Statistisches Bundesamt

Softwarebibliotheken zu .CORE für JAVA

Nutzerdokumentation

Version: 14
Stand: 02.01.2023
Status: Freigegeben
Kontakt: eSTATISTIK.core@destatis.de

Inhalt

Änderungsverlauf

Version	Datum	Änderung
1	16.03.04	Neuerstellung
2	06.09.04	Anpassung wegen C/C++-Schnittstelle
3	07.03.05	Anpassung an 1.0
	02.03.06	Anpassung an 1.03
4	03.03.06	Anpassung an 1.1
5	08.01.07	Erweiterungen in Kap. 3 und 8, Kap. 4, 5, 6, 7 neu
	01.03.07	Neuer Fehlerschlüssel 43022 in Kap. 7.4
6	26.03.07	Anpassungen an 1.2
7	20.03.12	Anpassungen an die Version 1.3 der Softwarebibliotheken zu .CORE (Auftrennung der Bibliothek CORE.connect in die Bibliotheken CORE.connect und CORE.inspector)
8	08.05.14	Umstellung auf JAVA 7 und die mögliche Verwendung von TLS 1.2. Optionale Erweiterung der expliziten Setzung der Prüfstufe zur Prüfung einer Datenlieferung. Zusätzliche Zertifikatsprüfung anhand der in JAVA hinterlegten Wurzelzertifikate.
9	16.06.15	Anpassungen an die Version 1.7 der Softwarebibliotheken zu .CORE. Unterstützung der Softwarebibliotheken zu JAVA 8.
10	03.08.15	Anpassungen an die Version 1.7.3 der Softwarebibliotheken zu .CORE.
11	12.06.18	Anpassungen an die Version 1.8.0 der Softwarebibliotheken zu .CORE.
12	13.04.21	Umstellung von CORE.connect auf neue Versionen der verwendeten Bibliotheken (Aktualisierung der verwendeten Dritt-Bibliotheken)
13	23.12.21	Abhängigkeit zu Log4J, sowie log4j.jar-Datei entfernt Protokollierung erfolgt über Apache-Commons-Logging einbindende Software muss die konkrete Implementierung der Protokollierung vorgeben und konfigurieren, ansonsten erfolgt die Protokollierung über Standardausgabe
14	02.01.23	Xerces von 2.12.1 auf 2.12.2 aktualisiert Xalan entfernt (war nur eine Runtime-Dependency und kann mittlerweile durch JRE-Standardfunktionalität ersetzt werden) Korrektur im CSV-Inspector (bei wiederholten Merkmalsgruppen innerhalb von wiederholten Merkmalsgruppen wurde immer nur die erste untergeordnete Merkmalsgruppe geprüft)

1 Einleitung

Die Softwarebibliotheken CORE.connect und CORE.inspector unterstützen Softwarehersteller und Befragte bei der Realisierung von Statistikmodulen für die Erstellung, Prüfung und den Versand von Datenlieferungen im XML-Format DatML/RAW an den gemeinsamen Dateneingang von eSTATISTIK.core der Statistischen Ämter des Bundes und der Länder. Das Lieferdatenformat DatML/RAW ist Teil des XÖV-zertifizierten Nachrichtenformats XStatistik. (s. [SPEZ]).

Ein Statistikmodul ist eine in die Unternehmenssoftware integrierte Softwarekomponente, die die Gewinnung der statistischen Daten aus der DV des Befragten durchführt oder steuert und aus den so gewonnenen Daten die Datenlieferung erstellt und versendet. Auf diese Weise lässt sich das gesamte Datenerhebungsverfahren beim Befragten automatisieren (s. Abb. 1).

Die Softwarebibliothek CORE.connect bietet alle notwendigen Funktionen, um eine Datenlieferung im XML-Format DatML/RAW zusammenzustellen und an den gemeinsamen Dateneingang zu versenden. Die gemeldeten Daten werden am gemeinsamen Dateneingang der amtlichen Statistik einer formalen Prüfung unterzogen und das Prüfergebnis in dem XML-Format DatML/RES (s. [SPEZ]) zurück übermittelt. Ein Verstoß gegen formale Prüfungen kann zur Abweisung von einzelnen Meldungen oder ggf. der gesamten Datenlieferung führen. Mit Hilfe der Softwarebibliothek CORE.inspector können formale Prüfungen vor Versenden einer Datenlieferung durchgeführt werden. Durch die maschinelle Erstellung und Prüfung der Datenlieferung werden die Aufwände bei den Befragten erheblich verringert und gleichzeitig die Qualität der statistischen Rohdaten deutlich verbessert. Die Softwarebibliothek CORE.connect und CORE.inspector werden kostenlos von den statistischen Ämtern zur Verfügung gestellt.

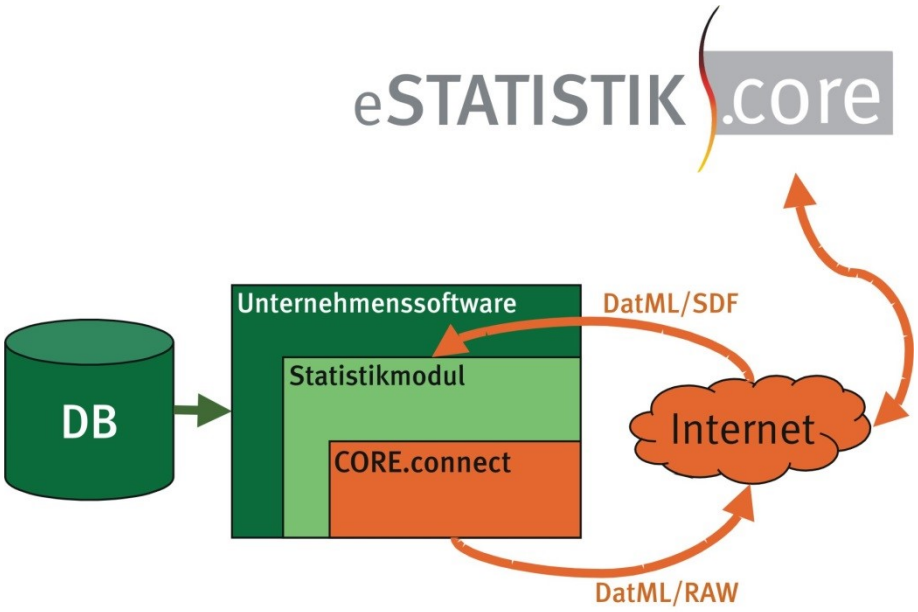

Abbildung 1: Verwendung der Softwarebibliothek CORE.connect

Die Softwarebibliotheken CORE.connect und CORE.inspector richten sich an DV-Experten bei den Befragten und bei Softwareherstellern, die die Erstellung, Prüfung und den Versand von Datenlieferungen in DatML/RAW in ihre Datenverarbeitung bzw. Softwareprodukte integrieren wollen. Für die Nutzung werden dementsprechend einschlägige DV-Kenntnisse vorausgesetzt.

Referenzdokumente

[SPEZ]	Spezifikation von XStatistik (DatML/RAW und DatML/RES): https://www.xrepository.deutschland-online.de/xrepository/index.xhtml → Spezifikation → XStatistik
[SDF]	Spezifikation von DatML/SDF Erhebungsportal https://erhebungsportal.estatistik.de/Erhebungsportal/# → Hilfsmittel und Automatisierung → Unterstützung für Entwickler → Spezifikation zu .CORE → Datenformate → Thema: „Das Format DatML/SDF“
[KOMM]	Kommunikationsschnittstelle des gemeinsamen Dateneingangs von eSTATISTIK.core Erhebungsportal https://erhebungsportal.estatistik.de/Erhebungsportal/# → grauer Bereich „Hilfsmittel und Automatisierung“ → Unterstützung für Entwickler → Spezifikation zu .CORE → CORE - Kommunikationsschnittstelle → . . . für JAVA-Entwickler → Thema „Die Java-Bibliothek CORE.connect“
[BSP]	Beispiel-Implementierungen der Bibliotheken für .CORE Erhebungsportal https://erhebungsportal.estatistik.de/Erhebungsportal/# → grauer Bereich „Hilfsmittel und Automatisierung“ → Unterstützung für Entwickler → Spezifikation zu .CORE → CORE - Kommunikationsschnittstelle → . . . für JAVA-Entwickler → Thema „Java-Projekt mit Beispielimplementierungen“

2 Installation und Deinstallation

2.1 Systemvoraussetzungen

Die Softwarebibliotheken CORE.connect und CORE.inspector sind auf jeder Hardwareplattform einsetzbar, für die eine Java-Laufzeitumgebung (JRE) Version 7 oder Version 8 zur Verfügung steht.

2.2 Installation

Die jeweilige Softwarebibliothek zu .CORE wird für alle Plattformen als ZIP-Archiv ausgeliefert. Für die Installation ist das Archiv CORE.connect und CORE.inspector in ein geeignetes Installationsverzeichnis zu entpacken. Weitere Installationsschritte sind nicht notwendig. Nach der Installation gibt es die folgenden Unterverzeichnisse:

CORE.connect

beispiele.client	enthält Beispieldaten und Beispielprogramme zum Versand einer Datenlieferung, sowie zum Abrufen der Prüfprotokolle
beispiele.generator	enthält Beispieldaten und Beispielprogramme zur Generierung von DatML/RAW-Lieferdatendokumenten
Bin	enthält den CORE.mapper und Beispiele zur Erzeugung von DatML/RAW-Lieferdatendokumenten
Doc	enthält die Javadoc-Dokumentation für die CORE.connect-Bibliothek
Jni	enthält die C/C++-Schnittstelle für die CORE.connect-Bibliothek
Lib	enthält die notwendigen JAR-Bibliotheken

CORE.inspector

beispiele	enthält Beispieldaten und Beispielprogramme zum Überprüfen von DatML/RAW-Lieferdatendokumenten
Doc	enthält die Javadoc-Dokumentation für die CORE.inspector-Bibliothek
Lib	enthält die notwendigen JAR-Bibliotheken (connect.jar und inspector.jar – s. Hinweis)

HINWEIS:

Die Softwarebibliothek CORE.inspector benötigt Ressourcen, die über das Package *de.destatis.core.resource* der CORE.connect-Bibliothek zu Verfügung gestellt werden. Aus diesem Grund ist es **nicht** möglich, die Softwarebibliothek CORE.inspector ohne die CORE.connect-Bibliothek zu verwenden. Die Softwarebibliothek CORE.connect ist ohne die Bibliothek CORE.inspector nutzbar.

2.3 Deinstallation

Die Softwarebibliotheken CORE.connect und CORE.inspector werden durch Löschen des Installationsverzeichnisses deinstalliert.

2.4 Protokollierung

Mit der Version 1.10.0 von CORE.connect wurde die Abhängigkeit zu Log4J grundsätzlich entfernt. Auch die log4j.jar-Datei wurde aus der Lieferung entfernt. Falls die einbindende Software weiterhin die Version 1.2.X von Log4J verwendet, bestehen keine Funktionseinschränkungen und es muss beim Austausch der Bibliothek nichts weiter beachtet werden.

Die Protokollierung innerhalb der Bibliothek erfolgt nun über Apache-Commons-Logging. Somit kann die einbindende Software die konkrete Implementierung der Protokollierung vorgeben und konfigurieren. Wird keine Protokollierung vorgegeben, erfolgt die Protokollierung über die Standardausgabe.

Hinweis: Wenn für die Protokollierung nicht Log4J in der Version 1.2.X verwendet wird, stehen die Funktionen zur programmatischen Vorgabe einer Logdatei und des Detaillierungsgrades über die CORE-Schnittstelle nicht mehr zur Verfügung.

3 Funktionsumfang

Die vorliegende Softwarebibliotheken CORE.connect und CORE.inspector sind Java-Bibliotheken und für die Verwendung in Java-Software konzipiert. Die zugehörigen JAR-Archive befinden sich im Verzeichnis `lib`. Über die in CORE.connect eingebundene JNI-Schnittstelle von Java wird zusätzlich eine C/C++-Schnittstelle angeboten, über die die Bibliothek auch in anderen Programmierumgebungen (C, C++, Visual Basic, Delphi) genutzt werden kann. Alle notwendigen Dateien einschließlich einem Beispiel für die C/C++-Schnittstelle sind im Verzeichnis `jni` zu finden.

Die Softwarebibliotheken zu .CORE bieten ab Version 1.3.0 folgende Funktionalitäten:

1. Erzeugung einer Datenlieferung durch Vorgabe eines DatML/RAW-Dokuments nach Transformation mit einem entsprechenden XSLT-Stylesheet.
2. Testen der Verbindung zum gemeinsamen Dateneingang von .CORE möglich.
3. Abrufen der aktuellen Erhebungsbeschreibung (DatML/SDF) und der aktuellen Liefervereinbarung (pdf-Dokument) zu einer Erhebung. DatML/SDF ist die formale, maschinell auswertbare Form der Liefervereinbarung im XML-Format (s. [SDF]).
4. Generierung einer Datenlieferung im Format DatML/RAW basierend auf den Vorgaben der Erhebungsbeschreibung.
5. Validierung einer erzeugten Datenlieferung bzgl. der Korrektheit der XML-Syntax und des Root-Elements.
6. Überprüfung einer erzeugten Datenlieferung gemäß den Vorgaben in einer Liefervereinbarung. Die Liefervereinbarung beschreibt für jede Statistik, welche Merkmale zu melden sind und in welchen Wertebereichen die Merkmalswerte liegen müssen. Durch die Prüfung gegen die Liefervereinbarung wird sichergestellt, dass die versendete Datenlieferung technisch weiterverarbeitet werden kann. Die Liefervereinbarungen liegen in einer maschinell auswertbaren Form -im XML-Format DatML/SDF- vor (s. [SDF]) und werden von der Softwarebibliothek bei der Überprüfung automatisch ausgewertet (s. Kap. 7).
7. Prüfen einer DatML/RAW-Datenlieferung gemäß den Vorgaben aus der Liefervereinbarung über den gemeinsamen Dateneingang von eSTATISTIK.core.
8. Versand der erzeugten und geprüften Datenlieferung an den gemeinsamen Dateneingang der Statistik Deutschlands. Die Übertragung erfolgt per HTTPS mit automatischer Authentifizierung beim Empfänger mit separat bereitgestellter Kennung und Passwort. Als Antwort auf die Datenlieferung erhält der Absender unmittelbar das Prüfergebnis im XML-Format (DatML/RES) und den Eingangsstempel zurück. Mit diesem Schlüssel (Eingangsstempel) kann der Absender zu einem späteren Zeitpunkt das Prüfprotokoll für die Datenlieferung über den gemeinsamen Dateneingang der amtlichen Statistik abholen.
9. Anforderung eines Prüfprotokolls zu einer Datenlieferung vom gemeinsamen Dateneingang der Statistik Deutschlands. Jede Datenlieferung wird beim Empfänger auf Korrektheit und Vollständigkeit geprüft. Das Ergebnis dieser Überprüfung kann der Absender auch durch Angabe des zu der Datenlieferung gehörenden Eingangsstempels anfordern. Das Prüfprotokoll wird im XML-Format DatML/RES [s. SPEZ] geliefert.

Weitere Informationen zu der Kommunikationsschnittstelle des gemeinsamen Dateneingangs von eSTATISTIK.core finden sich in der zugehörigen Beschreibung beim Erhebungsportal https://erhebungsportal.estatistik.de/Erhebungsportal/# → grauer Bereich „Hilfsmittel und Automatisierung“ → Unterstützung für Entwickler → Spezifikation zu .CORE → CORE - Kommunikationsschnittstelle.

4 Überblick über die Java-Pakete und -Klassen

Gemäß dem dargestellten Funktionsumfang erfolgt die Aufgliederung in die folgenden Softwarebibliotheken, Funktionsbereiche und Java-Pakete:

Bibliothek	Funktionsbereich	Paket
CORE.connect	Kommunikation	de.destatis.core.connect
	Dokumente	de.destatis.core.connect.document
	Generator	de.destatis.core.generator
	Ressourcen	de.destatis.core.resource
CORE.inspector	Prüfung	de.destatis.core.inspector

Für die Realisierung der Softwarebibliothek zu .CORE wurde die folgende Open-Source-Software verwendet:

- Apache HttpCore 4.2.5, URL: http://hc.apache.org/httpcomponents-core-ga
- Xerces, URL: http://xml.apache.org
- Jakarta Commons, URL: http://jakarta.apache.org/commons

4.1 **Paket** `de.destatis.core.connect`

Dieses Paket umfasst alle Klassen und Interfaces zur Implementierung der Kommunikationsschnittstelle des gemeinsamen Dateneingangs.

Name der Klasse	Beschreibung
Client	Hauptklasse des Pakets. Mit Hilfe dieser Klasse können Datenlieferungen, Prüfungs-, Protokoll- und Erhebungsbeschreibungsanfragen an den gemeinsamen Dateneingang der amtlichen Statistik gesendet werden.
CheckDeliveryRequest	Prüfen eines DatML/RAW-Dokuments durch den gemeinsamen Dateneingang.
ResponseToCheckDeliveryRequest	Ergebnis einer formalen Datenprüfung. Bei erfolgreicher Ausführung wird das Prüfprotokoll im XML-Format (DatML/RES) übermittelt.
DataDelivery	Erzeugen einer DatML/RAW-Datenlieferung für das Senden an den gemeinsamen Dateneingang.
ResponseToDataDelivery	Enthält Antwort vom gemeinsamen Dateneingang auf die Übermittlung einer DatML/RAW-Datenlieferung. Über diese Klasse können bei erfolgreicher Ausführung der Eingangsstempel und das Prüfprotokoll der Datenlieferung abgefragt werden.
ProtocolRequest	Erzeugen einer Anfrage an den gemeinsamen Dateneingang von eSTATISTIK.core zum Herunterladen eines Prüfprotokolls zu einer vorher versendeten Datenlieferung.
ResponseToProtocolRequest	Enthält die Antwort vom gemeinsamen Dateneingang auf die Anforderung eines DatML/RES-Prüfprotokolls.
GetSurveyResourceRequest	Erzeugen einer Anfrage zum Herunterladen einer gültigen Ressource zur Erhebung.
ResponseToGetResourceRequest	Enthält die Antwort auf eine Anfrage zum Herunterladen einer Ressource. Liefert die aktuellste Ressource.
ResponseStatus	Stellt Informationen über den Erfolg einer Anfrage (Status) zur Verfügung.
ClientStatusEvent	Stellt Informationen über den aktuellen Status einer Anfrage zur Verfügung.
UserPasswordAuthenticator	Authentifikationsklasse, die die Authentifizierungs-informationen für die Anmeldung am gemeinsamen Dateneingang bereitstellt.
PingRequest	Anfrage zum Testen der Verbindung zum gemeinsamen Dateneingang von .CORE.
ResponseToPingRequest	Zeigt das Ergebnis der Verbindung zum gemeinsamen Dateneingang an.

Name des Interface	Beschreibung
ClientAuthenticator	Schnittstelle für die Bereitstellung von Kennung und Passwort für die Authentifizierung beim CORE-Server.
ClientStatusListener	Schnittstelle für die Bereitstellung von aktuellen Statusinformationen über die Kommunikation mit dem CORE-Server.

4.2 **Paket** `de.destatis.core.connect.document`

Dieses Paket umfasst alle Klassen und Interfaces zur Implementierung der verschiedenen verwendeten Dokumenttypen aus der DatML-Familie. Das Paket besteht aus den folgenden wichtigsten Klassen und Interfaces.

Name der Klasse	Beschreibung
DatMLRawDocument	Repräsentiert ein DatML/RAW-Dokument.
DatMLResDocument	Repräsentiert ein DatML/RES-Dokument.
ByteDocumentSource	Implementierung einer DocumentSource auf Basis eines byte[].
FileDocumentSource	Implementierung einer DocumentSource, welche die Daten des Dokuments aus einer Datei liest.
StringDocumentSource	Implementierung einer DocumentSource auf Basis eines Strings.
DocumentSourceFactory	Factory für die Erzeugung von DocumentSources.

Name des Interface	Beschreibung
DocumentSource	Schnittstelle für die Datenquelle eines XML-Dokumentes

4.3 Paket `de.destatis.core.generator`

Dieses Paket umfasst alle Klassen und Interfaces zur Implementierung der Generierung von DatML/RAW-Dokumenten über

- eine Abfrage-Schnittstelle (Spezifische Implementierung innerhalb des Statistikmoduls)
- eine bereits vorliegende Standardimplementierung, die die Vorgabe der Daten im CSV- bzw. im Properties-Format (Feldname-Wert-Zuordnung) ermöglichen
- die Bereitstellung einer graphischen Benutzeroberfläche zur Definition der Abbildung der Erhebungsmerkmale auf die Position in der CSV-Datei oder durch Verwendung eigener Feldbezeichnungen

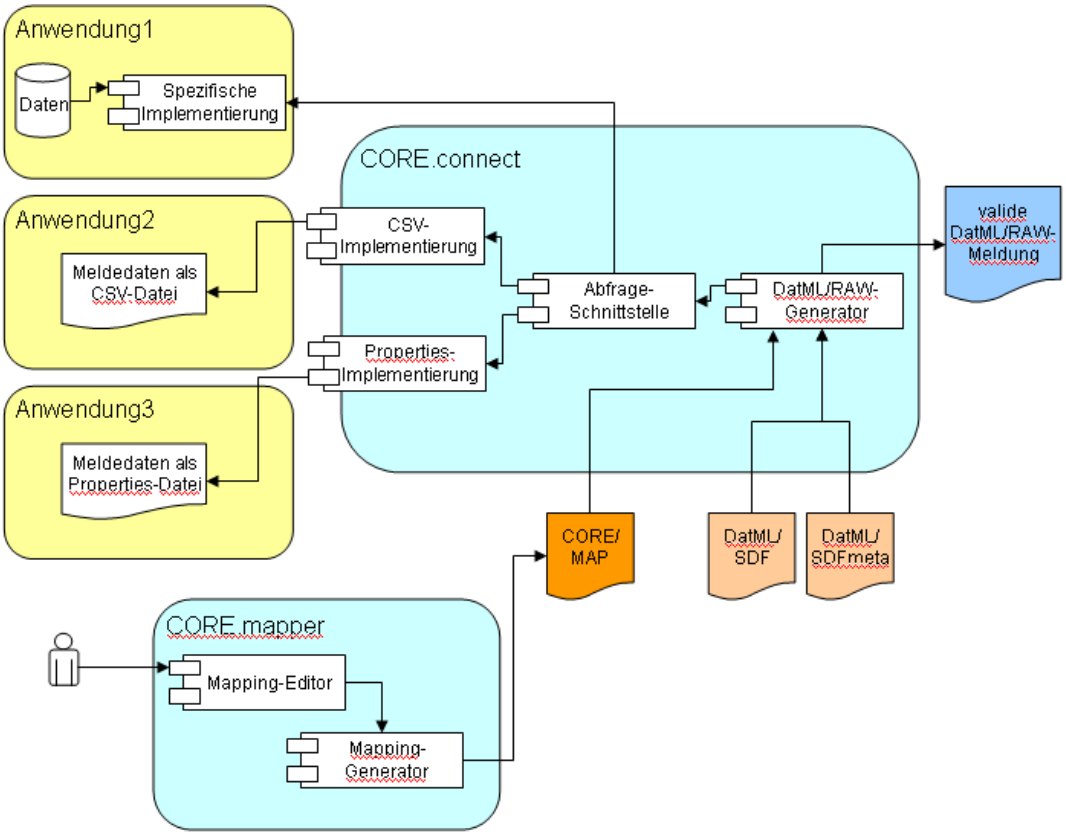

Abbildung 1: Datenfluß der erweiterten .CORE-Schnittstelle

Das Paket besteht aus den folgenden wichtigsten Klassen und Interfaces.

Name der Klasse	Beschreibung
CsvDataProvider	Implementierung eines DataProviders für CSV-Dateien. Setzt ein passendes Mapping und MetaMapping (Erstellung mittels des CORE.mappers) voraus.
MapDataProvider	Implementierung eines DataProviders, der die Werte aus einer Map ausliest (Properties-Datei mit Code-Wert-Zuweisungen).
DescriptorFactory	Fabrik für die Erzeugung von Descriptoren-Objekten.
FieldDescriptorImpl	Implementierung eines FieldDescriptors.
StructureDescriptorImpl	Implementierung eines StructureDescriptors.
SurveyDataDocumentGenerator	Diese Klasse erledigt die Generierung eines Lieferdatendokuments.

Name des Interface	Beschreibung
DataProvider	Schnittstelle für den Zugriff auf die Daten, die in das Ziel-Format DatML/RAW überführt werden sollen. Es wird dabei grundsätzlich zwischen Metadaten und statistikrelevanten Daten unterschieden.
Descriptor	Liefert allgemeine Informationen zum Feld bzw. Struktur (z.B. benutzerspezifischer Name, statistikspezifischer Name,…), die über die Schnittstelle DataProvider abgefragt werden.
FieldDescriptor	Liefert Informationen zum Feld (z. B. Datentyp, Position in der CSV-Datei,…), das über die Schnittstelle DataProvider abgefragt wird.
StructureDescriptor	Liefert Informationen zu Strukturen (z.B. Index für wiederholte Strukturen,…), die über die Schnittstelle DataProvider abgefragt werden.

4.3.1 Anwendungshinweise

Spezifische Implementierung (s. Abb. 1: Anwendung 1)

Hierzu ist das Interface `DataProvider` in der anwenderspezifischen Software einzubinden (Schnittstelle für den Zugriff auf die Metadaten/Daten) und die Methoden des Interfaces `DataProvider` zu implementieren.

- Methode `getDataProvider`(StructureDescriptor structure) dient zur Abfrage der Substrukturen (Nachrichten, Sätze, Merkmalsgruppen oder MetaMerkmalsgruppen).
- Methode `getFieldValue`(FieldDescriptor field) dient zur Abfrage der Metadaten und der statistikrelevanten Daten.
- Methode `close()` gibt die vom DataProvider erzeugten Ressourcen frei (Schließen von Streams etc.).

s. [BSP], `ExampleGenerationFromInterface.java`

Meldedaten als csv-Datei (s. Abb. 1: Anwendung 2)

Hierzu ist es erforderlich, das mittels des CORE.mappers (s. Softwarebibliothek CORE.connect im bin-Verzeichnis) das benötigte CORE/MAP erzeugt wird.

Vorgehensweise zur Erstellung von CORE/MAP

1. DatML/SDF oder DatML/SDFmeta in CORE.mapper einlesen
2. Datei ➔ Neu ➔ Mapping (für DatML/SDF)
3. Datei ➔ Neu ➔ Metadaten-Mapping (für DatML/SDFmeta)
4. evtl. Anpassungen am Mapping bzw. Metadaten-Mapping vornehmen
5. CORE/MAP für SDF bzw. SDFmeta erzeugen mittels Speichern

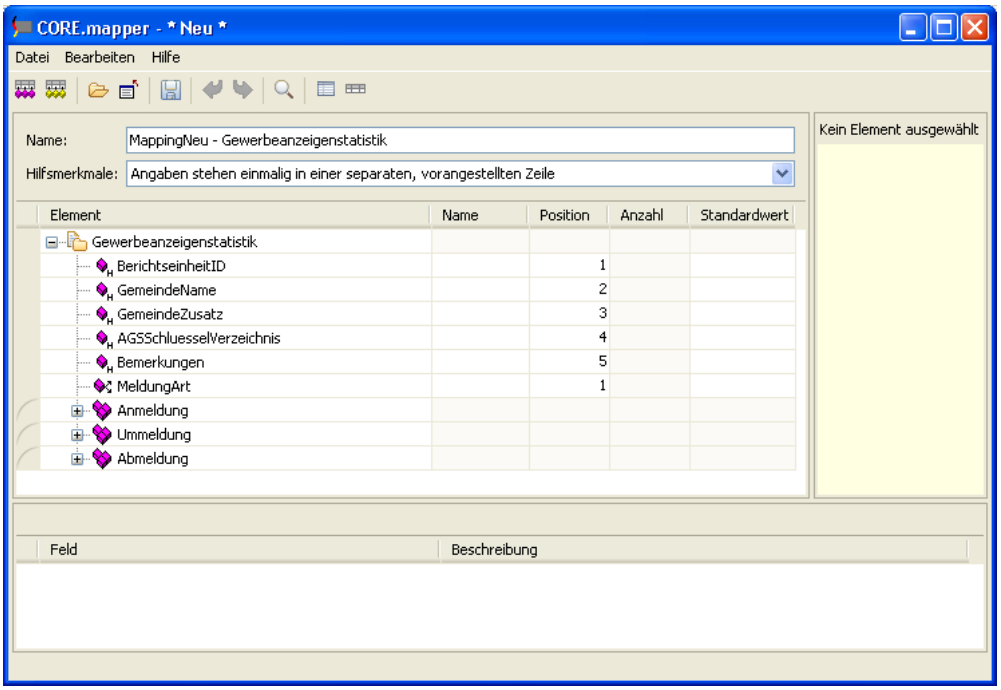

Abbildung 2: CORE.mapper zur Erzeugung von CORE/MAP

Erforderlicher struktureller Aufbau einer Csv-Datei

1. Die erste Zeile enthält die Metadaten des Dokuments

```
"0";"Testanwendung";"10";"Testfirma";"Kennung";"Organisation";"Niederlassung;"Zusatz";
...
```

2. Die zweite Zeile enthält die Metadaten der Nachricht

```
"0004";"200701";"00";"0001112";"Org";"";"";"";"";"Test";"Muster";"";"Straße";"1a";...
```

3. In der dritten Zeile folgen (je nach Konfiguration des übergebenen Mappings) ggf. die Hilfsmerkmale der Nachricht

```
"12345678";"GemeindeName";"GeZu";"AGS011234";"Bemerkungen"
```

4. Es folgen die Zeilen mit den Sätzen der Nachricht

```
"1";"0";"1";"GemSchlu";"GeZu";"GemMeldNr1";"12012010";"Verdachtsmomente1";"FKS011234";
...
"1";"0";"1";"GemSchl1";"GeZ1";"GemMeldNr2";"12012011";"Verdachtsmomente2";"FKS011235";
...
"1";"0";"1";"GemSchlu";"GeZu";"GemMeldNr3";"12012010";"Verdachtsmomente3";"FKS011234";
...
```

s. [BSP], `ExampleGenerationFromCSV.java`

Meldedaten als Properties-Datei (s. Abb. 1: Anwendung 3)

Die Properties-Implementierung ermöglicht die Erzeugung von DatML/RAW-Meldungen aus Feldname-Wert-Zuordnungstabellen (Properties-Dateien).

```
Lieferungsart=0
Anwendung.Anwendungsname=Testanwendung
Anwendung.Version=10
Anwendung.Hersteller=Testfirma
Absender.Kennung=Kennung
Absender.Organisation.Name=Organisation
Absender.Organisation.Niederlassung=Niederlassung
Absender.Organisation.Zusatz=Zusatz
Absender.Person.Anrede=
Absender.Person.Titel=
Absender.Person.Vorname=Vorname
Absender.Person.Nachname=Nachname
Absender.Person.Zusatz=
Absender.Adresse.Strasse=Test-Str.
Absender.Adresse.Hausnummer=3
Absender.Adresse.Postfach=
Absender.Adresse.Postfachleitzahl=
Absender.Adresse.Postfachort=
Absender.Adresse.Postleitzahl=12345
Absender.Adresse.Ort=Teststadt
Absender.Adresse.Kreis=TestKreis
Absender.Adresse.Bundesland=Hessen
Absender.Adresse.Land=Deutschland
...
```

s. [BSP], `ExampleGenerationFromProperties.java`

4.4 Paket `de.destatis.core.resource`

Dieses Paket umfasst alle Klassen und Interfaces zur Implementierung des Abrufes von aktuellen Ressourcen (DatML/SDF, DatML/SDFmeta, …) aus der Erhebungsdatenbank (URL: https://erhebungsdatenbank.estatistik.de/eid/) des Bundes und der Länder. Das Paket besteht aus den folgenden wichtigsten Klassen und Interfaces.

Name der Klasse	Beschreibung
ResourceConfig	Stellt allgemeine Eigenschaften, die innerhalb von CORE.connect für die Verwaltung der Ressourcen von verschiedenen Komponenten (Inspector, SurveyDataDocumentGenerator) benötigt werden, zur Verfügung.
ResourceType	Die Klasse kapselt den Typ (z.B. DatML/SDF) einer Ressource.
DatmlSdfSource	Enthält die RessourcenID und liefert einen Reader auf die ermittelte Erhebungsbeschreibung.
SDFMetaResource	Liefert die DatML/SDFmeta-Ressource.

Name des Interface	Beschreibung
ResourceStorage	Schnittstelle für die Ablage von Ressourcen.
SurveyResourceProvider	Schnittstelle für den Zugriff auf Erhebungsbeschreibungen z.B. durch den Inspector für die formale Datenprüfung (Prüfstufe 4) oder durch den SurveyDataDocumentGenerator.

4.5 **Paket** `de.destatis.core.inspector`

Dieses Paket umfasst alle Klassen zur Implementierung des Inspector-Moduls. Das Inspector-Modul bietet die Möglichkeit, ein DatML/RAW-Dokument hinsichtlich folgender Kriterien zu prüfen:

1. Syntax: Wohlgeformtheit und Gültigkeit gemäß XML-Format DatML/RAW.
2. Semantik: Einhaltung aller weiteren in der DatML/RAW-Spezifikation festgelegten Formatvorgaben.
3. Autorisierung: Einhaltung allgemeiner, nicht statistikspezifischer Vorgaben zur Datenlieferung mit DatML/RAW.
4. Daten: Einhaltung der Vorgaben lt. der zur jeweiligen Meldung gehörenden Erhebungsbeschreibung, also z.B. Anzahl und Art der anzugebenden Merkmale und Merkmalsgruppen, Angabe des Berichtszeitraums und der Erhebung.

Eine detaillierte Beschreibung des Inspector-Moduls findet sich in Kap. 7. Das Paket besteht aus den folgenden wichtigsten Klassen und Interfaces.

Name der Klasse	Beschreibung
Inspector	Hauptklasse des Inspector-Moduls, die die Prüffunktionen des Inspector-Moduls implementiert.
InspectionReport	Vom Inspector-Modul erstellte Zusammenfassung der Prüfergebnisse eines DatML/RAW-Dokuments.
InspectionProblem	Beschreibt ein bei der Prüfung festgestellte Unstimmigkeit. Hierbei kann es sich um einen Hinweis, Warnung oder Fehler handeln.
ProblemPosition	Beschreibt die Position einer festgestellten Unstimmigkeit in dem geprüften DatML/RAW-Dokument.

HINWEIS:

Alle weiteren Klassen aus dem Paket `de.destatis.core.inspector` wurden hauptsächlich zur Verwendung am gemeinsamen Dateneingang entwickelt und werden aus diesem Grunde nicht weiter beschrieben.

5 Konfigurationseinstellungen

Die Funktionen der Softwarebibliothek CORE.connect und CORE.inspector werden über Konfigurationseinstellungen gesteuert. Diese Einstellungen können, sofern sie von den Standardeinstellungen abweichen, bei der Instanzierung der beiden Klassen `Client` und `Inspector` in Form von `Properties`-Objekten mitgegeben sowie teilweise über entsprechende Methoden individuell eingestellt werden.

Im Folgenden werden alle Konfigurationseinstellungen mit ihrem Schlüssel, den möglichen Schlüsselwerten sowie ihre Bedeutung beschrieben.

Schlüssel:	`client.username`
Java-Konstante:	`PROPERTY_USERNAME`
Wertebereich:	Beliebige Zeichenkette
Standardwert:	-
Bedeutung:	Mit diesem Schlüssel wird die Benutzerkennung eingestellt, die bei der Kommunikation mit dem gemeinsamen Dateneingang verwendet werden soll.

Schlüssel:	`client.password`
Java-Konstante:	`PROPERTY_PASSWORD`
Wertebereich:	Beliebige Zeichenkette
Standardwert:	-
Bedeutung:	Mit diesem Schlüssel wird das Benutzerpasswort eingestellt, das bei der Kommunikation mit dem gemeinsamen Dateneingang verwendet werden soll.

Schlüssel:	`client.compression`
Java-Konstante:	`PROPERTY_COMPRESSION`
Wertebereich:	none, deflate, gzip
Standardwert:	none
Bedeutung:	Mit diesem Schlüssel kann eingestellt werden, ob die Daten komprimiert übertragen werden sollen.

Schlüssel:	`client.address.useURI`
Java-Konstante:	`PROPERTY_ADDRESS_USE_URI`
Wertebereich:	false, true
Standardwert:	false
Bedeutung:	Mit diesem Schlüssel kann eingestellt werden, ob für den Verbindungsaufbau zum gemeinsamen Dateneingang die IP-Adresse oder der DNS-Name des CORE-Servers verwendet werden soll.

Schlüssel:	`client.protocol`
Java-Konstante:	`PROPERTY_PROTOCOL`
Wertebereich:	https, osci
Standardwert:	https
Bedeutung:	Mit diesem Schlüssel kann eingestellt, welches Protokoll für die Datenübertragung an den gemeinsamen Dateneingang verwendet werden soll. Das Übertragungsprotokoll `osci` wird z.Zt. noch nicht unterstützt.

Schlüssel: `client.certificate.dir`

Java-Konstante: `PROPERTY_CERTIFICATE_DIR`

Wertebereich: Gültiger Verzeichnispfad

Standardwert: Internes Root-Zertifikat verwenden

Bedeutung: Mit diesem Schlüssel kann eingestellt werden, in welchem Verzeichnis weitere Root-Zertifikate für die Überprüfung des Server-Zertifikats des gemeinsamen Dateneingangs gesucht werden sollen. Diese Einstellung ist eine systemweite Einstellung, die beim Start der JVM als System-Property gesetzt und danach nicht mehr geändert werden kann.

Die beiden folgenden Schlüssel zur Protokollierung (`client.log.file` und `client.log.level`) stehen nur bei Verwendung von Log4J in der Version 1.2.X zur Verfügung.

Hinweis: Mit der Version 1.10.0 von CORE.connect wurde die Abhängigkeit zu Log4J grundsätzlich entfernt. Auch die log4j.jar-Datei wurde aus der Lieferung entfernt. Falls die einbindende Software weiterhin die Version 1.2.X von Log4J verwendet, bestehen keine Funktionseinschränkungen und es muss beim Austausch der Bibliothek nichts weiter beachtet werden.

Die Protokollierung innerhalb der Bibliothek erfolgt nun über Apache-Commons-Logging.

Wenn für die Protokollierung nicht Log4J in der Version 1.2.X verwendet wird, steht diese Funktion zur Logdatei nicht mehr zur Verfügung.

Schlüssel: `client.log.file`

Java-Konstante: `PROPERTY_LOG_FILE`

Wertebereich: Gültiger Dateiname oder CONSOLE

Standardwert: -

Bedeutung: Mit diesem Schlüssel kann der Name einer Datei angegeben werden, in die alle Protokollmeldungen gemäß der eingestellten Protokollstufe gespeichert werden. Bei Angabe von CONSOLE werden die Meldungen auf der Konsole ausgegeben. Wenn keine Protokolldatei angegeben ist, erfolgt auch keine Protokollierung. Diese Einstellung ist eine systemweite Einstellung, die beim Start der JVM als System-Property gesetzt und danach nicht mehr geändert werden kann.

Schlüssel: `client.log.level`

Java-Konstante: `PROPERTY_LOG_LEVEL`

Wertebereich: 0, 1, 2

Standardwert: 0

Bedeutung: Mit diesem Schlüssel kann der Detaillierungsgrad der Protokollierung eingestellt werden. Je höher der Detaillierungsgrad, desto detaillierter sind die Protokollmeldungen. Bei der Stufe 2 (=Debug) werden auch die zwischen Client und Server ausgetauschten Daten protokolliert. Wenn keine Protokolldatei angegeben ist, erfolgt auch keine Protokollierung. Diese Einstellung ist eine systemweite Einstellung, die beim Start der JVM als System-Property gesetzt und danach nicht mehr geändert werden kann.

Die folgenden Konfigurationseinstellungen sind nur relevant, wenn die Kommunikation mit dem gemeinsamen Dateneingang über einen Proxy-Server erfolgt.

Schlüssel: `client.proxy.user`

Java-Konstante: `PROPERTY_PROXY_USER`

Wertebereich: Beliebige Zeichenkette

Standardwert:	-
Bedeutung:	Mit diesem Schlüssel wird die Benutzerkennung eingestellt, die für die Authentifizierung bei einem Proxy-Server verwendet werden soll.

Schlüssel:	`client.proxy.password`
Java-Konstante:	`PROPERTY_PROXY_PASSWORD`
Wertebereich:	Beliebige Zeichenkette
Standardwert:	-
Bedeutung:	Mit diesem Schlüssel wird das Benutzerpasswort eingestellt, das für die Authentifizierung bei einem Proxy-Server verwendet werden soll.

Schlüssel:	`client.proxy.host`
Java-Konstante:	`PROPERTY_PROXY_HOST`
Wertebereich:	Gültiger DNS-Name oder IP-Adresse
Standardwert:	-
Bedeutung:	Mit diesem Schlüssel wird der Netzwerkname des Proxy-Servers eingestellt, über den die Kommunikation mit dem gemeinsamen Dateneingang erfolgen soll.

Schlüssel:	`client.proxy.port`
Java-Konstante:	`PROPERTY_PROXY_PORT`
Wertebereich:	Gültige Portnummer
Standardwert:	80
Bedeutung:	Mit diesem Schlüssel wird der Port des Proxy-Servers eingestellt, über den die Kommunikation mit dem gemeinsamen Dateneingang erfolgen soll.

Schlüssel:	`client.proxy.authmethod`
Java-Konstante:	`PROPERTY_PROXY_AUTH_METHOD`
Wertebereich:	auto, basic, digest, ntlm
Standardwert:	auto
Bedeutung:	Mit diesem Schlüssel wird eingestellt, welche Authentifizierungsmethode beim Proxy-Server bei der Kommunikation mit dem gemeinsamen Dateneingang zu verwenden ist. Bei der Methode `NTLM` wird nur die Version 1 unterstützt. Bei Angabe des Wertes `auto` wird die Authentifizierungsmethode automatisch bestimmt, indem eine Authentifizierung in der Reihenfolge `ntlm`, `digest` und `basic` versucht wird.

Schlüssel:	`client.proxy.domain`
Java-Konstante:	`PROPERTY_PROXY_DOMAIN`
Wertebereich:	Gültige Domainname
Standardwert:	-
Bedeutung:	Mit diesem Schlüssel wird der bei der Authentifizierungsmethode `NTLM` benötigte Domainname eingestellt. Bei allen anderen Authentifizierungsmethoden wird dieser Schlüssel ignoriert.

Die folgenden Konfigurationseinstellungen steuern das Verhalten des Inspector-Moduls, mit dem weitreichende Prüfungen von DatML/RAW-Dokumenten vor dem Versand durchgeführt werden können.

Schlüssel:	`inspector.inspectionLevel`
Java-Konstante:	`PROPERTY_INSPECTION_LEVEL`
Wertebereich:	1, 2, 3, 4
Standardwert:	2
Bedeutung:	Mit diesem Schlüssel wird die Prüfstufe bei der Inspektion von DatML/RAW-Daten eingestellt. Bei Stufe 4 werden die Daten am strengsten geprüft

Schlüssel:	`inspector.survey.dir`
Java-Konstante:	`PROPERTY_SURVEY_DIR`
Wertebereich:	Gültiger Verzeichnispfad
Standardwert:	-
Bedeutung:	Mit diesem Schlüssel wird das Verzeichnis eingestellt, in dem bei der Inspektion von DatML/RAW-Daten nach Erhebungsbeschreibungen gesucht wird. Diese Einstellung ist nur relevant, wenn die Prüfstufe 4 verwendet wird.

Schlüssel:	`inspector.schema.dir`
Java-Konstante:	`PROPERTY_SCHEMA_DIR`
Wertebereich:	Gültiger Verzeichnispfad
Standardwert:	Interne Schemadateien
Bedeutung:	Mit diesem Schlüssel wird das Verzeichnis eingestellt, in dem bei der Validierung von XML-Daten nach den notwendigen XML-Schemadateien gesucht wird. Im Normalfall sind die intern bereitgestellten Schemadateien hinreichend.

Schlüssel:	`inspector.survey.autoupdate`
Java-Konstante:	`PROPERTY_SURVEY_AUTOUPDATE`
Wertebereich:	no, yes
Standardwert:	no
Bedeutung:	Mit diesem Schlüssel wird eingestellt, ob bei der Inspektion von DatML/RAW-Daten automatisch nach aktuelleren Erhebungsbeschreibungen auf dem CORE-Server gesucht werden soll.

6 Statuscodes des gemeinsamen Dateneingangs

Bei der Kommunikation mit dem gemeinsamen Dateneingang enthält das durch die `Client`-Methode `getStatus` gelieferte `ResponseStatus`-Objekt einen anwendungsspezifischen Statuscode, der Auskunft über den Erfolg der jeweiligen Anfrage gibt. Dieser Statuscode kann die folgenden Werte und Bedeutungen haben (s. [KOMM] oder Java-Doc von `ResponseStatus`).

Statuscode (Wert, Java-Konstante)	Bedeutung
0 = OK	Es ist kein Fehler aufgetreten. Die Verarbeitung der Anfrage beim gemeinsamen Dateneingang war erfolgreich.
10 = BAD_REQUEST	Die Art und Anzahl der übermittelten `multipart/form-data`-Teile und/oder Feldnamen waren nicht korrekt. Es wurden z.B. Felder vergessen oder falsch angegeben.
20 = LOGIN_ERROR	Bei der Authentifizierung beim gemeinsamen Dateneingang ist ein Fehler aufgetreten, z.B. weil Kennung und Passwort falsch angegeben wurden.
30 = BAD_RESPONSE	Die vom gemeinsamen Dateneingang empfangene Antwort ist fehlerhaft und kann nicht korrekt weiterverarbeitet werden.
100 = FILE_RECEIVE_ERROR	Die übermittelten Daten konnten nicht verarbeitet werden, z.B. weil der angegebene Zeichensatz unbekannt ist oder die Daten zu umfangreich.
110 = NO_VALID_XML	Die übermittelten Daten sind kein gültiges DatML/RAW-Dokument.
200 = RES_NOT_AVAILABLE	Für die angegebene Datenlieferung ist noch kein Prüfprotokoll vorhanden, weil es noch nicht erstellt ist. Der Abruf des Prüfprotokolls sollte später wiederholt werden.
210 = RES_FORMAT_ERROR	Die angegebene Version des Formats DatML/RES wird nicht (mehr) unterstützt.
220 = RES_INVALID_ID	Dem gesendeten Eingangsstempel ist keine Datenlieferung des Absenders zugeordnet, wodurch auch kein Prüfprotokoll bereitgestellt werden kann.
230 = RES_DELETED	Für die angegebene Datenlieferung war ein Prüfprotokoll vorhanden, das aber inzwischen wegen Ablauf der Aufbewahrungsfrist gelöscht wurde.
300 = SDF_NOT_AVAILABLE	Für die angegebene Erhebung und Berichtszeitraum ist keine Erhebungbeschreibung vorhanden, z.B. weil eine ungültige Erhebung oder Berichtszeitraum angegeben wurde.
310 = SDF_FORMAT_ERROR	Die angegebene Version des Formats DatML/SDF wird nicht (mehr) unterstützt.
320 = SDF_UPTODATE	Für die angegebene Erhebung und Berichtszeitraum liegt keine aktuellere Erhebungsbeschreibung vor.

Wenn ein Fehler auf der Ebene des Übertragungsprotokolls, also HTTP(S), auftritt, dann werden der HTTP-Statuscode und -Fehlertext in der dabei ausgelösten Ausnahme `TransferProtocol-Exception` mitgeteilt. Ein `ResponseStatus`-Objekt steht in einem solchen Fall nicht zur Verfügung.

7 Fehlerschlüssel und -meldungen des Inspector-Moduls

Das Inspector-Modul (Klasse `Inspector`) ist eine konfigurierbare Komponente der Softwarebibliothek CORE.inspector, mit der DatML/RAW-Daten vor dem Versand einer eingehenden Prüfung unterzogen werden können. Diese Prüfungen beruhen nicht nur auf den Angaben in der XML-Schemadatei von DatML/RAW, sondern können auch die zu jeder Statistik bereitgestellten Erhebungsbeschreibungen (DatML/SDF) nutzen.

HINWEIS:

Die Softwarebibliothek CORE.inspector benötigt Ressourcen, die über das Package *de.destatis.core.resource* der CORE.connect-Bibliothek zu Verfügung gestellt werden.

Das Inspector-Modul untersucht DatML/RAW-Daten gemäß vorgegebener Prüfstufen und liefert im Ergebnis einen Prüfbericht (Klasse `InspectionReport`) mit einer Liste von Prüfergebnissen (Klasse `InspectionProblem`). Die Prüfung von DatML/RAW-Daten erfolgt auf den folgenden Prüfebenen:

1. Syntax: Wohlgeformtheit und Gültigkeit gemäß XML-Format DatML/RAW.
2. Semantik: Einhaltung aller weiteren in der DatML/RAW-Spezifikation festgelegten Formatvorgaben.
3. Autorisierung: Einhaltung allgemeiner, nicht statistikspezifischer Vorgaben zur Datenlieferung mit DatML/RAW.
4. Daten: Einhaltung der Vorgaben lt. der zur jeweiligen Meldung gehörenden Erhebungsbeschreibung, also z.B. Anzahl und Art der anzugebenden Merkmale und Merkmalsgruppen, Angabe des Berichtszeitraums und der Erhebung.

Aufbauend auf diesen Prüfebenen definieren sich die vier Prüfstufen des Moduls:

Prüfstufe	Prüfebenen
1	Syntax (1)
2	Syntax (1) + Semantik (2)
3	Syntax (1) + Semantik (2) + Autorisierung (3)
4	Syntax (1) + Semantik (2) + Autorisierung (3) + Daten (4)

Das Modul erhält das zu untersuchende DatML/RAW-Dokument und die zu verwendende Prüfstufe und liefert darauf hin einen entsprechenden Prüfbericht. Der Prüfbericht besteht aus summarischen Angaben zur Prüfung sowie einer Liste mit detaillierten Prüfergebnissen. Die Prüfergebnisse sind in vier Problemklassen unterteilt:

Problemklasse	Code=Kürzel	Bedeutung
Information	1=INFO	Information oder Hinweis
Warnung	2=WARN	Warnung
Fehler	3=ERROR	Fehler
Abbruch	4=FATAL	Nicht behebbarer Fehler

Neben der Problemklasse enthält ein Prüfergebnis die folgenden Informationen:

1. die Prüfebene, auf der das Problem festgestellt wurde,
2. einen eindeutigen Schlüssel zur Identifikation des Problems,
3. einer textuelle Erläuterung des Problems, sowie
4. die Position im Dokument, an der das Problem festgestellt wurde.

Die Position (Klasse `ProblemPosition`) im Dokument, an der ein Problem festgestellt wurde, wird in logischer Form durch eine Nachrichten-, Meldungs- und Satznummer sowie einen absoluten XPath-Ausdruck und eine Zeilen- und Spaltenposition angegeben. Wenn sich ein Problem keiner Nachricht oder Meldung oder Satz zuordnen lässt, dann wird als Nummer 0 verwendet.

Im Folgenden werden die einzelnen Prüfebenen und die möglichen Prüfergebnisse mit Problemkategorie, Problemschlüssel und Fehlertext erläutert.

7.1 Prüfebene: Syntax

Auf dieser Prüfebene wird das DatML/RAW-Dokument auf Wohlgeformtheit und Gültigkeit gemäß dem zugehörigen XML-Schema geprüft.

Für die Ergebnisse der Überprüfung werden die folgenden Schlüssel und Texte verwendet:

Kategorie	Schlüssel	Text
WARN	12001	Hier wird der Originaltext der `SAXParseException` übernommen. Alle Warnungen haben nur einen Schlüssel.
ERROR	13001	Hier wird der Originaltext der `SAXParseException` übernommen. Alle Fehler haben nur einen Schlüssel.
FATAL	14001	Hier wird der Originaltext der `SAXParseException` übernommen. Alle nicht behebbaren Fehler haben nur einen Schlüssel.

Die verschiedenen Arten von Parserfehlern werden in den Schlüsseln nicht weiter ausdifferenziert. Alle Parserfehler einer Kategorie erhalten einen einheitlichen Schlüssel. Zur Identifikation der Position des Prüfergebnisses werden die Nachrichten-, Meldungs- und Satznummer ausgewiesen sowie der absolute XPath-Ausdruck und eine Zeilen- und Spaltenposition. Die Zeilen- und Spaltennummer aus der `SAXParseException` wird in den Text übernommen.

7.2 Prüfebene: Semantik

Auf dieser Prüfebene wird das DatML/RAW-Dokument auf die Einhaltung aller weiteren, nicht im XML-Schema beschriebenen, Anforderungen geprüft (s. [RAW]). Das sind:

Für die Ergebnisse der Überprüfung werden die folgenden Schlüssel und Texte verwendet:

Kategorie	Schlüssel	Text
WARN	22001	Die Anzahl der Nachrichten muss mit der Angabe im Element `anzahl` übereinstimmen.
ERROR	23001	Jeder Meldung muss genau ein Element `erhebung` zugeordnet sein.
ERROR	23002	Jeder Meldung muss genau ein Element `berichtszeitraum` zugeordnet sein.
ERROR	23003	Jeder Meldung darf höchstens ein Element `berichtspflichtiger` zugeordnet sein.
ERROR	23004	Jeder Meldung darf höchstens ein Element `berichtsempfaenger` zugeordnet sein.
ERROR	23005	Jeder Meldung darf höchstens ein Element `material` zugeordnet sein.
ERROR	23006	Jeder Meldung darf höchstens ein Element `datenattribute` zugeordnet sein.
ERROR	23007	Ein Hilfsmerkmal muss einen Namen (Attribut `name`) haben.
ERROR	23008	Die Hilfsmerkmale einer Meldung müssen verschiedene Namen (Attribut `name`) haben.
ERROR	23009	Ein Merkmal muss einen Namen (Attribut `name`) haben.
ERROR	23010	Ein Merkmal (Attribut `name`) darf nur einmal in einem Datensatz oder einer Merkmalsgruppe vorkommen.
ERROR	23011	Eine Merkmalsgruppe muss einen Namen (Attribut `name`) haben.

Kategorie	Schlüssel	Text
ERROR	23012	Der Index einer Merkmalsgruppe (Attribut `index`) darf nur numerisch sein oder ein indexbildendes Merkmal in der Form `name(<merkmalsname>)`.
ERROR	23013	Das indexbildende Merkmal muss in der Merkmalsgruppe vorkommen und darf nicht leer sein.
ERROR	23014	Der Wert des indexbildenden Merkmals muss numerisch sein.
ERROR	23015	Die Indizes einer Merkmalsgruppe müssen alle verschieden sein.
ERROR	23016	Die Indizierungsmethode muss für alle Instanzen einer Merkmalsgruppe gleich sein.

Zur Identifikation der Position des Prüfergebnisses werden die Nachrichten-, Meldungs- und Satznummer ausgewiesen sowie der absolute XPath-Ausdruck und eine Zeilen- und Spaltenangabe. Bei den Prüfergebnissen mit den Schlüsseln 23007 bis 23016 wird zusätzlich der Name des betroffenen Hilfsmerkmals, Merkmals bzw. Merkmalsgruppe angegeben.

7.3 **Prüfebene: Autorisierung**

Auf dieser Prüfebene wird das DatML/RAW-Dokument auf die Einhaltung allgemeiner, nicht statistikspezifischer, Vorgaben für DatML/RAW-Datenlieferungen geprüft.

Für die Ergebnisse der Überprüfung werden die folgenden Schlüssel und Texte verwendet:

Kategorie	Schlüssel	Text
WARN	32001	Der Berichtszeitraum ist der Zeitraum, für den berichtet wird, und muss daher mindestens begonnen haben. Nur bei Testmeldungen kann der Berichtszeitraum auch in der Zukunft liegen.
WARN	32002	Berichtsempfänger maschinell korrigiert.
ERROR	33002	Für die Testkennung sind nur die Werte 100 oder 200 zulässig.
ERROR	33003	Die Erhebungskennung muss einen der Werte 0001, …, 9999 enthalten.
ERROR	33004	Die Klasse der Erhebungskennung muss den Wert ERHID haben.
ERROR	33005	Die Absenderkennung darf nicht leer sein.
ERROR	33006	Die Klasse der Absender- und Berichtspflichtigenkennung muss den Wert MELDID haben.
ERROR	33007	Die Empfängerkennung muss angegeben werden.
ERROR	33008	Die Empfängerkennung muss einen der Werte 00,…, 16, 99 enthalten.
ERROR	33009	Bei Datenlieferungen an den gemeinsamen Dateneingang muss zu jeder Meldung ein Berichtsempfänger angegeben werden.
ERROR	33010	Wenn ein Berichtsempfänger angegeben ist, dann muss auch die Kennung angegeben werden.
ERROR	33011	Die Berichtsempfängerkennung muss einen der Werte 00,…, 16 enthalten.
ERROR	33012	Die Klasse der Empfänger- und Berichtsempfängerkennung muss den Wert STAID haben.
ERROR	33013	Der Name der Organisation oder Person muss angegeben werden.
ERROR	33014	Die Postleitzahl und der Ort müssen angegeben werden.
ERROR	33015	Der Berichtszeitraum ist der Zeitraum, für den berichtet wird, und muss daher mindestens begonnen haben.

Kategorie	Schlüssel	Text
ERROR	33016	Die Identifizierung der BerichtseinheitID ist nicht möglich (falsche oder fehlende BerichtseinheitID). Bitte übermitteln Sie die Meldung erneut unter der Angabe der ID, die Ihnen das zuständige Statistische Amt mitgeteilt hat.

Zur Identifikation der Position des Prüfergebnisses werden die Nachrichten- und Meldungsnummer ausgewiesen sowie der absolute XPath-Ausdruck und eine Zeilen- und Spaltenangabe.

7.4 **Prüfebene: Daten**

Auf dieser Prüfebene wird das DatML/RAW-Dokument auf Einhaltung der Vorgaben gemäß der zu jeder Meldung gehörenden Erhebungsbeschreibung geprüft.

Für die Ergebnisse der Überprüfung werden die folgenden Schlüssel und Texte verwendet:

Kategorie	Schlüssel	Text
WARN	42001	Es sind keine Erhebungsbeschreibungen vorhanden.
WARN	42002	Zu der Erhebung <eeee> sind keine Erhebungsbeschreibungen vorhanden.
WARN	42003	Für die Erhebung <eeee> und den Berichtszeitraum <Jahr, Halbjahr, Semester, Quartal, Monat, Woche> gibt es keine Erhebungsbeschreibung.
WARN	42004	Für die Erhebung <eeee> und den angegebenen Berichtszeitraum gibt es keine Erhebungsbeschreibung.
WARN	42005	Für die Erhebung <eeee> und den Berichtsempfänger <bb> gibt es keine Erhebungsbeschreibung.
WARN	42006	Der Wert des Hilfsmerkmals liegt nicht im definierten Wertebereich.
WARN	42007	Der Wert des Merkmals liegt nicht im definierten Wertebereich.
WARN	42008	Die zur Erzeugung des DatML/RAW-Dokumentes verwendeten DatML/SDF-Ressource (%VERSION%) stimmt nicht mit der zur Prüfung des DatML/RAW-Dokumentes verwendeten DatML/SDF-Ressource (%VERSION%) überein.
ERROR	43001	Das Format der Erhebungsbeschreibung ist kein gültiges DatML/SDF.
ERROR	43002	Dateiname und Inhalt der Erhebungsbeschreibung passen nicht zusammen.
ERROR	43003	Die verwendete Notationssprache für Bedingungen wird nicht unterstützt.
ERROR	43004	Der Wert des Hilfsmerkmals liegt nicht im definierten Wertebereich.
ERROR	43005	Das Hilfsmerkmal muss angegeben werden.
ERROR	43006	Die Bedingung für die Angabe des Hilfsmerkmals ist erfüllt, so dass das Hilfsmerkmal angegeben werden muss.
ERROR	43007	Die Bedingung für die Angabe des Hilfsmerkmals ist nicht erfüllt und strikt anzuwenden, so dass das Hilfsmerkmal nicht angegeben werden darf.
ERROR	43008	Das Hilfsmerkmal ist an dieser Stelle unzulässig.
ERROR	43009	Der Satz muss einer der definierten Satzarten entsprechen.
ERROR	43010	Der Wert des Merkmals liegt nicht im definierten Wertebereich.
ERROR	43011	Das Merkmal muss angegeben werden.
ERROR	43012	Die Bedingung für die Angabe des Merkmals ist erfüllt, so dass das Merkmal angegeben werden muss.
ERROR	43013	Die Bedingung für die Angabe des Merkmals ist nicht erfüllt und strikt anzuwenden, so dass das Merkmal nicht angegeben werden darf.
ERROR	43014	Das Merkmal ist an dieser Stelle unzulässig.

Kategorie	Schlüssel	Text
ERROR	43015	Die Merkmalsgruppe muss <direkt, indirekt, durch Name> indiziert werden.
ERROR	43016	Das angegebene indexbildende Merkmal ist für die Merkmalsgruppe unzulässig.
ERROR	43017	Von der Merkmalsgruppe müssen/dürfen mindestens/genau/höchstens <n> Instanzen angegeben werden.
ERROR	43018	Die Merkmalsgruppe muss angegeben werden.
ERROR	43019	Die Bedingung für die Angabe der Merkmalsgruppe ist erfüllt, so dass die Merkmalsgruppe angegeben werden muss.
ERROR	43020	Die Bedingung für die Angabe der Merkmalsgruppe ist nicht erfüllt und strikt anzuwenden, so dass die Merkmalsgruppe nicht angegeben werden darf.
ERROR	43021	Die Merkmalsgruppe ist an dieser Stelle unzulässig.
ERROR	43022	Es müssen/dürfen mindestens/genau/höchstens <n> Datensätze geliefert werden.
FATAL	44001	Das konfigurierte Verzeichnis mit den verfügbaren Erhebungsbeschreibungen ist ungültig oder nicht vorhanden.

Zur Identifikation der Position des Prüfergebnisses werden die Nachrichten-, Meldungs- und Satznummer ausgewiesen sowie der absolute XPath-Ausdruck und eine Zeilen- und Spaltenangabe.

8 Weitere Dokumentation

Für die Softwarebibliotheken CORE.connect und CORE.inspector befinden sich im Verzeichnis `doc` die zugehörigen Javadoc-Dokumentationen.

8.1 Beispiele

Für die Softwarebibliotheken CORE.connect und CORE.inspector steht im Erhebungsportal unter der URL https://erhebungsportal.estatistik.de/Erhebungsportal/# → grauer Bereich „Hilfsmittel und Automatisierung“ → Unterstützung für Entwickler → Spezifikation zu .CORE → CORE - Kommunikationsschnittstelle → . . . für JAVA-Entwickler → Thema „Java-Projekt mit Beispielimplementierungen“ ein JAVA-Projekt mit diversen Beispielen bereit (s. a. [BSP]).

8.1.1 Beispiele CORE.connect

Für die Softwarebibliothek CORE.connect befinden sich im Verzeichnis `beispiele/client` Beispieldaten und Beispielprogramme zum Versand einer Datenlieferung für die Vierteljährliche Verdiensterhebung (`vve*.xml`), den Monatsbericht im Verarbeitenden Gewerbe (`mbb*.xml`) und die Monatserhebung im Einzelhandel (`mhg-eh*.xml`), sowie zum Abrufen der Prüfprotokolle.

Die Konfigurationsdatei `client.ini` wird von den Beispielprogrammen zur Konfiguration verwendet. Die Batch-Datei `client.bat` zeigt, wie der Versand von Datenlieferungen (Aufruf s. `Start_SEND_Client.bat`) und der Abruf von Prüfprotokollen (Aufruf s. `Start_FETCH_Client.bat`) ohne zusätzliche Programmierung direkt von der Kommandozeile aus durchgeführt werden kann.

Im Verzeichnis `beispiele/generator` sind Beispieldaten und Beispielprogramme zur Generierung von DatML/RAW-Lieferdatendokumenten aus einer csv-Datei mit Meldedaten und aus einer Properties-Datei mit Meldedaten (für DatML/RAW-Version 2.0 und für DatML/RAW-Version 2.1) enthalten.

Die Konfigurationsdatei `conf/generator.properties` wird von den Beispielprogrammen zur Konfiguration verwendet.

Die Batch-Datei `testGeneratorCsv2_0.bat` zeigt die DatML/RAW-Generierung mit einer csv-Datei für DatML/RAW-Version 2.0 und die Batch-Datei `testGeneratorCsv2__1.bat` die DatML/RAW-Generierung mit einer csv-Datei für DatML/RAW-Version 2.1 auf. Hierzu wird CORE/MAP für SDF und SDFmeta (zu finden im Verzeichnis `beispiele/generator/mappings`) benötigt, welche mit Hilfe des CORE.mappers (siehe Softwarebibliothek CORE.connect im bin-Verzeichnis) erstellt wurden.

Die Batch-Datei `testGeneratorProperties2_0.bat` zeigt die DatML/RAW-Generierung mit einer Properties-Datei für DatML/RAW-Version 2.0 und die Batch-Datei `testGeneratorProperties2_1.bat` die DatML/RAW-Generierung mit einer Properties-Datei für DatML/RAW-Version 2.1 auf.

8.1.2 Beispiele CORE.inspector

Für die Softwarebibliothek CORE.inspector befinden sich im Verzeichnis `beispiele` Beispieldaten und Beispielprogramme zum Überprüfen von DatML/RAW-Lieferdatendokumenten für die Vierteljährliche Verdiensterhebung (`vve*.xml`), den Monatsbericht im Verarbeitenden Gewerbe (`mbb*.xml`) und die Monatserhebung im Einzelhandel (`mhg-eh*.xml`). Die Anforderungen an den Inhalt von Lieferungen für diese Statistiken werden in der zugehörigen Erhebungsbeschreibungen (`0001*.xml`, `0003*.xml` und `0018*.xml`) beschrieben.

Mit dem Batch-Programm `inspector.bat` können beliebige Datenlieferungen (DatML/RAW-Dokumente) auf ihre Richtigkeit bzgl. der zugehörigen Erhebungsbeschreibung überprüft werden (siehe auch Beispiel `Start-Inspector.bat`). Weitere Erhebungsbeschreibungen im DatML/SDF-Format sowie die zugehörigen Liefervereinbarungen im PDF-Format können vom Erhebungsportal

https://erhebungsportal.estatistik.de/Erhebungsportal/# → Unterstützung für Entwickler (grauer Bereich) → Statistiken mit Online-Verfahren heruntergeladen werden. oder direkt über die Erhebungsdatenbank https://erhebungsdatenbank.estatistik.de/eid/' heruntergeladen werden.

8.2 Informationen zum gemeinsamen Dateneingang

Für den Versand von Datenlieferungen an den gemeinsamen Dateneingang der Statistik Deutschlands werden eine Kennung und ein Passwort benötigt. Beides kann über die Webanwendung von eSTATISTIK.core mit folgender URL https://core.estatistik.de/core/ angefordert werden.

Über die Webanwendung zu eSTATISTIK.core erhält der Melder gebündelt alle Funktionen zu .CORE zur Verfügung gestellt, die es ihm ermöglichen, neben einer Registrierung für .CORE, Lieferungen vorab zu prüfen, Daten zu übermitteln und unmittelbar den korrekten Eingang der Daten zu überprüfen. Desweiteren stehen Funktionalitäten zur Address- und Passwortänderung, sowie das Melderkonto, das Informationen zu bereits gelieferten Datenmeldungen mit den entsprechenden Protokollen enthält, zur Verfügung.

Der Eingang und die Annahme (s. u. Hinweis) einer Datenlieferung beim gemeinsamen Dateneingang der statistischen Ämter kann vom Absender über die Webanwendung zu eSTATISTIK.core wie folgt überprüft werden:

1. mit einem Browser die Webadresse https://core.estatistik.de/core/ aufrufen und durch Eingabe von Kennung und Passwort anmelden. Kennung und Passwort stimmen mit denen überein, die auch für den Versand der Datenlieferung verwendet wurden.
2. über *Lieferungen anzeigen* kann die Lieferhistorie (Melderkonto) eingesehen werden. Es besteht die Möglichkeit sich zu jeder Lieferung das Protokoll anzeigen zu lassen. Das Protokoll enthält alle wichtigen Informationen zum Status der Datenlieferung.

HINWEIS:

Die gemeldeten Daten werden am gemeinsamen Dateneingang der amtlichen Statistik einer formalen Prüfung unterzogen. Ein Verstoß gegen formale Prüfungen (Fehlerschlüssel der Kategorie ERROR) kann zur Abweisung von einzelnen Meldungen oder ggf. der gesamten Datenlieferung führen.

8.3 Weitere Informationen

Weitere Informationen zu den DatML-Datenformaten, zu Liefervereinbarungen für ausgewählte Statistiken und Software sind über das Erhebungsportal https://erhebungsportal.estatistik.de/Erhebungsportal/# abrufbar.