbst, auch in die Auswertung einfließen müssen, um Probleme bei der Verarbeitung der Logs zu erkennen.

Konkrete Szenarien zur Auswertung der Logeinträge sind in den Nutzungsvorgaben *Logging* [NutzungsvorgabenLogging] beschrieben.

Die Auswertung der Logeinträge mit Hilfe des Auswertungstools sind in den Nutzungsvorgaben des Logservers [NutzungsvorgabenLogserver] beschrieben.

6.1. Allgemeine Hinweise

In diesem Abschnitt werden allgemeine Hinweise und Besonderheiten zur Auswertung der Logs aufgeführt.

6.1.1 Kein Log-Level FATAL in den Logs

Wie in den Nutzungsvorgaben *Logging* [NutzungsvorgabenLogging] beschrieben, besitzen SLF4J und logback kein Log-Level FATAL. Log-Einträge im Level FATAL und ERROR erscheinen in den Logs beide im Level ERROR, können aber an Hand der Kategorie (FATAL oder ERROR) unterschieden werden:

- Log-Level ERROR: {"level":"ERROR", "kategorie":"ERROR", …}
- Log-Level FATAL: {"level":"ERROR", "kategorie":"FATAL", …}

6.2. Allgemeine Inhalte jedes Logeintrags

Die folgenden Attribute werden automatisch durch logstash gesetzt und sind in jedem Event enthalten:

Attribut	Beschreibung	Format / Beispiel
path	Absoluter Pfad der Datei, aus welcher der Logeintrag gelesen wurde.	`/var/log/xyz-ga/webserver01_xyz-ga_2014-05-10_1700.log`

6.3. Register Factory-konforme Anwendungen

Die Attribute, die durch Anwendungen geloggt werden, die konform zu `isy-logging` umgesetzt sind, werden durch die Entität ETY_Logeintrag in Abschnitt 4.1.1 beschrieben. Diese werden ergänzt um optionale Marker (Name/Wert-Paare). Beim Einsatz von isy-logging werden die folgenden Marker erstellt:

Attribut	Beschreibung	Format / Beispiel
parameter<1..n>	Parameter, die zur Ersetzung der Platzhalter in der Lognachricht übergeben wurden	`Freitext`
methode	Vollständige Signatur einer aufgerufenen Methode. Dieser Marker wird beim Loggen eines Methodenaufrufs mit den Hilfsklassen `LoggingMethodInvoker` und `LoggingMethod-Interceptor` automatisch gesetzt.	`public de.bund.bva. isyfact.logging. hilfsklassen. TestZielParameterPerson de.bund.bva.isyfact. logging.hilfsklassen. TestZielKlasse.setzeName Exception(de.bund.bva. Isyfact.logging. hilfsklassen.TestZiel ParameterPerson, java.lang.String) throws java.lang.Throwable`
dauer	Dauer eines Methodenaufrufs. Dieser Marker wird beim Loggen der Dauer eines Methodenaufrufs mit den Hilfsklassen `LoggingMethodInvoker` und `LoggingMethod-Interceptor` automatisch gesetzt.	`124`

6.4. Register Factory-konforme Anwendungen (vor Logging-Konzept Version 3.0)

Das Logging-Konzept wurde mit der Version 3 grundlegend überarbeitet. Die Logeinträge von Systemen, die konform zum Logging-Konzept vor Version 3 umgesetzt sind, stellen neben der Lognachricht die folgenden Informationen (analog zu Abschnitt 4.1.1.) bereit:

- `systemid`, `hostid`, `thread`, `logger`, `zeitstempel`, `korrelationsid`, `level`

6.5. Tomcat access-log

Die Logeinträge der Tomcat access-logs, die konform zur Register Factory konfiguriert sind, stellen neben der Lognachricht die folgenden Informationen bereit:

Bestandteil	Beschreibung	Format / Beispiel
`Tomcathost`	Remote Hostname, bzw. IP-Adresse falls der Hostname nicht verfügbar ist	Bspw.: `appserver01`
`Tomcatthread`	Der Thread-Name über den der Request verarbeitet wird	Bspw.: `main`
`Benutzername`	Der Remote Benutzername	
`zeitstempelroh`	Datum und Uhrzeit	02/Feb/2012:00:02:50Z
`zeitstempel`	Datum und Uhrzeit im ISO-8601 Format	2012-02-02T00:02:50.000Z
`request`	Erste Zeile des Request. Hieraus ist ersichtlich ob es eine GET/POST ist, welche URI aufgerufen wurde und welches Protokoll verwendet wurde	
`statuscode`	HTTP-Status Code des Response	Bspw.: `200`
`anzahlbytes`	Anzahl der Bytes	Ganzzahl
`thread`	Name des Threads, durch den der Logeintrag erstellt wurde.	Bspw.: `main`
`verarbeitungszeit`	Verarbeitungszeit in Millisekunden	Ganzzahl
`uniqueid`	Die vom Apache gernierte Correlation-ID. Diese wird von	24 Zeichen. Bspw.: `U08ZosCoAAM-AAC9CATgFAAAA`

	mod_jk im Request mitgeliefert.	
`Apachename`	Im Request von mod_jk wird der Name des Apaches geliefert, über den die Anfrage verarbeitet wurde.	Bspw.: `webserver01`

6.6. Wrapper-Log

Die Logeinträge des Wrapper-Logs, die konform zur Register Factory konfiguriert sind, stellen neben der Lognachricht die folgenden Informationen bereit:

Bestandteil	Beschreibung	Format / Beispiel
`zeitstempelroh`	Datum und Uhrzeit im Original des Logeintrags	2014/10/14 11:40:27.630
`zeitstempel`	Datum und Uhrzeit im ISO-8601 Format	2012-02-02T00:02:50.000Z
`prefix`	Präfix des Logeintrags.	Bspw.: `jvm 1`
`level`	Log-Level analog zu analog zu Abschnitt 4.1.1	Bspw.: `INFO`

6.7. Apache access-log und error-log

Die in den Logs der Apache HTTP-Server bereitgestellten Informationen werden im Konzept HTTP-Server [NutzungskonzeptHTTPServer] Apache beschrieben.

Da der HTTP-Server ein anderes Zeitstempel-Format verwendet, werden folgende Felder durch die Loginfrastruktur geändert bzw. ergänzt:

Bestandteil	Beschreibung	Format / Beispiel
`zeitstempelroh`	Datum und Uhrzeit, des ursprünglichen Felds „zeitstempel“ des Logeintrags.	12/Feb/2012:00:02:50 +0000
`zeitstempel`	Datum und Uhrzeit im ISO-8601 Format	2012-02-02T00:02:50.000Z

Wichtig: In den Apache-Logs werden die aufgerufenen URLs inkl. Requestparameter gelogged. Sollten die Requestparameter fachliche Daten enthalten (bspw. weil Suchanfragen über einen REST-Webservice realisiert wurden), so sind alle Apache-Logs des jeweiligen Webservers als „Fachdaten“ zu kennzeichnen (siehe Abschnitt 7.2.1.5).

6.8. Mailserver-log

Die Logs des Mailservers stellen neben der Lognachricht, die folgenden Informationen bereit:

Bestandteil	Beschreibung	Format / Beispiel
`zeitstempelroh`	Datum und Uhrzeit im Original des Logeintrags	Dec 12 10:03:28
`mailserver`	Identifikation des Mailservers	mailserver01

6.9. Suchverfahren-Log

Die Logs des Alphanumerischen Suchverfahrens der Register Factory stellen zum einen allgemeine Informationen bereit, die in jedem Logeintrag enthalten sind. Darüber hinaus werden je nach Logger (asv.search.request, asv.search.result, asv.update.request, asv.update.result, asv.compare.request, asv.compare.result) weitere Informationen bereitgestellt. Zudem wird zwischen zwei Verschiedenen Logformaten (shortData, longData) unterschieden, die ebenfalls die Auswahl der ausgegebenen Attribute beeinflusst. Sämtliche Attribute werden im Folgenden beschrieben. Zu beachten ist, dass die Logger-spezifischen Attribute in einer JSON-Map „`details`“ gekapselt sind.

Bestandteil	Beschreibung	Format / Beispiel
Allgemeine Attribute (bei jedem Eintrag)		
`zeitstempelroh`	Datum und Uhrzeit im Original des Logeintrags	2014-05-12/10:29:01.071
`zeitstempel`	Datum und Uhrzeit im ISO-8601 Format	2012-02-02T00:02:50.000Z
`level`	Log-Level analog zu analog zu Abschnitt 4.1.1	Bspw.: `INFO`
`logger`	Verwendeter Logger. Dieser gibt an, welche Operation (search, update compare) gelogged wurde und ob es sich um eine Anfrage (request) oder Ergebnis (result) handelt.	asv.search.request | asv.search.result | asv.update.request | asv.update.result | asv.compare.request | asv.compare.result
asv.search.request		

korrelationsid	Korrelations-Id analog zu Abschnitt 4.1.1	c15638a2-4c38-4d18-b887-5ebd2a1c427d; f60143b3-3408-4501-9947-240ec1c48667;
_logCategory	Logformat, in dem gelogged wurde.	shortData | longData
searchProfileId	Id des Suchprofils.	1
ATT_Register	Durchsuchtes Register. (nur bei Kurzform)	“xyz”
ATT_Teildatenbestand	Teildatenbestand der Suche. (nur bei Kurzform)	12
_request	Vollständiger Original-Request. (nur bei Langform)	
asv.search.result		
korrelationsid	Korrelations-Id analog zu Abschnitt 4.1.1	c15638a2-4c38-4d18-b887-5ebd2a1c427d; f60143b3-3408-4501-9947-240ec1c48667;
_logCategory	Logformat, in dem gelogged wurde.	shortData | longData
duration	Dauer der ausgeführten Anfrage.	142
count	Anzahl der Treffer.	8
records	Anzahl der zurückgegebenen Records.	5
_result	Bei Kurzform: Ids und Bewertung der zurückgelieferten Records. Bei Langform: Vollständiges Ergebnisobjekt.	{"_recordid":"P0-3","ATT_Bewertung":47}, ":[{"ATT_Namensart":"0", ... } ...]
message	Fehlermeldung.	invalid or missing searchProfileId

asv.update.request		
korrelationsid	Korrelations-Id analog zu Abschnitt 4.1.1	c15638a2-4c38-4d18-b887-5ebd2a1c427d; f60143b3-3408-4501-9947-240ec1c48667;
_logCategory	Logformat, in dem gelogged wurde.	shortData | longData
_ingestMode	Ingest-Mode.	speed|async, update|delete|add
_recordid	Id des Records.	P2
_revision	Revision des Records.	12346
ATT_Register	Durchsuchtes Register. (nur bei Kurzform)	“xyz”
ATT_Nummer	Registernummer. (nur bei Kurzform)	1234
ATT_Teildatenbestand	Teildatenbestand der Suche. (nur bei Kurzform)	12
_request	Vollständiger Original-Request. (nur bei Langform)	
asv.update.result		
korrelationsid	Korrelations-Id analog zu Abschnitt 4.1.1	c15638a2-4c38-4d18-b887-5ebd2a1c427d; f60143b3-3408-4501-9947-240ec1c48667;
_ingestMode	Ingest-Mode.	speed|async, update|delete|add
_modificationMode	Modification-Mode.	speed|async, update|delete|add
duration	Dauer der ausgeführten Anfrage.	142
message	Fehlermeldung.	invalid or missing recordid
asv.compare.request		
korrelationsid	Korrelations-Id analog zu Abschnitt 4.1.1	c15638a2-4c38-4d18-b887-5ebd2a1c427d; f60143b3-3408-4501-9947-240ec1c48667;
_logCategory	Logformat, in	shortData | longData

	dem gelogged wurde.	
ATT_Register	Register der Anfrage. (nur bei Kurzform)	“xyz”
records	Anzahl der Records in Trefferliste.	5
_request	Vollständiger Original-Request. (nur bei Langform)	
asv.compare.result		
`korrelationsid`	Korrelations-Id analog zu Abschnitt 4.1.1	`c15638a2-4c38-4d18-b887-5ebd2a1c427d; f60143b3-3408-4501-9947-240ec1c48667;`
_logCategory	Logformat, in dem gelogged wurde.	shortData | longData
searchProfileId	Id des Suchprofils.	1
ATT_Register	Durchsuchtes Register. (nur bei Kurzform)	“xyz”
ATT_Teildatenbestand	Teildatenbestand der Suche. (nur bei Kurzform)	12
_request	Vollständiger Original-Request. (nur bei Langform)	
asv.search.result		
`korrelationsid`	Korrelations-Id analog zu Abschnitt 4.1.1	`c15638a2-4c38-4d18-b887-5ebd2a1c427d; f60143b3-3408-4501-9947-240ec1c48667;`
_logCategory	Logformat, in dem gelogged wurde.	shortData | longData
duration	Dauer der ausgeführten Anfrage.	142
records	Anzahl der Records in Trefferliste.	5
_result	Bei Kurzform:	{"_recordid":"P0-

	Ids und Bewertung der zurückgelieferten Records. Bei Langform: Vollständiges Ergebnisobjekt.	3","ATT_Bewertung":47}, ":[{"ATT_Namensart":"0", ... } ...]
message	Fehlermeldung.	`Records missing in request`

6.10. logstash-log

Logstash stellt neben der Lognachricht, die folgenden Informationen bereit:

Bestandteil	Beschreibung	Format / Beispiel
`zeitstempelroh`	Datum und Uhrzeit im Original des Logeintrags	2014-12-11T12:18:19.580000Z
`zeitstempel`	Datum und Uhrzeit im ISO-8601 Format	2012-02-02T00:02:50.000Z
`level`	Log-Level analog zu analog zu Abschnitt 4.1.1	Bspw.: `INFO`

7. Anhang

7.1. Glossar

Begriff	Erläuterung	Synonym(e) oder Übersetzung(en)
Log	Speicherort (Datenbank, Datei, Log-Server) der aufgezeichneten Ereignisse.	
Log-Aufbereitung	Mechanismus innerhalb des Logging-Frameworks zur Umwandlung des Logeintrags in ein bestimmtes Format (bspw. JSON).	Log-Layout
Log-Auswertung	Analyse und Bewertung der in den Logeinträgen enthaltenen Informationen.	
Log-Infrastruktur	IT-Systeme und Mechanismen, die zur Verteilung, Verarbeitung, Auswertung und Archivierung der erstellten Logdateien eingesetzt werden und deren Zusammenspiel untereinander	
Log-Inhalte	Definition der Informationen, die in die Logs geschrieben werden.	
Log-Kategorie	Klassifizierung eines Logeintrags im Hinblick auf dessen Zweck (FEHLERANZEIGE, PROFILING, JOURNAL etc.).	
Log-Level	Klassifizierung eines Logeintrags im Hinblick auf dessen Wichtigkeit (FATAL, ERROR, WARN, INFO, DEBUG, TRACE).	
Log-Quelle	Knoten (System oder Datei) in der Log-Infrastruktur, aus dem Log-Einträge gelesen werden.	
Log-Rotation	Zeit- oder größenbasiertes „rotieren“ von Logdateien. „Rotieren“ bedeutet hierbei, dass die vorhandene Datei verschoben bzw. umbenannt (i.d.R. mit einem Index oder Zeitstempel versehen) und eine neue Logdatei begonnen wird.	
Ereignisschlüssel	Eindeutige Identifikation des Ereignisses und dem damit verbundenen Zweck, auf Grund	

Begriff	Erläuterung	Synonym(e) oder Übersetzung(en)
	dessen ein Logeintrag im Log-Level INFO erstellt wurde.	
Log-Ziel	Knoten (System oder Datei) in der Log-Infrastruktur, in den Log-Einträge geschrieben werden.	
Logausgabe	Sammelbegriff für alle Mechanismen und Vorgaben der Ausgabe (Format, Medium etc.) von Logeinträgen in einer Anwendung.	
Logdatei	Datei, in die die aufgezeichneten Ereignisse geschrieben werden.	
Logeintrag	Entität des Logs. Beim Aufzeichnen eines Ereignisses wird ein einzelner abgeschlossener Logeintrag geschrieben.	
Logerstellung	Sammelbegriff für alle Mechanismen und Vorgaben der Erstellung von Logeinträgen in einer Anwendung (Log-Inhalte, Logausgabe und Logging-Framework).	
Logformat	Format/Struktur in der die Logeinträge abgelegt werden.	
Logging	Aufzeichnung von Ereignissen eines Systems zur Analyse von System-, Nutzerverhalten oder Systemzuständen.	
Logging-Framework	Framework, welches zum Schreiben der Logeinträge innerhalb der Anwendungen verwendet wird (bspw. logback, log4j).	
Lognachricht	Menschenlesbare Nachricht, welches ein geloggtes Ereignis näher beschreibt. Fester Bestandteil jedes Logeintrags.	
Protokollierung	Mitschreiben von Informationen zu durchgeführten Geschäftsvorfällen auf Grund fachlicher Anforderungen. Eine detaillierte Abgrenzung zum Logging befindet sich in Kapitel 2.	

Begriff	Erläuterung	Synonym(e) oder Übersetzung(en)
Monitoring	Betriebliche Überwachung der Anwendungslandschaft zur frühzeitigen Eskalation von Problemen. Eine detaillierte Abgrenzung zum Logging befindet sich in Kapitel 2.	Überwachung

7.2. Konfigurationsvorlagen

Dieser Abschnitt enthält Konfigurationsvorlagen zur Einrichtung der Log-Infrastruktur.

7.2.1 Logstash (Shipper und Indexer)

7.2.1.1 Register Factory-konforme Anwendungen

Input im Shipper:

```
# Einlesen IsyFact-konformer Logdateien
file {
  path => "<Pfad zu Logdatei>"
  codec => json {
    charset => "UTF-8"
  }

  type => "isy"
}
```

Filter im Indexer:

```
# IsyFact-konforme Anwendungen
if [type] == "isy" {
  date {
    match => [ "zeitstempel", "yyyy-MM-dd'T'HH:mm:ss.SSS" ]
    timezone => "UTC"
  }
}
```

7.2.1.2 Register Factory-konforme Anwendungen (vor Logging-Konzept Version 3.0)

Input im Shipper:

```
# Einlesen IsyFact-konformer Logdateien (vor Version 3.0)
file {
  path => "<Pfad zu Logdatei>"
  codec => plain {
    charset => "UTF-8"
  }
  type => "isy"
}
```

Filter im Indexer:

```
# Register Factory-konforme Anwendungen vor Version 3.0
if [type] == "isy" {
  grok {
    match => ["message","\[D: %{GREEDYDATA:zeitstempel}\]
\[P: %{GREEDYDATA:level}\] \[K:
%{GREEDYDATA:korrelationsid}\] \[T: %{GREEDYDATA:thread}\]
\[L: %{GREEDYDATA:logger}\] - \[M: %{GREEDYDATA:nachricht}\]
"]
  }
  date {
    match => [ "zeitstempel", "yyyy-MM-dd'T'HH:mm:ss.SSS" ]
    timezone => "UTC"
  }
}
```

7.2.1.3 Tomcat access

Input im Shipper:

```
# Einlesen Tomcat-Access-Logs
file {
  path => "<Pfad zu Logdatei>"
  codec => plain {
    charset => "UTF-8"
  }
  type => "tomcat_access"
}
```

Filter im Indexer:

```
# Tomcat-Access-Log
if [type] == "tomcat_access" {
  grok {
    match => ["message","%{GREEDYDATA:tomcathost}
%{GREEDYDATA:tomcatthread} %{GREEDYDATA:benutzername}
\[%{GREEDYDATA:zeitstempelroh}\] '%{GREEDYDATA:request}'
%{GREEDYDATA:statuscode} %{GREEDYDATA:anzahlbytes}
%{GREEDYDATA:thread} %{GREEDYDATA:verarbeitungszeit}
%{GREEDYDATA:uniqueid} %{GREEDYDATA:tomcatname}"]
  }
  date {
    match => [ "zeitstempelroh", "dd/MMM/yyyy:HH:mm:ss Z" ]
    timezone => "UTC"
  }

  mutate {
    add_field => [ "zeitstempel", "%{@timestamp}" ]
  }
}
```

7.2.1.4 Wrapper-Log

Input im Shipper:

```
# Einlesen Tomcat-Wrapper-Logs
file {
  path => "<Pfad zu Logdatei>"
  codec => plain {
    charset => "UTF-8"
  }
  type => "wrapper"
}
```

Filter im Indexer:

```
# Wrapper-Log
if [type] == "wrapper" {
  grok {
    match => ["message","%{GREEDYDATA:level} \|
%{GREEDYDATA:prefix} \| %{GREEDYDATA:zeitstempelroh} \|
%{GREEDYDATA:nachricht}"]
  }
  date {
    match => ["zeitstempelroh", "dd MMM yyyy HH:mm:ss.SSS"]
      timezone => "UTC"
  }
# Leerzeichen entfernen
  mutate {
    strip => [ "prefix" ]
    add_field => [ "zeitstempel", "%{@timestamp}" ]
  }
}
```

7.2.1.5 Apache access-Log und error-Log

Input im Shipper:

```
# Einlesen Apache-Access-Logs
file {
  path => "<Pfad zu Logdatei>"
  codec => json {
    charset => "UTF-8"
  }
  type => "apacheaccess"
}

# Einlesen Apache-Error-Logs
file {
  path => "<Pfad zu Logdatei>"
  codec => plain {
    charset => "UTF-8"
  }
  type => "apacheerror"
}
```

Filter im Indexer:

```
# Apache-Logs
if [type] == "apache_access" or [type] == "apache_error" {
  mutate {
    add_field => [ "zeitstempelroh", "%{zeitstempel}" ]
  }
  date {
    match => ["zeitstempelroh", "dd/MMM/yyyy:HH:mm:ss Z"]
    timezone => "UTC"
  }
  mutate {
    add_field => [ "zeitstempel", "%{@timestamp}" ]
  }
  # (Optional) Apache-Logs müssen als „Fachdaten“
gekennzeichnet werden, falls in den Requestparametern der URL
fachliche Daten enthalten sein können. Dies ist insbesondere
bei REST-Services der Fall, die Suchen anbieten.
  mutate {
    add_field => [ "fachdaten", "true" ]
  }
}
```

7.2.1.6 Mailserver-Logs

Input im Shipper:

```
# Einlesen der Mailserver-Logs
file {
  path => "<Pfad zu Logdatei>"
  codec => plain {
    charset => "UTF-8"
  }  type => "mailserver"
}

# Einlesen der Mailserver-Logs (Error)
file {
  path => "<Pfad zu Logdatei>"
  codec => plain {
    charset => "UTF-8"
  }  type => "mailerror"
}

# Einlesen der Mailserver-Logs (Error)
file {
  path => "<Pfad zu Logdatei>"
  codec => plain {
    charset => "UTF-8"
  }  type => "smtpprotokoll"
}
```

Filter im Indexer:

```
# Mailserver-Logs
if [type] == "mailserver" or [type] == "mailerror" or [type]
== "smtpprotokoll" {
  grok {
    match =>
["message","%{GREEDYDATA}%{SYSLOGTIMESTAMP:zeitstempelroh}
%{DATA:mailserver} %{GREEDYDATA:nachricht}"]
  }
  mutate {
    strip => ["mailserver"]
  }

  # Locale muss gesetzt werden, da die Monatsangaben auf
  # Englisch sind (Dec vs. Dez)
  date {
    match => ["zeitstempelroh", "MMM dd HH:mm:ss", "MMM  d
HH:mm:ss"]
    timezone => "UTC"
    locale => "en"
  }
  mutate {
    add_field => [ "zeitstempel", "%{@timestamp}" ]
  }
}
```

7.2.1.7 Suchverfahren-Log

Das Alphanumerische Suchverfahren erstellt mehrere Logdateien (asv-searches.log, asv-updates.log, asv-compares.log), die alle auf die gleiche Weise verarbeitet werden können.

Input im Shipper:

```
# Einlesen der ASV-Logs
file {
  path => "<Pfad zu Logdatei>"
    codec => "plain"
    type => "asv"
```

```
  }
}
```

Filter im Indexer:

```
# asv-Log
if [type] == "asv" {
  grok {
    match => ["message", '%{DATA:zeitstempelroh} %{DATA}
%{DATA:level} %{GREEDYDATA} - %{GREEDYDATA:nachrichtroh}']
  }
  json {
    source => "nachrichtroh"
    target => "details"
  }
  date {
   match => [ "zeitstempelroh", "yyyy-MM-dd/HH:mm:ss.SSS" ]
   timezone => "UTC"
  }
  mutate {
    add_field => [ "zeitstempel", "%{@timestamp}" ]
    replace => ["korrelationsid", "_correlationId" ]
  }
}
```

7.2.1.8 logstash-Log

Input im Shipper:

```
# Einlesen der logstash-Logs
file {
  path => "<Pfad zu Logdatei>"
  codec => plain {
    charset => "UTF-8"
  }
  type => "logstash"
}
```

Filter im Indexer:

```
# logstash-Log
if [type] == "logstash" {
  grok {
    match => ["message",
'{:timestamp=>"(?<timestamp>.*)",(.*)
:message=>"(?<nachricht>.*)",(.*) :level=>:(?<level>.*)}']
  }
  mutate {
    replace => ["zeitstempelroh", "%{timestamp}" ]
    remove_field => msg
  }
  date {
    match => [ "zeitstempelroh", "ISO8601" ]
    remove_field => 'timestamp'
  }
  mutate {
    add_field => [ "zeitstempel", "%{@timestamp}" ]
  }
}
```

8. Quellenverzeichnis

[NutzungskonzeptHTTPServer]

Nutzungskonzept HTTP Server
30_Plattform/Nutzungskonzept_Apache_HTTP_Server.pdf.

[NutzungsvorgabenLogging]

Nutzungsvorgaben Logging
20_Bausteine\Logging\Nutzungsvorgaben_Logging.pdf.

[NutzungsvorgabenLogserver]

Nutzungsvorgaben Logserver
Link wird ergänzt, wenn Dokument fertiggestellt wurde.

[ProtokollierungKonzept]

Konzept Protokollierung
20_Bausteine\Protokollierung_Protokollrecherche\Konzept_Protokollierung.pdf .

[ServicekommunikationKonzept]

Grundlagen der Servicekommunikation innerhalb der Plattform
10_Blaupausen\Integrationsplattform\Grundlagen_interne_Servicekommunikation.pdf .

[ÜberwachungKonfigKonzept]

Konzept Überwachung und Konfiguration
20_Bausteine\Ueberwachung_Konfiguration\Konzept_Ueberwachung-Konfiguration.pdf .

9. Abbildungsverzeichnis

Abbildung 1: ETY_Logeintrag und ETY_Marker 12
Abbildung 2: Logging-Fassade 18
Abbildung 3: IsyLogger 19
Abbildung 4: Übersicht der Aufbereitung von Logeinträgen 20
Abbildung 5: Hilfsklassen 20
Abbildung 6: Logische Komponenten der Log-Infrastruktur 22