DALE-UV

DGUV

Java-Prüfmodul

Benutzerhandbuch

Version 1.12

Reg-Nr. 201.1.2

Stand: 16.12.2022

werum
SOFTWARE & SYSTEMS

Java-Prüfmodul
Benutzerhandbuch
Version 1.12
Reg-Nr. 201.1.2
Stand: 16.12.2022

Änderungen vorbehalten.

Werum Software & Systems AG
Wulf-Werum-Straße 3
21337 Lüneburg
Tel. +49 (0) 4131/8307-0
Fax +49 (0) 4131/8307-200
info@werum.de
www.werum.de

Dokumenthistorie

Version	Stand	Erläuterungen
1.0	17.07.2006	Ersterstellung
1.1	17.08.2006	Beschreibung der Java-Bibliotheksschnittstelle erweitert
1.2	11.10.2006	Beschreibung der Java-Bibliotheksschnittstelle in das Dokument dale_uv_spezifikation.doc verschoben
1.3	23.10.2006	Tag pl_version eingefügt
1.4	06.12.2006	Benötigte Bibliotheken ergänzt
1.5	05.06.2007	Beschreibung der Aufrufparameter angepasst
1.6	30.07.2007	Beschreibung der Aufrufparameter ergänzt
1.7	22.04.2008	Prüfschlüssel eingefügt
1.8	30.05.2008	Fehler in der Abbildung 2 korrigiert
1.9	28.08.2008	Beschreibung der Java-Schnittstelle angepasst.
1.10	01.10.2012	Abrufen von Versionsinformationen dokumentiert.
1.11	20.09.2013	Unterstützung für Java 1.4 eingestellt
1.12	16.12.2021	Log4J entfernt

Inhaltsverzeichnis

1 Einleitung

Das Java-Prüfmodul wurde im Rahmen des Projekts DALE-UV (Datenaustausch mit Leistungserbringern in der gesetzlichen Unfallversicherung) entwickelt. Das Programm übernimmt die Prüfung einer Datenlieferung im vorgeschriebenen XML-Format und erstellt ein Prüfprotokoll ebenfalls im XML-Format. Aus Kompatibilitätsgründen zur Vorgängerversion kann über die Aufrufparameter das Prüfprotokoll auch in textueller Form ausgegeben werden. Zum Starten des Programms werden einige Konfigurationseinstellungen benötigt, die über ein dafür vorgesehenes Dialogprogramm definiert und in einer INI-Datei gespeichert werden können. Darüber hinaus bietet das Prüfmodul die Möglichkeit die Prüfungen in ein anderes Java-Programm einzubinden. Die entsprechende Schnittstelle sieht nicht nur die Gesamtprüfung einer XML-Datei, sondern auch die Prüfung von Einzelfeldern vor.

2 Konfiguration

Für die Konfiguration des Prüfmoduls starten Sie das Programm **daleuv_conf.exe**.

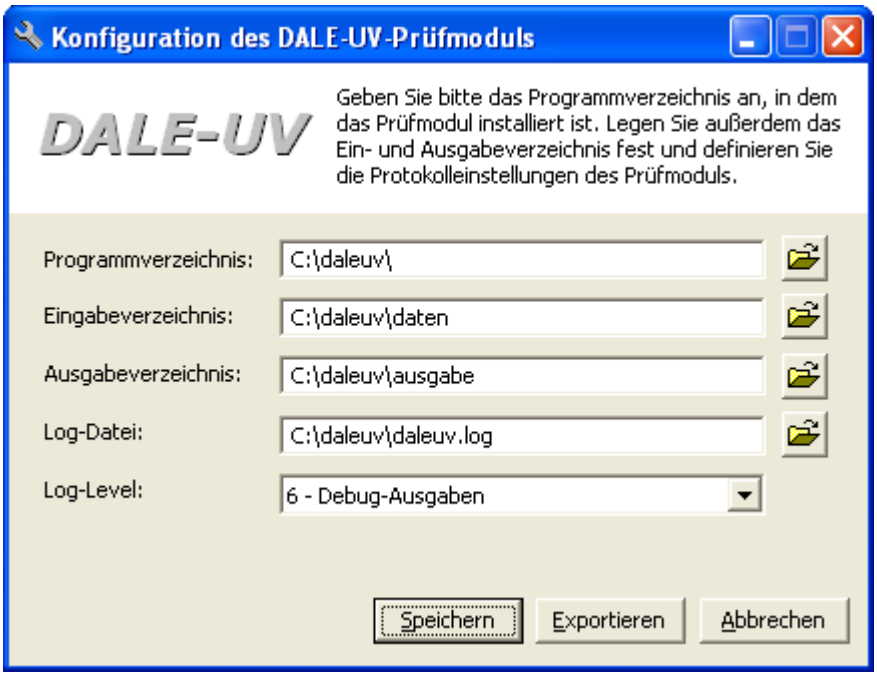

Abbildung1: Dialog für die Konfiguration des Prüfmoduls

Geben Sie im Eingabefeld **Programmverzeichnis** das Installationsverzeichnis des Prüfmoduls ein. Wählen Sie außerdem das **Eingabeverzeichnis** aus, in dem die zu prüfenden XML-Dateien zu finden sind, und legen Sie das **Ausgabeverzeichnis** fest, in das die Prüfergebnisse geschrieben werden. Optional kann durch die Angabe eines **Log-Levels** die Protokollierung des Prüfablaufs für Testzwecke konfiguriert werden. Der Parameter **Log-Datei** wird durch das Java-Prüfmodul nicht mehr unterstützt, damit erfolgt diese Protokollierung immer über die Standardausgabe.

Um die vorgenommenen Einstellungen in der Windows-Registry zu speichern, klicken Sie auf **Speichern**. Daraufhin werden unter dem Schlüssel HKEY_CURRENT_USER\Software\DALE-UV\Pruefmodul die Daten gespeichert. Beim wiederholten Aufruf des Konfigurationstools werden die Felder mit den gespeicherten Einträgen vorbelegt.

Um die vorgenommenen Einstellungen im INI-Format zu exportieren, klicken Sie auf **Exportieren**. Daraufhin wird die Datei daleuv.ini ins Programmverzeichnis generiert.

Achtung: Das Prüfprogramm lädt die Konfigurationseinstellungen ausschließlich aus der INI-Datei. Die Einträge in der Windows-Registry werden nur dafür verwendet, um die Konfigurationseinstellungen zwischenzuspeichern.

3 Prüfung

3.1 Aufrufparameter

Zum Prüfen einer XML-Datei verwenden Sie das Programm **SVC_PR_XMLPRUE.bat**. Es handelt sich hierbei um eine Konsolenanwendung mit folgenden Aufrufparametern:

```
SVC_PR_XMLPRUE.bat <XML-Datei> [-ini <INI-Verzeichnis>]
```

Die angegebene Datei aus dem konfigurierten Eingabeverzeichnis wird geprüft und die Prüfergebnisse werden in die Datei <XML-Datei>_error.xml im Ausgabeverzeichnis geschrieben z.B.:

```
SVC_PR_XMLPRUE.bat DABE.xml
```

Eingabedatei: c:\daleuv\daten\DABE.xml

Ausgabedatei: c:\daleuv\ausgabe\DABE.xml_erorr.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fehlerlog>
   <datenquelle>E:\daleuv\daten\DABE.xml</datenquelle>
   <pl_version>DABE_07_1_01</pl_version>
   <pruefschluessel>1</pruefschluessel>
   <fehler>
      <id>abs[0].abs_5#abs_5</id>
      <feldname>abs_5</feldname>
      <feldbeschriftung>Länderkennzeichen des Absenders</feldbeschriftung>
      <referenzstruktur>Merkmalpruefung</referenzstruktur>
      <gewicht>9</gewicht>
      <stufe>2</stufe>
      <code>ALL27</code>
      <text_kurz>Fehler im Feld Länderkennzeichen des Absenders (<abs_5>):
                 Ungültiges Länderkennzeichen.</text_kurz>
   </fehler>
   <fehler>
      <id>#ALL26_beh_13</id>
      <feldname>beh_13</feldname>
      <feldbeschriftung>IK des D-Arztes</feldbeschriftung>
      <referenzstruktur>beh</referenzstruktur>
      <gewicht>1</gewicht>
      <stufe>2</stufe>
      <code>ALL26</code>
      <text_kurz>Fehler im Feld IK des D-Arztes (<beh_10>): Die IK-Nummer muss eine
                 gültige IK-Nummer sein.</text_kurz>
   </fehler>
</fehlerlog>
```

Abbildung 2: Beispiel einer XML-Ausgabedatei mit Fehler

Im Element <pruefschluessel> wird bei einer fehlerfreien Eingabedatei ein 30-stelliger Schlüssel ausgegeben. Dieser Schlüssel muss anschließend in das Element <swh_8> eingetragen werden. Die Eingabedatei darf ansonsten nicht mehr angepasst werden. Wurden bei der Prüfung Fehler festgestellt, steht im Element <pruefschluessel> eine 1. Die aufgeführten Fehler müssen daraufhin korrigiert werden und die Eingabedatei muss erneut geprüft werden. Beachten Sie bitte, dass ein Bericht ohne einen Prüfschlüssel oder mit einem nicht korrekten Prüfschlüssel vom Server abgelehnt wird.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fehlerlog>
   <datenquelle>E:\daleuv\daten\DABE.xml</datenquelle>
   <pl_version>DABE_08_1_01</pl_version>
   <pruefschluessel>08101080828J3aANu9DkCg9K7FeUhp</pruefschluessel>
</fehlerlog>
```

Abbildung 3: Beispiel einer XML-Ausgabedatei ohne Fehler

Mit der zusätzlichen Angabe -ini <INI-Verzeichnis> kann das Verzeichnis vorgegeben werden, in dem nach der Datei **daleuv.ini** gesucht werden soll z.B.:

```
SVC_PR_XMLPRUE.bat DABE.xml –ini c:\daleuv
```

Wenn das INI-Verzeichnis nicht angeben ist, sucht das Programm nach der Datei **daleuv.ini** zunächst im aktuellen Arbeitsverzeichnis und dann im Programmverzeichnis, in dem das Prüfmodul installiert wurde.

Das Programm liefert je nach Ergebnis unterschiedliche Rückgabewerte:

0	Die Prüfung war erfolgreich, es wurden keine Fehler festgestellt.
1	Die Prüfung war erfolgreich, es wurde mindestens ein Fehler der Stufe 1 festgestellt (Anwendungsfehler)
2	Die Prüfung war erfolgreich, es wurde mindestens ein Fehler der Stufe 2 festgestellt (Datenfehler)
3	Die Prüfung war erfolgreich, es wurde mindestens ein Fehler der Stufe 3 festgestellt (Warnung)
-1	Die Prüfung wurde wegen eines Fehlers abgebrochen.

Die Anwendung kann auch im Kompatibilitätsmodus gestartet werden, in dem die Ausgabedatei in textueller Form mit der Dateiendung _SVC_PR_XMLPRUE.txt generiert wird. Zusätzlich wird eine Datei erzeugt, die nur den Rückgabewert beinhaltet. Diese Datei muss absolut bzw. relativ zum Arbeitsverzeichnis angegeben werden.

Optional kann ein Verzeichnis vorgegeben werden, in dem nach der Datei **daleuv.ini** gesucht werden soll (siehe oben).

```
SVC_PR_XMLPRUE.bat <Rückgabewert-Datei> <XML-Datei> [<INI-Verzeichnis>]
```

Beispiel eines Aufrufs im Kompatibilitätsmodus:

```
SVC_PR_XMLPRUE.bat c:\return.txt DABE.xml
```

Eingabedatei: c:\daleuv\daten\DABE.xml

Ausgabedatei: c:\daleuv\ausgabe\DABE.xml_SVC_PR_XMLPRUE.txt

Rückgabewert-Datei: c:\return.txt

Beispiel eines Aufrufs im Kompatibilitätsmodus mit der Angabe des INI-Verzeichnisses:

```
SVC_PR_XMLPRUE.bat c:\return.txt DABE.xml c:\daleuv
```

Falls die Batch-Datei `SVC_PR_XMLPRUE.bat` nicht im aktuellen Verzeichnis liegt, muss beim Aufruf der absolute Pfad zu dieser Datei angegeben werden.

Beispiel einer Text-Ausgabedatei:

```
Strukturpruefungen der Datei: <E:\daleuv\daten\DABE.xml>
am 07.17.06 um 16:32:37 Uhr
Prueftabellenversion: DABE:01:2:UV
################################################################################
################################################################################

Ungültiges Länderkennzeichen.
Fehler-Nr.: <ALL27>, Fehlerstufe: <2>
Fehler im 5.XML-Tag in <abs>.
################################################################################
Die IK-Nummer muss eine gültige IK-Nummer sein.
Fehler-Nr.: <ALL26>, Fehlerstufe: <2>
Fehler im 13.XML-Tag in <beh>.
################################################################################
Bei der Prüfung der Datei wurde mindestens ein Fehler mit der Fehlerstufe <2>
gefunden.
Pruefschluessel: 1
```

Abbildung 4: Beispiel einer Text-Ausgabedatei

3.2 Versionsinformationen

Die Versionsinformationen zur aktuellen Lieferung des DALE-UV-Prüfmoduls sind ebenfalls über das Programm **SVC_PR_XMLPRUE.bat** abrufbar:

```
SVC_PR_XMLPRUE.bat -versionInfo
```

Das Programm liefert folgende Ausgabe:

```
Version;Nachrichtentyp;Lieferdatum;Generierungsdatum
08_1_01;ABRZ;20080914;20080913
07_3_01;ABRZ;20071021;20070917
08_1_01;DABE;20080914;20080913
…
```

Die Ausgabe kann relativ einfach in eine Tabellendarstellung gewandelt werden. Die Ausgabe enthält pro enthaltene Strukturprüfungsversion eines Nachrichtentyps eine Zeile. Jede Zeilen enthält die Versionsnummer, den Nachrichtentypen, das Datum der Lieferung des Prüfmoduls (bzw. der Compilierung der Java-Klassen) und das Datum der Generierung der PL-Spezifikationen (aus dem PL-Editor).

3.3 Weitere Dateien

Auf dem Zielsystem muss Java in der Version 1.6 oder neuer installiert sein. Wenn das Programm java.exe nicht über die Umgebungsvariable PATH zu finden ist, muss in der Batch-Datei SVC_PR_XMLPRUE.bat der vollständige Pfad zu dieser Datei angegeben werden.

Das Prüfmodul benötigt für die Ausführung alle im Unterverzeichnis **lib** befindlichen Jar-Dateien. Diese werden über den folgenden Eintrag der Datei SVC_PR_XMLPRUE.bat eingebunden:

```
java -cp .;./lib/daleuv_java_lzu.jar …
```

Wenn sich das lib-Verzeichnis nicht im Arbeitsverzeichnis des Programms befindet, muss der absolute Pfad zur Jar-Datei daleuv_java_lzu.jar im Aufrufparameter -cp (Classpath) in der Batch-Datei SVC_PR_XMLPRUE.bat angegeben werden.

Beispiel:

```
java -cp .;D:/my_inst/daleuv/lib/daleuv_java_lzu.jar …
```

Alle anderen Jar-Dateien werden im Verzeichnis der Jar-Datei daleuv_java_lzu.jar gesucht.

4 Java-Schnittstelle

In diesem Abschnitt wird die Einbindung der Bibliotheksschnittstelle erläutert. Weiterführende Informationen, wie die API-Dokumentation und Beispiele für den Aufruf der Prüffunktionen, befinden sich im Verzeichnis **doc** der Lieferung. Die Startseite der API-HTML-Dokumentation ist die Datei **index.html**.

4.1 Einbinden der Bibliotheksschnittstelle

Für die Einbindung der Bibliotheksschnittstelle werden die Jar-Dateien plausi.jar, commons-logging.jar, daleuv_java_lzu.jar und pl_generator.jar benötigt. Diese sind im **lib**-Verzeichnis der Lieferung vorhanden.

4.2 Verwendung der Bibliotheksschnittstelle

Voraussetzung für die Verwendung der Bibliotheksschnittstelle ist der Import und die Instanziierung des `PlausiPerformer`:

```
import de.dale_uv.parser.PlausiPerformer;
PlausiPerformer plausiPerformer = new PlausiPerformer();
```

Es gibt 3 verschiedene Möglichkeiten eine Plausibilisierung durchzuführen:

1. die Eingabedaten liegen in einer XML-Datei vor vorhanden und sollen mit der Standardplausiversion geprüft werden; <Pfad zur XML-Datei> gibt an, wo die zu plausibilisierende Datei liegt:

   ```
   String filename = <Pfad zur XML-Datei>;
   FehlerLog daleuvreport = plausiPerformer.doPlausi(filename);
   ```

2. die Eingabedaten sind in einem DOM-Tree vorhanden und sollen mit der Standardplausiversion geprüft werden:

   ```
   DOM domtree = null; //entsprechend initialisieren
   FehlerLog daleuvreport = plausiPerformer.doPlausi(domtree);
   ```

3. es soll nur ein bestimmtes Feld geprüft werden:

   ```
   FehlerLog daleuvreport = plausiPerformer.doFeldPlausi(<XPath zum
                       Feld>,<Wert>,<Nachrichtenname>);
   //Beispielaufruf
   FehlerLog daleuvreport = plausiPerformer.doFeldPlausi(
                "/dabe_file/dabe/vin/vin_9","17.06.2006","DABE");
   ```

Die hier aufgeführten Methoden sind in der Schnittstelle jeweils auch als Version mit einem weiteren Parameter (Plausi-Kontext) vorhanden. Diese sind aber nur für den Serverbetrieb vorgesehen und können clientseitig (beim Arzt) nicht verwendet werden.

Das Fehlerlog kann in einer Fehlerdatei ausgegeben werden. Hierfür gibt es zwei Möglichkeiten:

1. neues Fehlerdateiformat (XML):

```
int returncode = plausiPerformer.writeErrorFile(<Pfad zur Fehler-Datei>,
            daleuvreport);
```

2. altes Fehlerdateiformat (TXT und Returncodedatei):

```
int returncode = plausiPerformer.writeErrorFileInOldFormat(
            <Pfad zur Fehler-Datei>, daleuvreport);
//schreiben der Returncodedatei
plausiPerformer.writeReturncodeFile(<Pfad zur Returncode-Datei>,returncode);
```

<Pfad zur Fehler-Datei> gibt an, wohin die Datei mit den Fehlern geschrieben werden soll.

<Pfad zur Returncode-Datei> gibt an, wohin die Datei mit dem Returncode geschrieben werden soll.

Beispiel für das Auslesen eines Fehlerlogs:

```
FehlerLog daleuvreport = plausiPerformer.doPlausi(filename);

// getDatenquelle() beinhaltet den Eingabedateinamen, wenn die Eingabedaten als
// XML-Datei eingelesen wurden oder den Namen der Plausi, wenn die Eingabedaten als
// DOM-Tree oder durch eine Feldplausi eingelesen wurden
String datenquelle = daleuvreport.getDatenquelle();

// getPlausiVersion() Liefert die Version der Plausibilitätsprüfungen.
String plausiVersion = daleuvreport.getPlausiVersion();

// getPlausiDatum() Liefert das Erstelldatum der Plausibilitätsprüfungen.
String plausiDatum = daleuvreport.getPlausiDatum();

// getPruefschluessel() Liefert den neu berechneten Prüfschlüssel.
String pruefschluessel = daleuvreport.getPruefschluessel();

// getFehlerliste() gibt alle aufgetretenen Fehler als Collection zurück
Iterator fehlerIterator = daleuvreport.getFehlerliste().iterator();
while (fehlerIterator.hasNext())
{
      Fehler fehler = (Fehler) fehlerIterator.next();
      //getFehlerID() gibt die FehlerID zurück (z.B. dabe[0].abs[0].abs_2#abs_2)
      String fehlerID = fehler.getFehlerID();

      //getFeldname() gibt den XML-Feldnamen aus (z.B. abs_2)
      String feldname = fehler.getFeldname();

      //getFeldbezeichnung() gibt die Feldnamen im Formular aus (z.B. Absendername)
      String feldbezeichnung = fehler.getFeldbezeichnung();

      //getReferenzstruktur() gibt die Referenzstruktur der Prüfung an,
      // bei Merkmalsprüfungen = Merkmalpruefung und bei anderen Prüfungen
      // der Name des Themenbereichs zu dem die Prüfung gehört
      fehler.getReferenzstruktur();

      // getFehlerGewicht() gibt das Gewicht des Fehlers vom PL-Editor an
      String fehlerGewicht = fehler.getFehlerGewicht();
      // getFehlercode() gibt den Fehlercode zurück (z.B. ALL27)
      String fehlerCode = fehler.getFehlercode();

      // getFehlerstufe() gibt die DALE-UV Fehlerstufe zurück
      int fehlerstufe = fehler.getFehlerstufe();
```

```
        // getFehlertextKurz() gibt den Fehlertext_kurz
        String fehlertextKurz = fehler.getFehlertextKurz();

        // isAusnahmefehler() true, wenn bei der Prüfung ein Ausnamhefehler
        // aufgetreten ist
        if (fehler.isAusnahmefehler())
        {
               //getAusnahmefehlerKlasse() gibt den Exceptiontyp zurück
               String ausnahmefehlerKlasse = fehler.getAusnahmefehlerKlasse();
               // getAusnahmefehlertext() gibt die Message der Exception zurück
               String ausnahmefehlertext= fehler.getAusnahmefehlertext();

               //gibt den Stacktrace aus
               Iterator ausnahmefehlerStacktraceIterator =
                      fehler.getAusnahmefehlerStacktrace().iterator();
               while (ausnahmefehlerStacktraceIterator.hasNext())
               {
                      StackTraceElement element = (StackTraceElement)
                             ausnahmefehlerStacktraceIterator.next();
               }
        }
}
```

Die genaue Beschreibung der Programmierschnittstelle für das Einbinden der JAVA-Prüfungen als Bibliothek liegt als JAVA-Doc im HTML-Format vor.

4.3 Logging

Die Bibliotheksschnittstelle verwendet für das Logging die Apache-Commons-Logging-Schnittstelle. Die einbindende Anwendung kann (z.B. über eine entsprechende Bridge) den zu verwendenden Logging-Mechanismus vorgeben. Ist dies nicht der Fall, werden die Logausgaben in die Standardausgabe geschrieben.