

# Funktionsbeschreibung der DVDV-Bibliothek (Java)

Überblick über Funktionalität und Programmiermodell  
des Software-Development-Kits (SDK) zum  
Deutschen Verwaltungsdienstverzeichnis

Version 1.4.6

05. November 2009

## Änderungsverzeichnis

Änderung			Geänderte Kapitel	Beschreibung der Änderung	Autor
Nr.	Datum	Version			
1	14.09.2005	0.1		Initiale Dokumenterstellung	H. Krause
2	07.07.2006	1.1.0	alle	Überarbeitung für V. 1.1.0	H. Krause
3	01.11.2006	1.2.0		Überarbeitung für V. 1.2.0	O. Radomski
4	15.01.2007	1.2.1	2.3, 3.4, 3.5,	Überarbeitung für V. 1.2.1	H. Krause
5	31.10.2007	1.3.0	1, 3, Anhang	Überarbeitung für V. 1.3.0	H. Krause
6	28.04.2008	1.4.0	1	Überarbeitung für V. 1.4.0	H. Krause
7	07.07.2009	1.4.5	2.2., 3.2, Anhang	Überarbeitung für V 1.4.5	O. Radomski
8	05.11.2009	1.4.6	Anhang	Überarbeitung für V 1.4.6	H. Krause

## Prüfverzeichnis

Die folgende Tabelle zeigt einen Überblick über alle Prüfungen, die das vorliegende Dokument in den Zustand „fertig gestellt“ überführt haben.

Datum	Geprüfte Version	Anmerkungen	Prüfer
06.11.2007	1.3.0	Generelle Qualitätssicherung	O. Moehrke

# Inhaltsverzeichnis

<b>1</b>	<b>Überblick</b>	<b>3</b>
1.1	Bestandteile des SDK	4
1.2	Systemvoraussetzungen	4
<b>2</b>	<b>Anwendungsfälle</b>	<b>5</b>
2.1	find.servicedescription-Anfrage	5
2.2	find.authoritydescription-Anfrage	6
2.3	verify.category-Anfrage	8
<b>3</b>	<b>Programmierschnittstelle</b>	<b>10</b>
3.1	Überblick Paket-Struktur	10
3.2	DVDV-Nachrichtentypen	11
3.3	Elemente einer Dienstbeschreibung	15
3.4	Die Fassadenklasse DVDVManager	16
3.5	Fehler-Codes	20
	<b>WSDL-Dokument des Query-Services</b>	<b>21</b>
	WSDL mit OSCI-Binding	21
	WSDL mit http-POST-Binding	25
	<b>Änderungshistorie</b>	<b>27</b>
	<b>Referenzen</b>	<b>28</b>
	<b>Abbildungsverzeichnis</b>	<b>28</b>



## 1.1 Bestandteile des SDK

Die Distribution zum DVDV-SDK enthält drei Dateordner mit folgendem Inhalt:

- `<DVDV-SDK>/lib`  
Dieses Verzeichnis enthält die eigentliche DVDV-Bibliothek (Java-Archive) "dvdv-sdk-1.4.5.jar" sowie in dem Unterordner `<DVDV-SDK>/lib/ext` alle benötigten externen Java-Bibliotheken. Sämtliche Bibliotheken sind in den CLASSPATH entsprechend aufzunehmen.
- `<DVDV-SDK>/docs`  
Unterhalb des Verzeichnisses `docs/api` ist die Java-API-Dokumentation als HTML-Dateien zu finden (Start mit „index.html“). Ferner sind alle Lizenzen der verwendeten externen Software-Komponenten enthalten.
- `<DVDV-SDK>/samples`  
Hier ist ein einfaches Beispiel-Programm<sup>2</sup> abgelegt, das den Umgang mit dem DVDV-SDK demonstrieren soll. Die WSDL-Datei zum Query-Service und die Zertifikate sind für ein reales System anzupassen (s. Anhang WSDL-Datei des Query-Services).

## 1.2 Systemvoraussetzungen

Die DVDV-Bibliothek und alle benötigten externen Bibliotheken sind getestet gegen J2SE JRE/JDK 1.4.2, 1.5.0 und 1.6.0. Weitere Systemvoraussetzungen bestehen nicht.

---

<sup>2</sup> Um verschiedene Beispiele für unterschiedliche Schnittstellen ohne Verdoppelung von Quellcode erstellen zu können gibt es eine Klasse mit der allgemeinen Ablaufsteuerung und je eine abgeleitete Klasse mit konkreten Anpassungen an die OSCI-Schnittstelle und http-POST-Schnittstelle des DVDV.

## 2 Anwendungsfälle

Die DVDV-Bibliothek unterstützt die gegenwärtig definierten drei Anfragedienste des DVDV

- *find.servicedescription*
- *find.authoritydescription*
- *verify.category*

Alle Anfragedienste sind vom DVDV-Server implementiert als OSCI-Transport-Services des Kommunikationstyps *request/response*. Sowohl Anfrage als auch die Antwort werden transportiert in Form von XML-Dokumenten als Inhalt eines OSCI-Inhaltcontainers.

Da das DVDV ein öffentliches Verzeichnis ist, sind keine Maßnahmen zum Schutz der Vertraulichkeit erforderlich. Daher wird weder die Anfrage noch die Antwort verschlüsselt übertragen. Lediglich zur Absicherung des Dialoges kommt Verschlüsselung im Rahmen des Challenge/Response-Verfahrens von OSCI-Transport zum Einsatz.

Grundsätzlich kann jeder Anfragen an DVDV-Server stellen. Daher kann der DVDV-Server die anfragenden Klienten nicht kennen und somit keine Authentizitätsprüfungen vornehmen; Anfragen werden daher nicht elektronisch signiert. Aus Sicht des anfragenden Klienten stellt der DVDV-Server eine vertrauenswürdige Instanz dar. Um die Integrität und Authentizität des Antwortdokumentes zusichern zu können, werden die XML-Dokumente auf Ebene der Inhaltsdaten vom DVDV-Server signiert. Da die Antwort-Dokumente die Daten der Anfragen mit beinhalten, ist dem Klienten garantiert, dass er die zur Anfrage passende Antwort erhalten hat.

### 2.1 find.servicedescription-Anfrage

Die *find.servicedescription*-Anfrage des DVDV dient dazu, alle notwendigen Informationen zu ermitteln, die ein Klientensystem (*service consumer*) benötigt, um eine Dienstimplementierung einer Behörde nutzen zu können. Primär sind dies technische Verbindungsparameter wie Netzwerkadressen und Zertifikate der Infrastruktursysteme, aber ggf. auch Schemata zu Inhaltsdaten, Anforderungen ans Signaturniveau und weitere Informationen.

Gesucht werden die Informationen mittels zweier Parameter, die in einem XML-Dokument als Anfrage an den DVDV-Server gesendet werden:

- Angabe des gewünschten Dienstes ausgedrückt durch eine eindeutige URI
- ein Behördenschlüssel im jeweiligen fachlichen Kontext des Dienstes (z.B. der Amtliche Gemeindeschlüssel für XMeld-Rückmeldung).

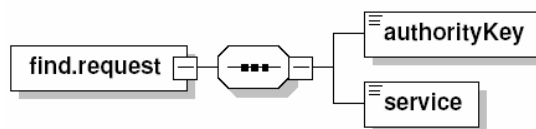


Abbildung 2: XML-Schema der *find.servicedescription*-Anfrage

Wird zur Anfrage eine passende Dienstimplementierung im DVDV gefunden, erhält der Klient als Antwort ein WSDL-Dokument, das sämtliche Informationen zum Dienst bereithält. WSDL ist eine XML-basierte Sprache zur präzisen Beschreibung von Softwareservices [WSDL]. Nähere Informationen zur Struktur der WSDL-Dokumente – insbesondere zur Unterstützung von OSCI-Transport basierten Diensten — sind unter [WSDLOSCI] enthalten.

Das WSDL-Dokument enthält ergänzend den Behördenschlüssel und die Angabe der Anfragezeit (Serverzeit des DVDV-Systems), um die Gültigkeit der signierten Antwort zusichern zu können.

## 2.2 find.authoritydescription-Anfrage

Mit Hilfe einer *find.authoritydescription*-Anfrage<sup>3</sup> kann ein Klientensystemen nähere Informationen zu einer Behörde erhalten. Diese Informationen umfassen allgemeine Daten zur Behörde und deren Stellvertretern wie Name, postalische Anschrift, Behördenschlüssel<sup>4</sup> und identifizierende Client-Zertifikate sowie eine Auflistung aller von der Behörde und deren Stellvertretern angebotenen Diensten. Bei der Auswertung der Liste ist der Gültigkeitszeitraum der Dienste zu beachten, da auch noch nicht oder nicht mehr gültige Dienste aufgelistet werden.

Gegenwärtig ist eine Anfrage implementiert, deren XML-Anfragedokument zwei Parameter beinhaltet:

- ein Behördenschlüssel im jeweiligen fachlichen Kontext des Dienstes (z.B. der Amtliche Gemeindeschlüssel für Xmeld-Rückmeldung)
- die exakte Behördenkategorie der gesuchten Behörde



Abbildung 3: XML-Schema der *find.authoritydescription*-Anfrage

Das signierte Antwortdokument enthält neben den Daten zur Behörde auch die beiden Anfrageparametern zur Zusicherung der Zusammengehörigkeit von Anfrage und Antwort.

<sup>3</sup> Die "FindAuthorityDescription"-Anfragen des SDK 1.4.5 werden nur von Slaves der Version 1.4.5 (oder neuer) beantwortet.

<sup>4</sup> Die Liste der Behördenschlüssel eines Stellvertreters enthält nur Schlüssel, die auch bei der Behörde selbst hinterlegt sind.

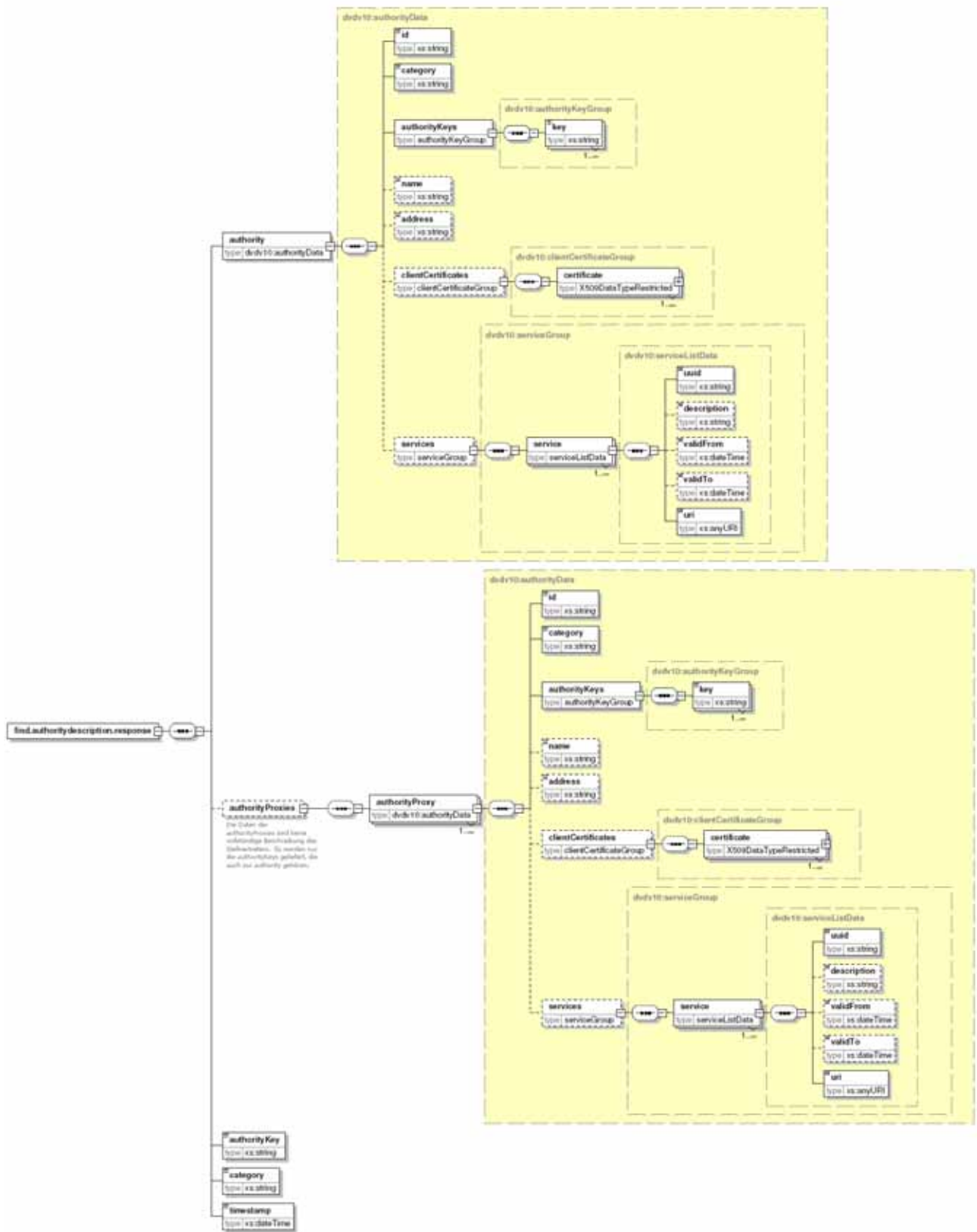


Abbildung 4: XML-Schema der `find.authoritydescription`-Antwort

## 2.3 verify.category-Anfrage

Die *verify.category*-Anfrage wird von Klientensystemen in der Rolle eines Diensteanbieters (*service supplier*) gestellt, d.h. von Behörden, die einen fachlichen Dienst implementiert und im DVDV publiziert haben. Der *verify.category*-Dienst ermöglicht dem Diensteanbieter, Nutzer ihres fachlichen Dienstes ggf. zum Zwecke einer Autorisierung zu überprüfen.

Zwar stellt das DVDV kein Autorisierungssystem dar, jedoch kann das DVDV Informationen zum Dienstanbieter liefern, die der Diensteanbieter für eine Autorisierung verwenden kann. Mit Hilfe des *verify.category*-Dienstes kann ein Diensteanbieter überprüfen lassen, ob das den Dienst nutzende Klientensystem einer bestimmten Behördenkategorie zugeordnet ist. Die Zuordnung erfolgt über ein X.509-Zertifikat, welches den Dienstanbieter identifiziert<sup>5</sup>.

Das DVDV macht keine Vorgaben, durch welches kryptographische Verfahren die Identifizierung eines Dienstanbieters zu erfolgen hat. Jedes Verfahren (i.a. als Teil eines Protokolls), das garantiert, dass der Dienstanbieter im Besitz des zum Zertifikat zugehörigen privaten Schlüssels ist, kann verwendet werden. Im Falle von OSCI-Transport-basierten fachlichen Dienstimplementierungen können die Signaturzertifikate eines Autors herangezogen werden. In anderen Umgebungen sind z.B. auch SSL-Client-Zertifikate denkbar. Welches Zertifikat für die Authentifizierung zu verwenden ist, legt die Spezifikation des jeweiligen fachlichen Dienstes fest<sup>6</sup>. Einer Behörde in der Rolle des Dienstanbieters können im DVDV mehrere authentifizierende X.509-Zertifikate zugeordnet werden.

Als Parameter im XML-Anfragedokument werden die beiden Parameter an das DVDV übermittelt:

- identifizierende Zertifikat (in Form einer Restriktion des `X509DataType` der DigSig-Spezifikation)
- zu prüfende Behördenkategorie

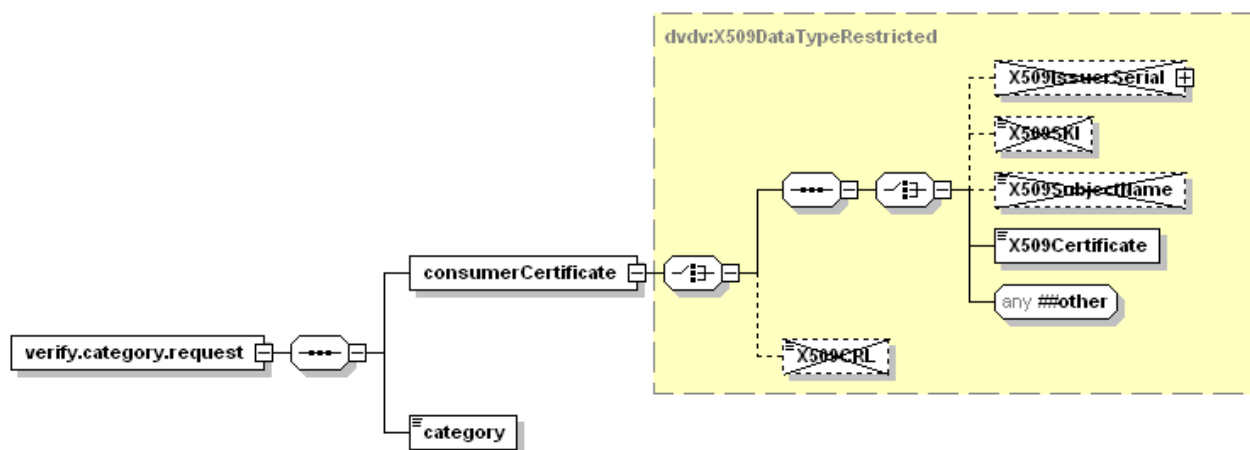


Abbildung 5: XML-Schema der *verify.category*-Anfrage

Das Antwortdokument enthält — neben den beiden Anfrageparametern zur Zusicherung der Zusammengehörigkeit von Anfrage und Antwort — einen booleschen Wert, der das Ergebnis der Prüfung enthält. Der Wert *true* gibt an, dass im DVDV mindestens eine Behörde der übermittelten Behördenkategorie registriert ist, der das übermittelte Zertifikat zugeordnet ist.

<sup>5</sup> Die Behörden-identifizierenden X.509-Zertifikate werden mittels DVDV-Pflegeclient als so genannte „Client-Zertifikate“ den Behörden zugeordnet.

<sup>6</sup> Für Dienste von OSCI-XMeld ist das Signaturzertifikat des Inhaltscontainers `XMELD_DATA`, welches das XMeld-Dokument enthält, zum Zwecke der Authentifizierung des Dienstanbieters zu verwenden. Mit der OSCI-Bibliothek des KoopA ist dies z.B. aus der `ResponseToFetchDelivery`-Nachricht zu erhalten mit: `contentContainer.getSigners().getSignatureCertificate()`

Bei hierarchischen Kategorien mit mehreren Stufen der Spezialisierung (z.B. „Oberverwaltungsgericht, Verwaltungsgericht, Gericht“) wird eine Übermittlung der allgemeineren Kategorie auch dann mit *true* beantwortet, wenn das Zertifikat einer Behörde einer spezielleren Kategorie zugeordnet ist (ein Verwaltungsgericht *ist ein* Gericht).

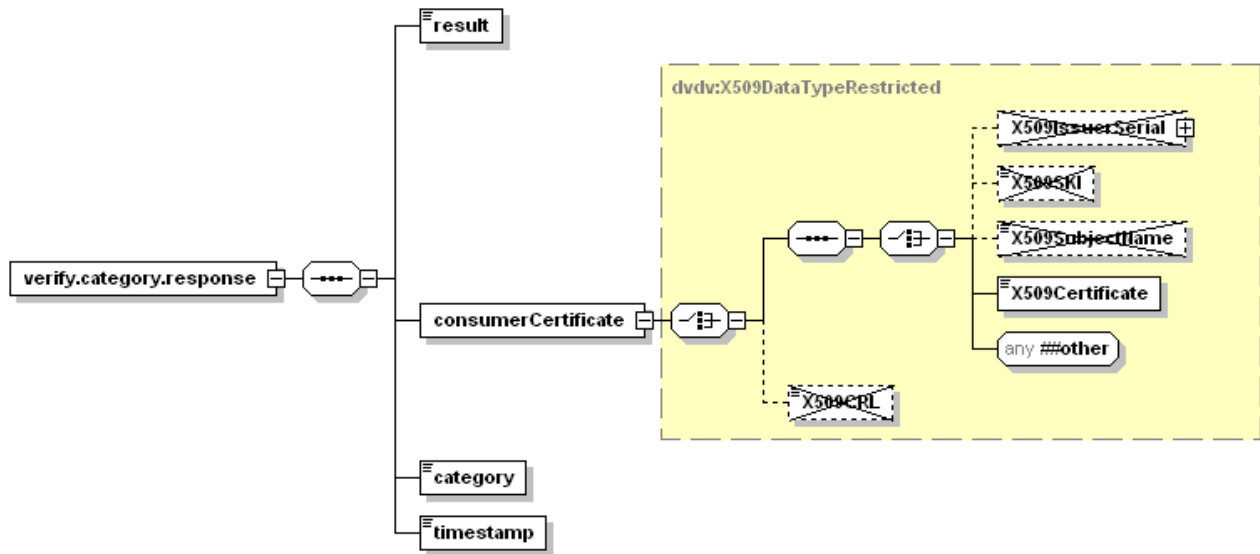


Abbildung 6: XML-Schema der `verify.category`-Antwort

## 3 Programmierschnittstelle

Im folgenden Abschnitt wird die Struktur der DVDV-Bibliothek, die darin enthaltenen Klassen und Interfaces sowie das zugrunde gelegte Programmiermodell beschrieben. Die Erläuterungen zu Klassen und Interfaces erfolgt jedoch nur in groben Zügen; für eine detailliertere Beschreibung sei auf die Java-API-Doc als Bestandteil des SDK verwiesen.

### 3.1 Überblick Paket-Struktur

Die Klassen und Interfaces der DVDV-Bibliothek gliedern sich vornehmlich in drei Blöcke mit differenzierten Aufgaben und Verantwortlichkeiten. Die Gliederung spiegelt sich wieder in Java-Packages und Subpackages (siehe Abbildung 7).

Das Paket `de.dvdv.manager` bzw. Subpaket enthält die zentrale Fassadenklasse der Bibliothek, die Methoden zur Abwicklung der Anfragen bereithält. Die Fassadenklasse stützt sich auf Interfaces im Paket `de.dvdv.object` und den Subpaketen, die als Objekt-Repräsentanten für Nachrichten und Teilnachrichten dienen (*value objects*). Das Paket `de.dvdv.cache` enthält ein Interface zur Festlegung der Caching-Funktionalität, auf die sich die Bibliothek stützen kann.

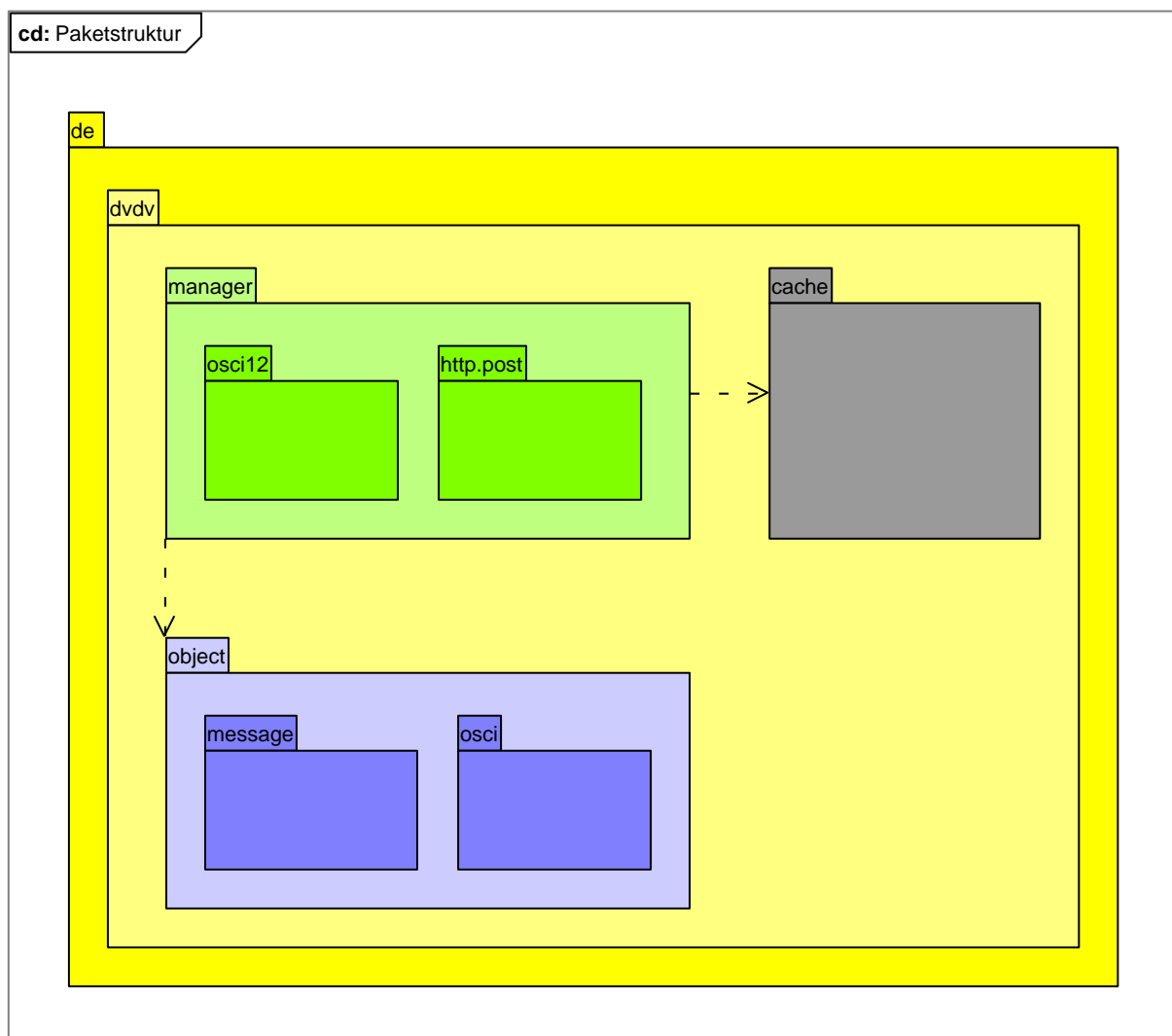


Abbildung 7: Paket-Struktur der DVDV-Bibliothek

## 3.2 DVDV-Nachrichtentypen

Das Paket `de.dvdv.object.message` enthält Objekt-Repräsentanten der XML-Anfrage- und Antwortnachrichten, die zwischen Klientensystem und DVDV-Server ausgetauscht werden. Die Repräsentanten sind alle als Interfaces ausgelegt; sie werden nicht direkt durch den Anwendungsentwickler instanziiert. Request-Objekte werden durch Factory-Methoden der Klasse `DVDVManager` konstruiert. Response-Objekte werden indirekt durch die Methoden `processRequest` und `processRequests` der Klasse `DVDVManager` erzeugt.

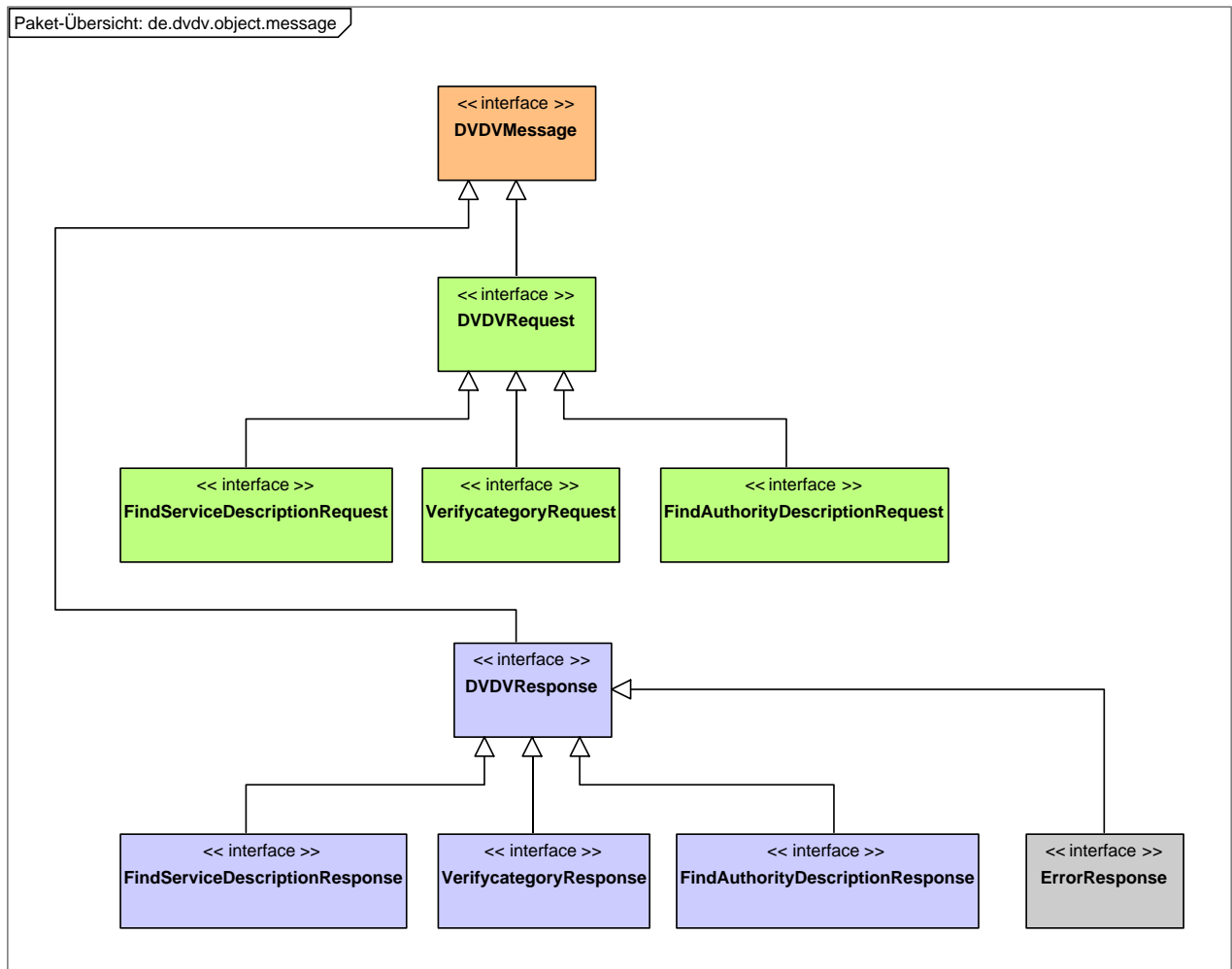


Abbildung 8: Interfaces als Repräsentanten der DVDV-Anfrage- und Antwortnachrichten

Im Folgenden werden die Nachrichten bzw. ihr Zweck kurz erläutert.

- **FindServiceDescriptionRequest**

Dieses Interface definiert den Objekt-Repräsentanten einer DVDV-Anfrage, die zum Auffinden einer angebotenen Dienstbeschreibung einer Behörde benutzt wird. Die Beschreibung eines Dienstes in DVDV erfolgt mittels WSDL. Für jeden angebotenen Dienst einer Behörde existiert eine entsprechende WSDL-Vorlage, die zum Zeitpunkt der Anfrage mit den aktuellen Beschreibungsinhalten des Dienstes gefüllt wird und als Inhalt einer DVDV-Antwort `FindServiceDescriptionResponse` zurückgeliefert wird. Zum Auffinden einer Dienstbeschreibung einer Behörde ist daher sowohl der Behördenschlüssel, als auch die URI (der `targetNamespace`) der WSDL-Vorlage anzugeben, die den gewünschten fachlichen Dienst beschreibt.

Eine Instanz dieser Nachricht kann mit der Methode `createFindServiceDescriptionRequest` des `DVDVManager` erzeugt werden.

- **FindServiceDescriptionResponse**

Dieses Interface definiert den Objekt-Repräsentanten einer DVDV-Antwort, welcher die Dienstbeschreibung enthält, die durch die zuvor gestellte DVDV-Anfrage `FindServiceDescriptionRequest` für eine Behörde angefordert wurde. Die Beschreibung des Dienstes liegt in der Antwort als WSDL-String vor. Somit besteht die Möglichkeit, die Beschreibung nach unterschiedlichen Kriterien und mit vorhandenen XML-Parsern zu analysieren. Die DVDV-Antwort `FindServiceDescriptionResponse` bietet aber außerdem die Möglichkeit, auf die wesentlichen Dienstbeschreibungselemente (`ServiceElement`) über deren Schlüsselnamen zuzugreifen.

Wenn zum Beispiel folgende Dienstbeschreibung in einer Antwort `FindServiceDescriptionResponse` vorliegt,

```
<defintions>
...
<service name="osciRueckmeldungService">
  <!--OSCI Infratstrukturserver -->
  <osci:devices>
    <osci:intermediary uri="http://192.10.83.12:8080/intermed"
                      name="TESTAIntermediär">
      <osci:signatureCertificate>
        ...
      </osci:signatureCertificate>
    </osci:intermediary>
  </osci:devices>
</service>
</defintions>
```

dann kann auf das Dienstbeschreibungselement `OsciIntermediaryElement` mit dem Schlüsselnamen „*TESTAIntermediär*“ wie folgt zugegriffen werden:

```
...
FindServiceDescriptionResponse response =
    manager.processRequest(findRequest);
OsciIntermediaryElement element =
    response.getOsciIntermediaryElement("TESTAIntermediär");
...
```

Näheres zu der Klasse `ServiceElement` und dessen Subklassen ist im Abschnitt 3.3 erläutert.

- **FindAuthorityDescriptionRequest**

Der Objekt-Repräsentant einer DVDV-Anfrage nach Informationen zu einer Behörde wird durch dieses Interface definiert. Zur Referenzierung der gesuchten Behörde werden die Suchparameter Behördenschlüssel (nicht ID) und Behördenkategorie übermittelt. Eine Instanz dieses Nachrichten-Objekts kann mit Hilfe der Factory-Methode `createFindAuthorityDescriptionRequest` des `DVDVManager` erzeugt werden.

- **FindAuthorityDescriptionResponse**

Die Antwortnachricht mit den Informationen zu einer Behörde wird durch dieses Interface repräsentiert. Die Daten sind umfangreicher und in Informationsbestandteile strukturiert. Analog zu den anderen Antworten enthält dieses Interface die Anfrageparameter sowie den Anfragezeitpunkt. Der Zugriff auf die Informationen zur Behörde und Stellvertretern erfolgt über die Methode `getAuthority`, die eine Instanzen zum Interface `Authority`<sup>7</sup> liefert. Details sind der Java-API-Doc zu entnehmen, die Bestandteil des SDK ist.

- **VerifyCategoryRequest**

Dieses Interface definiert den Objekt-Repräsentanten einer DVDV-Anfrage, mit dem festgestellt werden kann, ob eine Dienst-nutzende Behörde einer bestimmten Behördenkategorie (wie z.B. Meldebehörde oder Gericht) zugeordnet ist. Die Zuordnung wird durch ein identifizierendes X.509-Zertifikat („Client-Zertifikate“ der Behörde) geprüft. Eine Instanz dieser Nachricht kann mit der Methode `createVerifyCategoryRequest` des `DVDVManager` erzeugt werden.

- **VerifyCategoryResponse**

Dieses Interface definiert den Objekt-Repräsentanten einer DVDV-Antwort, die das Ergebnis der Verifikationsanfrage enthält. Neben den Anfrageparametern und dem Anfragezeitpunkt enthält es das boolesche Ergebnis. Beispiel:

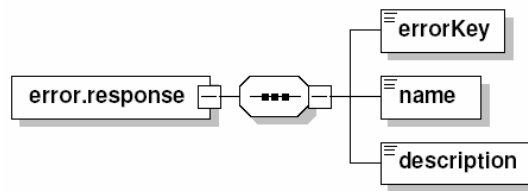
```
...
X509Certificate cert = ...;
VerifyCategoryRequest verifyRequest =
    manager.createVerifyCategoryRequest(cert, "Meldebehörde");
VerifyCategoryResponse verifyResponse =
    manager.processRequest(verifyRequest);
if (verifyResponse.getResult() == false)
    throw new Exception();
...
```

---

<sup>7</sup> Zur Umsetzung des CR "Erweiterung der FindAuthorityDescriptionResponse" ist das Interface `Authority` erweitert worden. Aus Kompatibilitätsgründen liefert die Implementierung von `getServices()` weiterhin alle Dienste (die der Behörde und die der Stellvertreter) und die von `getClientCertificates()` weiterhin nur die Zertifikate der Behörde selbst.

- **ErrorResponse**

Dieses Interface definiert den Objekt-Repräsentanten einer DVDV-Antwort, der einen Fehler in der Bearbeitung einer zuvor gestellten DVDV-Anfrage beschreibt. Ein Fehler wird durch einen Fehlercode, einen Fehlernamen und eine Fehlerbeschreibung spezifiziert. Fehlermeldungen des DVDV sind gemäß folgendem XSD-Schema aufgebaut:



**Abbildung 9: XML-Schema einer DVDV-Fehlerantwortnachricht**

Fehler, die während der Bearbeitung von Anfragen auf dem DVDV-Server auftreten, werden dem aufrufenden Programm über Objekte dieses Typs mitgeteilt und nicht über den sonst üblichen Exception-Mechanismus. Der Grund ist, dass zur Steigerung des Durchsatzes Anfragen (auch unterschiedlichen Typs) wahlweise auch im Stapel-Modus an den DVDV-Server übermittelt werden können, so dass korrekt und fehlerhaft bearbeitete Anfragen innerhalb eines Dialogs kombiniert auftreten können.

In der Verantwortung des Anwendungsprogramms liegt es, erhaltene Antwort-Objekte (im Stapel- oder Single-Modus) hinsichtlich ihres Typs zu überprüfen (mit dem *instanceof*-Operator), um auf Fehler in Bearbeitung reagieren zu können.

### 3.3 Elemente einer Dienstbeschreibung

Die vom DVDV-Server gelieferte Antwort auf eine *find*-Anfrage ist – sofern eine passende Dienstimplementierung gefunden wurde – in Form eines WSDL-Dokumentes kodiert. Es steht dem Anwendungsentwickler frei, das Dokument mittels XML-Parser selbst auszuwerten.

In der Mehrzahl der Fälle wird das Anwendungsprogramm bzw. Fachverfahren jedoch nur an den technischen Elementen („Verbindungsparametern“) interessiert sein, die individuell für jede Dienstimplementierung ausgeprägt sind. Die allgemeinen und für jede Dienstimplementierung identischen Informationen zum Dienst werden dem Anwendungsprogramm i.d.R. bekannt sein. Ein direkter Zugriff auf diese individuellen Elemente wird durch die DVDV-Bibliothek unterstützt. Mit Hilfe von Schlüsselnamen, die der WSDL-Vorlage zum Dienst jeweils entnommen werden können, kann auf Objekt-Repräsentanten für die sechs verschiedenen Typen von Dienstanbieter abhängigen Elementen zugegriffen werden.

Für diese sechs Objekt-Repräsentanten sind jeweils Interfaces definiert, die *getter*-Methoden auf die relevanten Daten bereitstellen. Sie leiten sich alle vom Interface *ServiceElement* ab.

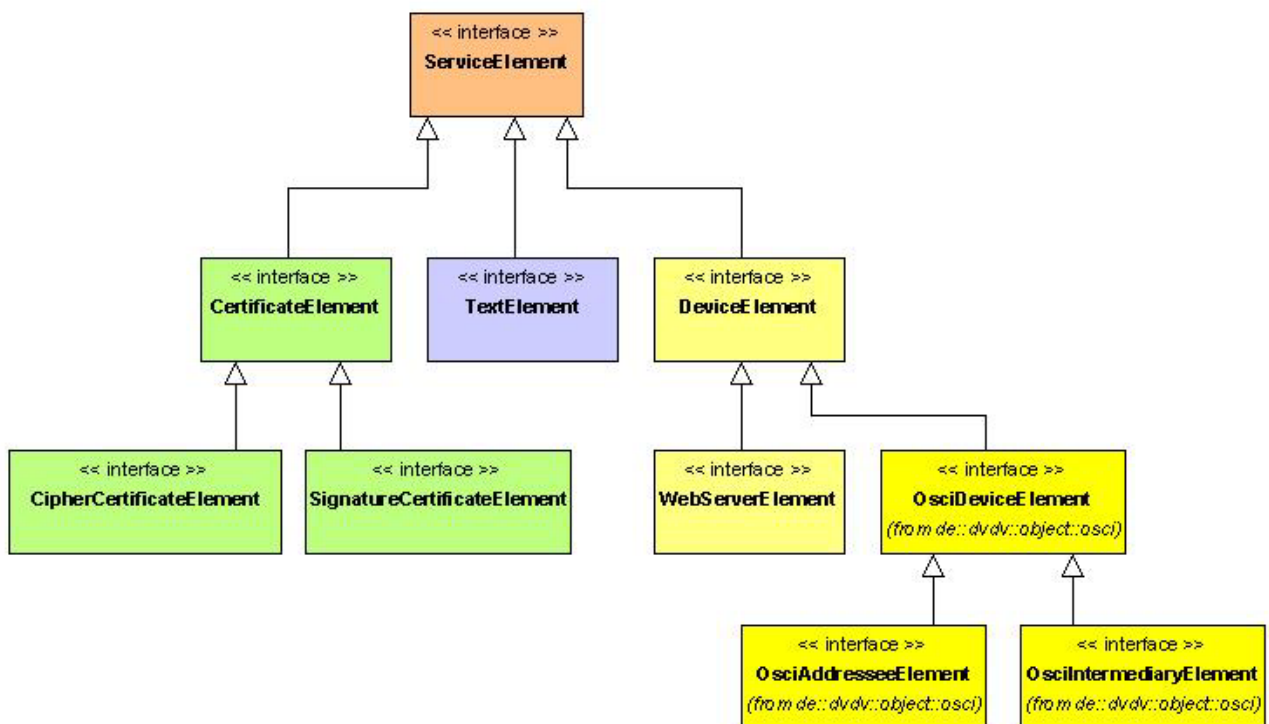


Abbildung 10: Interfaces der Dienstbeschreibungselemente

Die Objekte vom Typ *OsciIntermediaryElement* tragen die Informationen zu OSCI-Intermediären (Signatur- und Verschlüsselungszertifikat und URI des Netzwerkknotens). Ein Anwendungsprogramm, das zur Nutzung eines fachlichen Dienstes eine OSCI-Transport-Nachricht z.B. mit der OSCI-Bibliothek konstruieren möchte, benötigt diese Informationen, um *Intermediary*-Rollenobjekte der OSCI-Bibliothek instanzieren zu können.

Objekte vom Typ `OsciAddresseeElement` tragen die Informationen zu OSCI-Empfängern (Verschlüsselungszertifikat und optional die URI des Netzwerkknotens<sup>8</sup>). Das Zertifikat wird benötigt, um Addressee-Rollenobjekte der OSCI-Bibliothek instanzieren zu können.

Objekte des Typs `CipherCertificateElement` repräsentieren Verschlüsselungszertifikate von Lesern. Das Zertifikat wird benötigt, um Nachrichten korrekt für den Leser verschlüsseln zu können.

Objekte des Typs `SignatureCertificateElement` repräsentieren Signaturzertifikate, die ein Anwendungsprogramm benötigt, um die Authentizität von signierten Antworten eines fachlichen Dienstes verifizieren zu können.

Das Interface `TextElement` definiert Objekte, die Textkonstanten repräsentieren. Textkonstanten können z.B. für den Betreff einer OSCI-Transport-Nachricht oder dem `soapAction`-Attribut eines SOAP-Requests benötigt werden.

Objekte vom Typ `WebServer` repräsentieren beliebige Web- bzw. SOAP-Server und beinhalten gegenwärtig lediglich die URL (Netzwerkadresse) des Servers.

### 3.4 Die Fassadenklasse DVDVManager

Die abstrakte Klasse `DVDVManager` bzw. deren direkte Subklassen `Osci12DVDVManager` und `HttpPostDVDVManager` stellen eine Fassade zu Implementierungen dar, welche die Konstruktion und das Versenden der Anfragenachrichten, dem Empfang der Antworten sowie die Auswertungen der Antwortnachrichten und des Übermittlungsstatus realisieren.

Die Klasse `Osci12DVDVManager` ist die Implementierung eines `DVDVManager`, der für die Verarbeitung von DVDV-Nachrichten mit einem DVDV-Server über das OSCI Transportprotokoll V.1.2 kommuniziert – der gegenwärtig einzigen zulässigen Kommunikationsschnittstelle über externe Netze. Die Klasse `HttpPostDVDVManager` ist die Implementierung zur Performance-optimierten Kommunikation über http, die angewendet werden kann, wenn DVDV-Slave und abfragendes System innerhalb einer Sicherheitszone eines Rechenzentrums kommunizieren.

Die Fassaden `Osci12DVDVManager` und `HttpPostDVDVManager` sind daher die zentralen Schnittstellenklassen für den Anwendungsentwickler.

#### Konstruktion und Initialisierung Osci12DVDVManager

Für die Kommunikation via OSCI-Transport ist dem Manager der OSCI-Sender bekanntzumachen. Der Challenge/Response-Mechanismus des Protokolls erfordert es, dass der Manager Zugriff auf den privaten Schlüssel zum Zwecke der Entschlüsselung erhält. Drei alternative Konstruktoren werden angeboten, um den Zugriff zu initialisieren:

- `Osci12DVDVManager(de.osci.osci12.extinterfaces.crypto.Decrypter decrypter)` mit einer Instanz der Klasse `Decrypter` der OSCI-Bibliothek, die den Zugriff auf den privaten Schlüssel kapselt.
- `Osci12DVDVManager(java.security.KeyStore store, String pin)` mit `KeyStore` des `java.security`-Pakets samt PIN, aus dem das erste Zertifikat ausgelesen wird.
- `Osci12DVDVManager(String fileName, String pin)` mit Namen einer Datei, in der ein privater Schlüssel im PKCS#12-Format abgelegt ist (typischerweise `.p12`- oder `.pfx`-Datei).

---

<sup>8</sup> Um Abwärtskompatibilität mit der Version 1.1.0 zu gewährleisten, enthält das Interface „`OsciAddresseeElement`“ weiterhin eine Methode zum Abfragen eines Signaturzertifikates. Diese Methode liefert jedoch immer „null“ und sollte nicht mehr verwendet werden (*deprecated*).

Eine notwendige Initialisierung des Managers ist die Angabe der Verbindungsparameter zu einem oder mehreren DVDV-Servern. Zu diesem Zweck nutzt die DVDV-Bibliothek selbst WSDL-Dienstbeschreibungen (DVDV-Query-Service), die von den Betreibern der DVDV-Server verteilt werden müssen. Da DVDV-Server für anfragende Klientensysteme als vertrauenswürdige Instanzen betrachtet werden, stellen diese WSDL-Dokumente den „boot-strap“ einer transitiven Vertrauensketten dar.

Zur Erhöhung der Verfügbarkeit können dem Manager mehrere WSDL-Dokumente zu alternativen DVDV-Servern bekannt gemacht werden. Falls ein DVDV-Server nicht erreichbar ist, wird durch den Manager ein Versuch mit dem nächsten der Liste vorgenommen. Die Angabe der zu verwendenden DVDV-Server erfolgt durch:

```
...
String[] dvdvQueryWsdl = { „DVDVQuery1.wsdl“, „DVDVQuery2.wsdl“ };
manager.loadDvdvServerParameters(dvdvQueryWsdl);
...
```

Gegebenenfalls kann die Nichtverfügbarkeit von DVDV-Servern erst nach relevanten Wartezeiten wahrgenommen werden (z.B. Timeouts bei Netzzugriffen). Würde jede Anfrage grundsätzlich gegen die Liste der Server gestellt, könnte der Durchsatz erheblich eingeschränkt werden.

Aus diesem Grund werden – sofern der primäre DVDV-Server (repräsentiert durch das erste WSDL-Query-Service-Dokument) nicht verfügbar ist – neue Anfragen nicht unmittelbar wieder gegen diesen versucht. Stattdessen werden für einen konfigurierbaren Zeitraum alle weiteren Anfragen gegen den letzten erfolgreichen Server gestellt. Erst nach Verstreichen dieser Zeitspanne wird wieder ein erneuter Versuch mit dem ersten Server der Liste unternommen.

Der Standardwert für diesen Zeitraum beträgt 10 Minuten. Der Wert lässt sich ändern durch eine *setter*-Methode des Managers:

```
...
manager.setRetryInterval(3600); // neue Zeitspanne in Sekunden
...
```

### **Konstruktion und Initialisierung HttpPostDVDVManager**

Soll direkt über http anstelle von OSC-Transport mit dem DVDV-Slave kommuniziert werden (innerhalb eines Rechenzentrum-Netzwerks), ist die Fassadenklasse `HttpPostDVDVManager` zu instanzieren. Ein Verschlüsselungszertifikat bzw. ein privater Schlüssel ist hierzu nicht erforderlich, demzufolge existiert nur der Default-Konstruktor ohne Argumente.

Die Initialisierung des Objektes mit den Verbindungsparametern des DVDV-Slaves erfolgt analog wie oben beschrieben: Mit der Interface-Methode `manager.loadDvdvServerParameters()` werden eine oder mehrere WSDL-Dokumente des Slave-Betreibers ausgelesen. Allerdings muss es sich um WSDL-Dokumente mit http-POST-Binding statt OSC-Binding handeln (siehe Anhang: WSDL mit http-POST-Binding).

## Erzeugung der Anfrage-Nachrichten

Die Anfragenachrichten *find.servicedescription*, *find.authoritydescription* und *verify.category* werden repräsentiert durch Objekte vom Typ `FindServiceDescriptionRequest`, `FindAuthorityDescriptionRequest` und `VerifyCategoryRequest`. Die Manager-Fassade bietet Factory-Methoden, um die Objekte zu erzeugen. Beispiel:

```
FindServiceDescriptionRequest findServiceRequest =
    manager.createFindServiceDescriptionRequest(authorityKey, wsdlURI);

FindAuthorityDescriptionRequest findAuthorityRequest =
    manager.createFindAuthorityDescriptionRequest(authorityKey, "Meldebehörde");

VerifyCategoryRequest verifyRequest =
    manager.createVerifyCategoryRequest(certificate, "Meldebehörde");
```

Um eine Stapelverarbeitung durchzuführen, sind die so erzeugten Anfragenachrichten in einer Liste zusammen zu führen. Auch Anfragen unterschiedlichen Typs dürfen in einer Stapelverarbeitung zusammengefasst werden. Beispiel:

```
ArrayList requests = new ArrayList();
requests.add(findServiceRequest);
requests.add(verifyCategoryRequest);
```

## Durchführung der Anfrage

Der Manager erlaubt sowohl Einzelanfragen, als auch die Verarbeitung mehrerer Anfragen im Stapel (Batch). Die Stapelverarbeitung bewirkt verbesserte Gesamtantwortzeiten, da nur eine einzige Kommunikation mit dem DVDV-Server zur Verarbeitung aller Anfragen notwendig ist und somit der Zusatzaufwand bei Kommunikation und Kryptographie drastisch reduziert wird.

Die Durchführung der Stapelverarbeitung erfolgt mit der Methode `processRequests()`, der die Liste mit den Anfrageobjekten als Argument übergeben wird:

```
BatchResult batchResult = manager.processRequests(requests);
```

Als Rückgabe erhält man ein Objekt vom Typ `BatchResult` als `Collection` über die Antwortnachrichten. Die Einzelantworten können wahlweise über den Index der Anfrageobjekte oder über das Anfrageobjekt selbst aus der `Collection` entnommen werden:

```
...
DVDVResponse findResponse = batchResult.get(findRequest);
...
DVDVResponse verifyResponse = batchResult.get(1);
...

```

Einzelanfragen erfolgen mit der Methode `processRequest()`, der ein einzelnes Anfrage-Objekt als Argument übergeben wird. Die Methode gibt die Antwort direkt in Form eines Objektes vom Typ `DVDVResponse` zurück.

Die Antwortobjekte bieten *getter*-Methoden auf alle relevanten Daten der Antwort (siehe dazu Java-API-Doc).

## Auswertung der Antwortnachrichten

Antworten können entweder vom Typ `FindServiceDescriptionResponse`, `FindAuthorityDescriptionResponse` und `VerifyResponse` (je nach Anfrageobjekt) oder `ErrorResponse` sein. Das Anwendungsprogramm hat vor dem Zugriff auf Objektmethoden mit dem `instanceof`-Operator zu prüfen, um welche Ausprägung einer Antwort es sich bei einer konkreten Instanz handelt.

Fehlerantworten in Form von `ErrorResponse`-Objekten treten auf, wenn die Verarbeitung einer (Einzel-)Anfrage nicht erfolgreich war. Gründe können fehlerhafte Anfrageargumente sein (keine Treffer bei *find*- oder fehlerhaftes Zertifikat bei *verify*-Anfragen) oder sonstige Fehler in der Verarbeitung der Anfrage auf dem DVDV-Server.

## Auswertung des Übermittlungsstatus

Jedem Anfrageprozess (eine Stapelverarbeitung oder eine Einzelanfrage) wird nach Abarbeitung ein Status zugeordnet, der den Erfolg des Übermittlungsprozesses kennzeichnet. Hierzu zählen insbesondere die Prüfung des OSCI-Feedbacks und der Signatur der Antwort (sofern mittels OSCI-Transport mit dem DVDV-Slave kommuniziert wird). Eine grobe Klassifizierung des Status („Ampel“) liefert die Methode `getResponseState()`. Die möglichen Status-Werte sind:

- `OK` - alle Prüfungen erfolgreich
- `WARNING` - die Prüfung enthält Warnungen bzw. nicht alle Prüfungen konnten durchgeführt werden.
- `ERROR` - Fehler bei der Prüfung
- `CACHED` - keine Aussage möglich, da Antwort aus lokalem Cache
- `INITIALIZED` - Anfrageprozess noch nicht abgeschlossen

Detailliertere Auswertungen des OSCI-Status sind möglich, indem direkt auf das `ResponseToMediateDelivery`-Objekt der OSCI-Bibliothek zugegriffen wird. Der Zugriff erfolgt mit der Methode `getOsciResponse()`.

### 3.5 Fehler-Codes

In der folgenden Tabelle sind die Fehlernachrichten mit Code und Bezeichnung aufgeführt, die ein DVDV-Queryserver bei einer fehlerhaften Bearbeitung der drei Anfragetypen versenden kann.

Code	Fehlername	Beschreibung
5003	LDAP missing serviceelement error	Elemente (Intermediär, Empfänger, Zertifikate etc.) einer Dienstimplementierung sind aufgrund eines fehlerhaften Datenbestands nicht vorhanden <sup>9</sup> .
5004	LDAP no service hits error	Keine gültige Dienstimplementierung zur <i>find.servicedescription</i> -Anfrage gefunden (Hinweis: evtl vorhandener Dienst könnte ggf. nicht mehr / noch nicht gültig sein).
5007	LDAP invalid certificate error	Das in einer <i>verify.category</i> -Anfrage übermittelte Zertifikat ist syntaktisch ungültig.
5009	LDAP no authority hits error	Bei einer <i>find.authoritydescription</i> -Anfrage ist keine zu den Anfrageparametern passende Behörde gefunden worden.
5018	LDAP ambiguous authority key error	Die <i>find.servicedescription</i> -Anfrage nach einer Dienstbeschreibung findet mehr als einen Treffer. Mindestens zwei Behörden ist der gleiche Behördenschlüssel zugeordnet und diese haben denselben Dienst publiziert. Dieser Fehler kann nur auftreten, wenn der Datenbestand fachlich inkorrekt ist <sup>10</sup> .
9004	DVDV decryption error	Eine Anfrage kann nicht entschlüsselt werden. Die Situation tritt ein, wenn das in der WSDL zum DVDV-Queryservice hinterlegte Verschlüsselungszertifikat für den OSCI-Empfänger nicht das des DVDV-Queryservers ist <sup>11</sup> .
9005	DVDV batch limit exceeded error	Der Provider eines DVDV-Servers (Replikationsslave) kann Obergrenzen für die Größe von Stapelanfragen festlegen, um sich gegen übermäßige Ressourcenbeanspruchung durch einzelne Klienten zu schützen.
9999	DVDV internal error	Ein interner Fehler im DVDV-Serversystem ist aufgetreten.

<sup>9</sup> Der DVDV-Pflegeclient lässt die Publikation von Dienstbeschreibungen zu Behörde nur zu, wenn der Dienst vollständig konfiguriert wurde und alle Dienstelemente (Intermediäre, Empfänger, Zertifikate etc.) zum Zeitpunkt auch vorhanden sind. Die Fehlersituation kann jedoch eintreten, wenn ein Provider Dienstelemente, auf die eine Dienstbeschreibung verweist, ohne Prüfung aus dem Verzeichnis entfernt.

<sup>10</sup> Der DVDV-Pflegeclient prüft bei Datenänderungen, welche Behördenschlüssel oder Dienstbeschreibungen betreffen, ob die fachliche Fehlersituation einträte und weist ggf. den Änderungsauftrag ab. Die Situation sollte daher nicht eintreten können.

<sup>11</sup> Sofern das in der WSDL zum DVDV-Queryservice hinterlegte Verschlüsselungszertifikat des OSCI-Intermediärs inkorrekt ist, wird eine `DVDVOSCIException` geworfen, die über den OSCI-Fehler: 9100 „Empfangene Nachricht stellt keine gültige OSCI-Nachricht dar“ informiert.

## WSDL-Dokument des Query-Services

In diesem Anhang werden die nötigen Anpassungen einer vorhandenen WSDL-Datei an ein reales System beschrieben.

Das SDK wertet nur die Verbindungsparameter der WSDL-Datei des Query-Services aus. Die übrige Struktur hat nur informativen Charakter.

### WSDL mit OSCI-Binding

Zu den sicherheitskritischen Verbindungsparametern für eine Kommunikation über OSCI-Transport 1.2 mit dem DVDV-Slave gehören:

- URI des verwendeten Intermediärs
- Verschlüsselungszertifikat des verwendeten Intermediärs
- Signaturzertifikat des verwendeten Intermediärs
- URI des verwendeten DVDV-Applikationsservers
- Transport-Verschlüsselungszertifikat des verwendeten DVDV-Applikationsservers
- Inhaltsdaten-Signaturzertifikat des verwendeten DVDV-Applikationsservers

Beim Anpassen der WSDL-Datei ist die Kodierung („UTF-8“) zu beachten<sup>12</sup>. Ansonsten können die URIs von Intermediär und Applikationsserver einfach ersetzt werden. Zum Anpassen der Zertifikate müssen diese Base64-kodiert vorliegen. Diese Zertifikate können dann einfach mit einem Text-Editor geöffnet werden, um die Base64-kodierte Zeichenfolge<sup>13</sup> auszuschneiden und in die WSDL-Datei zu übernehmen.

Zusätzlich sollte der „`dvdv:supplier`“ Abschnitt der WSDL-Datei angepasst werden<sup>14</sup>, um einen schnellen Überblick zu ermöglichen.

In Abbildung 11 sind die anzupassenden Stellen in einer Beispiel WSDL-Datei gelb hinterlegt.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Service -->
<definitions name="dvdvQuery"
targetNamespace="http://www.dvdv.de/2006/09/dvdvQuery.wsdl"
xmlns:tns="http://www.dvdv.de/2006/09/dvdvQuery.wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:osci="http://www.osci.de/2006/07/wsdl/"
xmlns:dvdv="http://www.dvdv.de/messages/1.0/query"
xmlns:error="http://www.dvdv.de/messages/1.0"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <import namespace="http://www.osci.de/2006/07/wsdl" location="WSDL-OSCI-Binding_1.2.xsd"/>
```

<sup>12</sup> Hierzu ist ein UTF-8-fähiger Editor zu verwenden (z.B. XML-Spy). Sofern der Editor eine Markierung der Bytereihenfolge (Byte Order Mark, BOM) zu Beginn der Datei schreibt, sollte diese entfernt werden.

<sup>13</sup> Die Zeichenfolgen "-----BEGIN CERTIFICATE-----" und "-----END CERTIFICATE-----", die das Base64-kodierte Zertifikat eventuell umschließen, sind nicht in die WSDL zu übernehmen.

<sup>14</sup> Dem `AuthorityKey` sollte immer der Präfix „`dvdv:`“ vorangestellt werden.



```
<!--findServiceDescription Operation -->
<operation name="findServiceDescription">
  <osci:operation communicationType="request-response-noprocol"
    signatureLevel="none" encrypted="false"/>
  <input>
    <osci:container>
      <osci:part>
        <osci:content part="tns:findServiceRequest"/>
      </osci:part>
    </osci:container>
  </input>
  <output>
    <osci:container signatureLevel="advanced" encrypted="false">
      <osci:authorRef ref="DVDVAuthor"/>
      <osci:part>
        <osci:content part="tns:findServiceResponse"/>
      </osci:part>
    </osci:container>
  </output>
</operation>

<!--findAuthorityDescription Operation -->
<operation name="findAuthorityDescription">
  <osci:operation communicationType="request-response-noprocol"
    signatureLevel="none" encrypted="false"/>
  <input>
    <osci:container>
      <osci:part>
        <osci:content part="tns:findAuthorityRequest"/>
      </osci:part>
    </osci:container>
  </input>
  <output>
    <osci:container signatureLevel="advanced" encrypted="false">
      <osci:authorRef ref="DVDVAuthor"/>
      <osci:part>
        <osci:content part="tns:findAuthorityResponse"/>
      </osci:part>
    </osci:container>
  </output>
</operation>

<!--verifyCategory Operation -->
<operation name="verifyCategory">
  <osci:operation communicationType="request-response-noprocol"
    signatureLevel="none" encrypted="none"/>
  <input>
    <osci:container signatureLevel="none" encrypted="false">
      <osci:part>
        <osci:content part="tns:verifyCategoryRequest"/>
      </osci:part>
    </osci:container>
  </input>
  <output>
    <osci:container signatureLevel="advanced" encrypted="false">
      <osci:authorRef ref="DVDVAuthor"/>
      <osci:part>
        <osci:content part="tns:verifyCategoryResponse"/>
      </osci:part>
    </osci:container>
  </output>
</operation>
</binding>
```



## WSDL mit http-POST-Binding

Der einzige Verbindungsparameter für eine Kommunikation über einfaches http-POST mit dem DVDV-Slave ist die URI des DVDV-Applikationsservers. Eine Anpassung beschränkt sich daher auf das Ersetzen der URI. Ergänzend sollte Slave-Provider-Name und Zeitpunkt der Erstellung des Dokumentes eingetragen werden.

In Abbildung 12 sind die anzupassenden Stellen in einer Beispiel WSDL-Datei gelb hinterlegt.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Service -->
<definitions name="dvdvQuery"
targetNamespace="http://www.dvdv.de/2006/09/dvdvQuery.wsdl"
xmlns:tns="http://www.dvdv.de/2006/09/dvdvQuery.wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:dvdv="http://www.dvdv.de/messages/1.0/query"
xmlns:error="http://www.dvdv.de/messages/1.0"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <!--Datenstrukturen -->
  <types>
    <xs:schema targetNamespace="http://www.dvdv.de/2006/09/dvdvQuery.wsdl">
      <xs:import namespace="http://www.dvdv.de/messages/1.0/query"/>
      <xs:import namespace="http://www.dvdv.de/messages/1.0"/>
    </xs:schema>
  </types>

  <!--Nachrichten -->
  <message name="findServiceDescriptionRequest">
    <part name="findServiceDescriptionRequest" element="dvdv:find.servicedescription.request"/>
  </message>
  <message name="findServiceDescriptionResponse">
    <part name="findServiceResponse" element="wsdl:definitions"/>
  </message>
  <message name="findAuthorityDescriptionRequest">
    <part name="findAuthorityRequest" element="dvdv:find.servicedescription.request"/>
  </message>
  <message name="findAuthorityDescriptionResponse">
    <part name="findAuthorityResponse" element="dvdv:find.authoritydescription.request"/>
  </message>
  <message name="verifyCategoryRequest">
    <part name="verifyCategoryRequest" element="dvdv:verify.category.request"/>
  </message>
  <message name="verifyCategoryResponse">
    <part name="verifyCategoryResponse" element="dvdv:verify.category.response"/>
  </message>
  <message name="fault">
    <part name="fault" element="error:error.response"/>
  </message>
```

```
<!--Interfaces -->
<portType name="dvdvQueryInterface">
  <operation name="findServiceDescription">
    <input message="tns:findServiceDescriptionRequest"/>
    <output message="tns:findServiceDescriptionResponse"/>
    <fault message="tns:fault" name="fault"/>
  </operation>
  <operation name="findAuthorityDescription">
    <input message="tns:findAuthorityDescriptionRequest"/>
    <output message="tns:findAuthorityDescriptionResponse"/>
    <fault message="tns:fault" name="fault"/>
  </operation>
  <operation name="verifyCategory">
    <input message="tns:verifyCategoryRequest"/>
    <output message="tns:verifyCategoryResponse"/>
    <fault message="tns:fault" name="fault"/>
  </operation>
</portType>

<!--Bindung von Protokoll und Interface -->
<binding name="httpBinding" type="tns:dvdvQueryInterface">
  <http:binding verb="POST"/>
</binding>

<!--Endpoints des Services -->
<service name="dvdvQueryService">
  <port name="dvdvQueryHTTPPort" binding="tns:httpBinding">
    <documentation>
      Dieser Port ist nicht optional
    </documentation>
    <http:address location="http:// MySlave t:8080/DVDVWeb/PostAdapter"/>
  </port>
</service>

<!--Hinweise zum Dienstanbieter -->
<dvdv:supplier>
  <dvdv:authorityKey>dvdv:MyRZ-Provider</dvdv:authorityKey>
  <dvdv:timestamp>2007-10-31T10:30:00.000</dvdv:timestamp>
</dvdv:supplier>
</definitions>
```

Abbildung 12: Beispiel für angepasste WSDL-Datei des Query-Services mit http-POST-Binding

## Änderungshistorie

Version	Änderung
1.0.0	Initiale Version
1.0.1	„Meldebehörde“ statt „Meldeamt“
1.0.2	DVDVServerUnreachableException in DVDVNoResponseException umbenannt und DVDVOsciException hinzugefügt; Iteration über mehrere DVDV-Server im Fehlerfall korrigiert Default Messages für DVDVExceptions hinzugefügt DVDVAmbiguousKeyException hinzugefügt (Dienst-URI und Behördenschlüssel nicht eindeutig) Factory Methode für OSCI-Transportl hinzugefügt
1.1.0	find.authoritydescription hinzugefügt Interface Authority erweitert find in find.servicedescription umbenannt verify in verify.category umbenannt
1.2.0	Request-Response ohne Protokollierung „Optional Syntax“ im OSCI-Binding angepasst, so dass sie analog für HTTP-Bindung,... nutzbar ist Signaturzertifikat des Dienstelements „OSCI-Addressee“ entfernt
1.2.1	keine weitere Nachricht innerhalb eines OSCI-Dialoges, wenn 9xxx-Feedback vom Intermediär erhalten wurde Optimierung der Umschaltung auf Fallback-Server Osci-Bibliothek (1.2.2) und Bouncy-Castle (bcprov-jdk14-131.jar) eingebunden
1.3.0	CR „HTTP-Rechenzentrums-Query-Schnittstelle“ xsd-Dateien der Nachrichten in Distribution enthalten
1.3.2	Provider Interface angepasst (jetzt Serializable) Beispiele angepasst
1.4.0	JRE 1.5.0 u. 1.6.0 und OSCI-Bibliothek 1.3.0 integriert
1.4.5	CR „Erweiterung der FindAuthorityDescriptionResponse“ Log-Meldung bei fehlgeschlagener Antwort-Signaturprüfung angepasst Ausführbares Beispiel aufgenommen
1.4.6	Auswertung der Java Networking Properties "http.proxyHost", "http.proxyPort" und "http.nonProxyHosts" im HttpPostDVDVManager hinzugefügt. "VerifyCategoryResponse"-Parser korrigiert Umstellung auf jaxen-1.1.1 "FindAuthorityDescriptionResponse"-Parser liefert jetzt statt "null" einen leeren Vector, falls kein Proxy in der Antwort angegeben ist

## Referenzen

- [OSCIBib] Downloadbereich unter <http://www.osci.de>
- [WSDL] Spezifikation WSDL 1.1 unter <http://www.w3.org/TR/wsdl>
- [WSDLOSCI] Dokumentation der OSCI-Extension für WSDL „WSDL-Extension-OSCI.pdf“

## Abbildungsverzeichnis

Abbildung 1: Die DVDV-Bibliothek im Kontext beteiligter Systeme und Komponenten .....	3
Abbildung 2: XML-Schema der <i>find.servicedescription</i> -Anfrage .....	5
Abbildung 3: XML-Schema der <i>find.authoritydescription</i> -Anfrage.....	6
Abbildung 4: XML-Schema der <i>find.authoritydescription</i> -Antwort .....	7
Abbildung 5: XML-Schema der <i>verify.category</i> -Anfrage .....	8
Abbildung 6: XML-Schema der <i>verify.category</i> -Antwort.....	9
Abbildung 7: Paket-Struktur der DVDV-Bibliothek .....	10
Abbildung 8: Interfaces als Repräsentanten der DVDV-Anfrage- und Antwortnachrichten .....	11
Abbildung 9: XML-Schema einer DVDV-Fehlerantwortnachricht.....	14
Abbildung 10: Interfaces der Dienstbeschreibungselemente .....	15
Abbildung 11: Beispiel für angepasste WSDL-Datei des Query-Services mit OSCI-Binding.....	24
Abbildung 12: Beispiel für angepasste WSDL-Datei des Query-Services mit http-POST-Binding ..	26