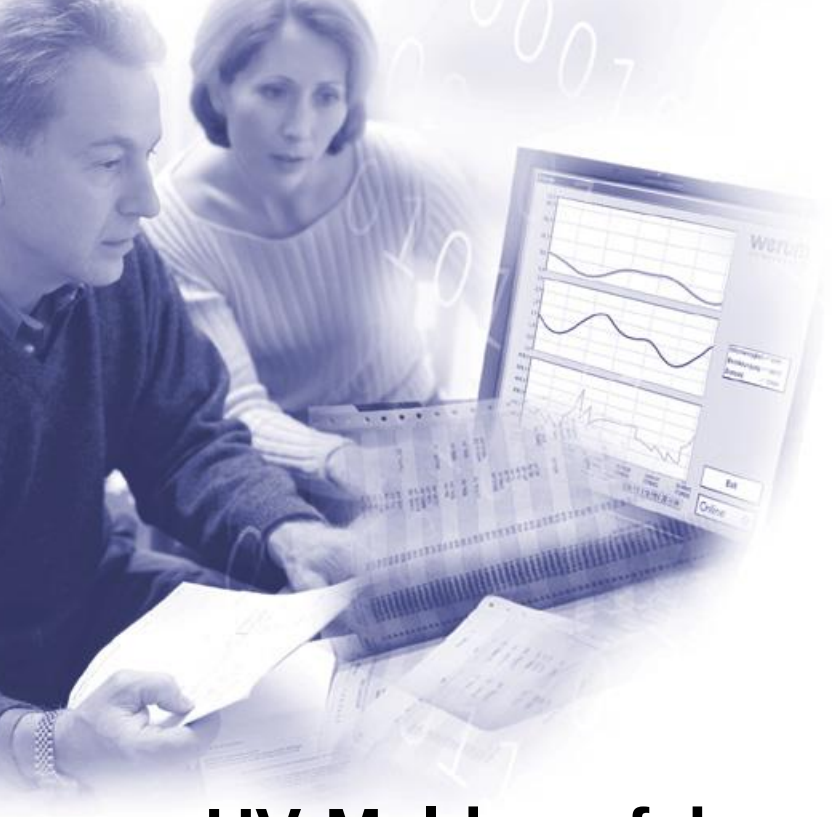

DGUV

UV-Meldeverfahren Kernprüfmodule DSLN, DSAS, DSKO

C++-Prüfmodul

Anwendungshandbuch

Version 1.0

Reg-Nr. keine

Stand: 20.06.2016

werum
SOFTWARE & SYSTEMS

C++-Prüfmodul
Anwendungshandbuch
Version 1.0
Reg-Nr. keine
Stand: 20.06.2016

Änderungen vorbehalten.

Werum Software & Systems AG
Wulf-Werum-Straße 3
21337 Lüneburg
Tel. +49 (0) 4131/8900-0
Fax +49 (0) 4131/8900-20
info@werum.de
www.werum.de

Dokumenthistorie

Version	Stand	Erläuterungen
1.0	22.06.2016	Ersterstellung

Inhaltsverzeichnis

1 Einleitung

Das C++-Kernprüfmodul des UV-Meldeverfahrens führt die Kernprüfungen der Datensätze DSLN, DSAS und DSKO aus. Das C++-Prüfmodul wird von den Anwendern als Bibliotheksmodul eingebunden. Eine Prüfung wird immer mit der entsprechenden Kernprüfklasse auf einem Datensatz aufgerufen. Der Datensatz wird hierbei als Zeichenkette vorgegeben. Das Ergebnis einer Plausibilitätsprüfung wird als C++-Ergebnisobjekt mit einer entsprechenden Fehlerliste zurückgeliefert.

Zusätzlich zum Aufruf der Prüfungen als Bibliotheksfunktion (API) ist für den Test auch der Aufruf der als Programm mit Ein- und Ausgabedatei möglich.

2 Installation und Konfiguration

Alle mit dem Prüfmodul gelieferten C++-Bibliotheken (DLL-Dateien) müssen vor der Verwendung des Prüfmoduls im Arbeitsverzeichnis der Anwendung bereitgestellt werden.

Anschließend können die Plausibilitätsprüfungen auf Basis der API des Prüfmoduls aufgerufen werden.

Das Prüfmodul umfasst folgende C++-Bibliotheken:

uv-kp-mod.dll	Die UV-Meldeverfahren-spezifische Laufzeitumgebung der Kernprüfungen mit der API und den spezifischen Prüfklassen.
uv-kp.dll	Die UV-Meldeverfahren-spezifischen Plausibilitätsprüfungen.
pl_runtime.dll	Die Basis-Laufzeitumgebung der Plausibilitätsprüfungen.
msvcp100.dll **msvcr100.dll**	Vom Prüfmodul verwendete allgemeine Funktionen. Diese Dateien müssen nur berücksichtigt werden, wenn sie nicht bereits im Betriebssystem enthalten sind.
include\UV_KP.h	C++-Header-Datei zum Einbinden des Prüfmoduls in andere C++-Programme
lib\uv-kp-mod.lib	Statische Bibliothek zum Einbinden des Prüfmoduls in andere C++-Programme
doc*	API-Dokumentation des Prüfmoduls

Alle DLL- Dateien müssen im Arbeitsverzeichnis des Prüfprogramms oder alternativ in einem der über die Umgebungsvariable **PATH** definierten Verzeichnis (z.B. **%windir%\system32**) liegen.

3 API des Prüfmoduls

Die API des Prüfmoduls besteht aus wenigen Klassen und Interfaces (zu finden im Namespace "`uvkp`"), die es der Anwendung ermöglichen, die Kernprüfungen aufzurufen. Abbildung 1 gibt einen Überblick der API.

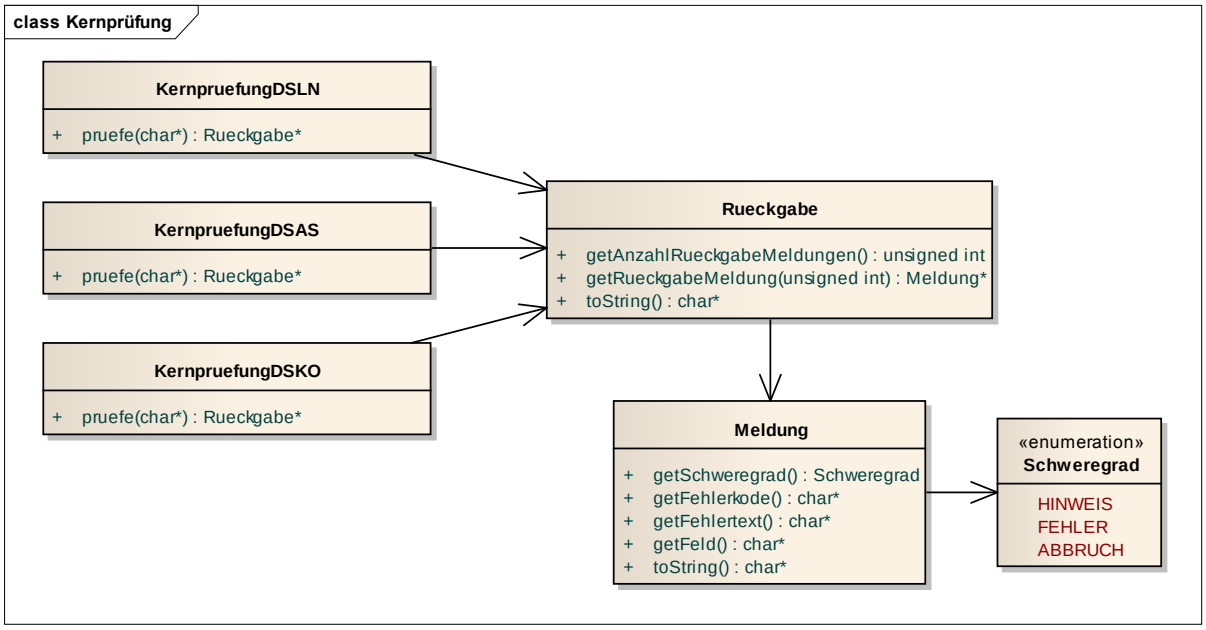

Abbildung 1: Überblick über die API des Kernprüfmoduls

Für jede Satzart stellt das C++-Kernprüfmodul eine Kernprüfungsklasse zur Verfügung (siehe Abbildung 1 und Tabelle 1).

Kernprüfungsklasse	Satzart	Beschreibung
`KernpruefungDSLN`	DSLN	Datensatz eines Lohnnachweises.
`KernpruefungDSAS`	DSAS	Datensatz für die Abfrage von Stammdaten.
`KernpruefungDSKO`	DSKO	Datensatz für die Kommunikation.

Tabelle 1: Kernprüfungsklassen

Die genaue Beschreibung der Programmierschnittstelle für das Einbinden des C++-Prüfmoduls liegt im HTML-Format vor und ist in der Lieferung enthalten.

3.1 Aufruf einer Kernprüfung

Der Aufruf einer Kernprüfung auf einem Datensatz einer Satzart erfolgt durch die Konstruktion eines Objekts der Kernprüfungsklasse dieser Satzart und den Aufruf der Methode `pruefe()`. Beim Aufruf dieser Methode wird der Satz als Zeichenkette (`char*`) vorgegeben. Das Ergebnis der Kernprüfung ist ein Objekt der Schnittstelle `Rueckgabe`. Das Rückgabeobjekt hält den Ergebnisstatus der Kernprüfung und die aufgetretenen Fehler als Array von `Meldung`-Objekten.

***Hinweis**: Da der Datensatz dem Kernprüfmodul als char* vorgegeben wird, ob liegt die Interpretation von Dateien und ihrem Encoding der umgebenden Anwendung.*

Beispiel:

```
#include <iostream>
#include <UV_KP.h>
using namespace std;
using namespace uvkp;

int main(int argc, char* argv[])
{
   // Pruefung ausführen
   KernpruefungDSLN kernpruefung; // DSLN-Prüfung
   Rueckgabe* rueckgabe = kernpruefung.pruefe(argv[1]);

   // Alle Meldungen einzeln ausgeben
   cout << rueckgabe->getAnzahlRueckgabeMeldungen() << " Meldungen:" << endl;
   for (unsigned int i = 0; i < rueckgabe->getAnzahlRueckgabeMeldungen(); i++)
   {
      Meldung* meldung = rueckgabe->getRueckgabeMeldung(i);
      char* meldungString = meldung->toString();
      cout << meldungString << endl;
      delete[] meldungString; // meldungString löschen und den Speicher freigeben
   }

   // Alle Meldungen als DBFE-Bausteine ausgeben
   char* rueckgabeString = rueckgabe->toString();
   cout << rueckgabeString << endl;
   delete[] rueckgabeString; // rueckgabeString löschen und den Speicher freigeben

   delete rueckgabe; // rueckgabe löschen und den Speicher freigeben
}
```

3.2 Ergebnis der Kernprüfung interpretieren

Das Ergebnis der Kernprüfung wird als `Rueckgabe`–Objekt geliefert und muss nach der Auswertung gelöscht werden. Die Methode `getReturnCode()` liefert das Gesamtergebnis der Kernprüfung als `int`-Wert. Tabelle 2 stellt die möglichen Ergebniswerte dar.

Wert	Beschreibung
0	Kernprüfung fehlerfrei (String-Array `getAnzahlRueckgabeMeldungen()` ist 0)
1	Kernprüfung enthält Hinweise
2	Kernprüfung enthält Fehler
3	Kernprüfung enthält Fehler und Hinweise
4	Kernprüfung ist abgebrochen

Tabelle 2: Mögliche Werte des Ergebnisses einer Kernprüfung

Die Methode `getAnzahlRueckgabeMeldungen()` liefert die Anzahl aller der bei der Kernprüfung erkannten Fehler. Über die Methode getRueckgabeMeldung() kann jeder einzelne Fehler ausgelesen und anhand des Schweregrads, der Fehlermeldung, der Fehlernummer und des betroffenen Felds ausgewertet werden.

Die Methode `toString()` liefert eine Zeichenkette mit maximal neun Meldungen der bei der Kernprüfung erkannten Fehler als DBFE-Bausteine (siehe Kernprüfungsspezifikation).

Die möglichen Fehlernummern und Fehlertexte sind der Kernprüfungsspezifikation zu entnehmen.