

# VSPV-STANDARD 101

## TECHNISCHES DATENMODELL FÜR DEN DATENAUSTAUSCH IM GELEGENHEITSVERVERKEHR





VSPV-Standard

Reihe 100 – Mobilitätsdaten

Version 0.91

Bearbeiter Sascha Waltemate

Verband des privaten gewerblichen Straßenpersonenverkehrs Nordrhein-Westfalen VSPV e.V.  
Benninghofer Straße 152  
44269 Dortmund

0231 528227  
info@vspv-nrw.de



# Inhaltsverzeichnis

1	Einleitung.....	5
1.1	Ziel des VSPV-Standards 101 .....	5
1.2	Ausgangslage.....	6
1.3	Funktionserweiterung der NeTeX-Schnittstelle .....	8
1.4	Erläuterungen zum Dokument .....	13
1.4.1	Überblick über die Dokumentbestandteile .....	13
1.4.2	Verweis auf zugrundeliegende Normen und externe Dokumente .....	14
1.4.3	Konventionen bei Begriffen, Schreibweise und Beispielen .....	16
1.4.4	Hinweise zu Implementationshinweisen und Mappingtabellen .....	17
1.4.5	Umgang mit Erweiterungen und individuellen Anpassungen.....	18
2	Grundlagenmodellierung.....	19
2.1	Zentrale Modellierungsprinzipien und Grundelemente .....	24
2.2	Technische Konventionen und Erweiterungsmechanismen.....	34
2.3	Einordnung und Abgrenzung zu anderen Standards .....	41
3	Technische Konventionen .....	46
3.1	Einleitung und Zielsetzung.....	46
3.2	Persistente Identifikatoren und ID-Schemata .....	46
3.3	Feldgrößen, Zeichensätze und Namenskonventionen.....	63
3.4	Buchungs- und Zahlungslogik im Datenmodell.....	71
3.5	Flexible Bedienformen und technische Objekte.....	86
3.5.1	Technische Objekte zur Abbildung von Gelegenheitsverkehren.....	87
3.5.2	Praxisbeispiel: Krankenfahrt per Kioskbuchung aus der Klinik.....	90
3.6	Erweiterungsmechanismen (KeyValue & XML-Erweiterungen).....	91
3.6.1	Zweck und Geltungsbereich.....	91
3.6.2	Entscheidungslogik: KeyValue vs. Extensions .....	91
3.6.3	Namensräume, Governance und Versionierung.....	92
3.6.4	KeyValue-Profil (verbindliche Schlüssel, Datentypen, Validierung) .....	92
3.6.5	XML-Erweiterungen (<Extensions>) – Struktur und Leitplanken.....	93
3.6.6	Validierung und Prüfredeln .....	94
3.6.7	Kompatibilität, Migration und Interoperabilität.....	94
3.6.8	Beispiel-Set .....	95
3.6.9	Mappingtabelle (Auszug) .....	96
3.6.10	Konformitätskriterien (prüfbar).....	96
3.6.11	Anhang 3.6-A: VSPV KeyValue-Registry v0.91 (Auszug) .....	97
3.6.12	Anhang 3.6-B: Schematron-Snippets (Validierung) .....	99

3.6.13	Anhang 3.6-C: Mini-Registry als YAML (für Tooling) .....	103
3.7	REST-API-Schnittstellen für Buchung und Verfügbarkeit .....	104
3.7.1	Ziel und Abgrenzung .....	104
3.7.2	Grundlegende Designprinzipien .....	105
3.7.3	Ressourcenübersicht .....	105
3.7.4	Datenmodelle (REST) und Mapping zu NeTEx/VDV-462.....	106
3.7.5	Statusmodell (Buchung) .....	109
3.7.6	Sicherheit & Autorisierung .....	110
3.7.7	Idempotenz, Nebenläufigkeit, Fehlermanagement .....	110
3.7.8	Sonderfälle & Pflichtattribute .....	111
3.7.9	Webhooks.....	111
3.7.10	Beispielabläufe (konkret) .....	112
3.7.11	Mappingtabelle Kapitel 3.7 (Auszug).....	112
3.7.12	Konformitätskriterien (prüfbar).....	113
3.8	Echtzeit- und Auskunftsschnittstellen (TRIAS, SDG-Plattformen) .....	113
3.9	Versionierung, Validierung und Qualitätssicherung.....	115
3.9.1	Versionierung von Objekten .....	115
3.9.2	Gültigkeitszeiträume .....	116
3.9.3	Liefermodelle (Initial- und Delta-Updates).....	116
3.9.4	Technische Validierung (Schema) .....	116
3.9.5	Semantische Validierung.....	117
3.9.6	Qualitätssicherungsprozess (von der Erstellung bis Go-Live) .....	117
3.9.7	Artefakte für Validierung & QS .....	118
3.9.8	Konformität und Änderungsdisziplin.....	118
3.9.9	Beispiel: Delta-Lieferung (Versionserhöhung & Enddatierung) .....	118
4	Schnittstellen zu ÖPNV-Auskunfts- und Buchungssystemen (Integrationsstufen SDGV) ..	118
4.1	Statischer Datenaustausch (Integrationsstufe 1) .....	118
4.2	Buchungslinks (Integrationsstufe 2) .....	122
4.2.1	Modellierungsprinzip in VDV 462/NeTEx .....	122
4.2.2	Einheitliches URL-Schema (normativ) .....	123
4.2.3	Sicherheit & Datenschutz (Stufe 2) .....	123
4.2.4	Platzierung im Datenmodell.....	124
4.2.5	Kompatibilität zu Bestandswelten .....	124
4.2.6	Validierung (ergänzend zu Kap. 3.3) .....	124
4.2.7	Praxisbeispiel (Stufe 2, Kiosk/App) .....	125
4.2.8	Mappingtabelle .....	125

4.3	Tiefenintegration (Integrationsstufe 3: TRIAS-Flows & REST-Bridge, SSO-Hinweis → Kap. 7)	126
4.3.1	Zielbild & Architektur .....	126
4.3.2	Vorgangstypen (Pflichtenheft) .....	126
4.3.3	TRIAS-Profil (Anfrage/Antwort – konzeptionell) .....	127
4.3.4	REST-Bridge (Mapping auf Kap. 3.7) .....	128
4.3.5	Daten und Datenschutz (Auszug) .....	129
4.3.6	Sicherheit (Kurzprofil; Details in Kap. 8 & Kap. 7) .....	129
4.3.7	Fehler- & Zeitverhalten.....	129
4.3.8	Validierung & QS (Querschnitt).....	129
4.3.9	Praxisbeispiel (End-to-End, vereinfacht) .....	130
4.3.10	Normative Mindestanforderungen (Stufe 3) .....	130
5	Back-End-Integration in Betreiber-/Anbietersysteme (Disposition).....	131
5.1	Datenschnittstellen zur Disposition (Verfügbarkeit, Zuweisung, Status) .....	131
5.1.1	Zielbild und Abgrenzung .....	131
5.1.2	Grundsätze (normativ) .....	131
5.1.3	Schnittstellenübersicht .....	131
5.1.4	Datenmodelle (Auszug, normativ) .....	134
5.1.5	Zustandsautomaten & Mapping .....	134
5.1.6	Fehler-/Timeout-Regeln .....	135
5.1.7	Sicherheit & Compliance (Kurzprofil; Details Kap. 7/8) .....	135
5.1.8	Leistungskennzahlen & Betriebsgrenzen .....	136
5.1.9	Praxisbeispiel (konkret, Kiosk-Krankenfahrt → Taxi/BTW).....	136
5.1.10	Mappingtabelle (fachlich → technisch).....	136
5.2	Integration von Dritt-Bestellern (Arztpraxis, Krankenhaus, Kiosk) über die zentrale Buchungs-API.....	136
5.2.1	Zweck und Geltungsbereich.....	136
5.2.2	Rollen, Objekte, Kanäle (normativ) .....	137
5.2.3	Sicherheit und Autorisierung.....	137
5.2.4	Datenminimierung (DSGVO-konform) .....	138
5.2.5	Spezifika für Krankenfahrten mit Muster-4-Transportverordnung.....	138
5.2.6	Datenmodell-Erweiterung medicalOrdinance.....	138
5.2.7	Endpunkte: Übergabe von Muster-4-Daten .....	139
5.2.8	QR-Code-Import (Praxissoftware/Kiosk) .....	140
5.2.9	Validierung (spezifisch Muster-4) .....	141
5.2.10	Fehlercodes (Ergänzungen).....	141
5.2.11	Praxisbeispiel: Entlassfahrt mit QR-Übernahme (Klinikum Hamm) .....	141

5.2.12	Ergänzung der Mappingtabelle (Krankenfahrten) .....	142
5.2.13	Endpunkte für Dritt-Besteller (fachlich kanonisch) .....	142
5.2.14	Pflichtfelder nach Kanal.....	144
5.2.15	Kostenträger & Abrechnung (Schnittstelle zu Kap. 6) .....	144
5.2.16	Validierung, Idempotenz, Fehlercodes .....	145
5.2.17	Terminal-/Kiosk-Lifecycle .....	145
5.2.18	Praxisbeispiele.....	145
5.2.19	Mappingtabelle (fachlich → technisch).....	146
5.3	Sonderfall: Anbindung von Rettungsleitstellen (niedrigschwellige Krankentransporte) 146	
6	Abrechnungslogik und Kostenträger-Integration.....	151
6.1	Zahlungsinformationen und Fahrpreisbildung .....	151
6.2	Abrechnung mit Kostenträgern (Krankenkassen, Kommunen) .....	154
6.3	Clearing und Erlösaufteilung (ÖPNV-Taxi, Pooling, Fördermodelle).....	156
6.4	Fehlercodes, Idempotenz, Audit (kapitelübergreifend).....	158
6.5	Mappingtabelle Kapitel 6.....	158
7	Nutzermanagement und Authentifizierung .....	159
7.1	Benutzerrollen und Rechte .....	159
7.1.1	Rollen (RBAC).....	159
7.1.2	Scopes (feingranular) .....	160
7.1.3	Delegation („acting on behalf of“) .....	161
7.2	Authentifizierungsverfahren.....	161
7.2.1	Endnutzer & Kiosk: OIDC Authorization Code (mit PKCE).....	161
7.2.2	System-zu-System: mTLS-Client-Credentials .....	162
7.2.3	SSO-Handshakes (Verweis auf Kap. 4.3).....	162
7.3	Datenschutz und Datenminimierung .....	162
7.3.1	Prinzipien.....	162
7.3.2	Identitäts- und Personenobjekte (Minimalsatz).....	163
7.3.3	Delegationsnachweis (Krankenfahrten) .....	163
7.3.4	Audit & Nachvollziehbarkeit.....	163
7.4	Tokenformate, Lebensdauern, Schlüssel.....	164
7.5	Mappingtabelle Kapitel 7 .....	164
8	Sicherheits- und Zugriffskonzepte .....	165
8.1	Datenautorisation und Zugriffskontrolle .....	165
8.1.1	Grundprinzipien (normativ) .....	165
8.1.2	Autorisierungsregeln pro API-Bereich (Auszug) .....	165
8.1.3	Schlüssel- & Geheimnisverwaltung.....	166

8.1.4	Datenklassifikation & Zugriffspfade .....	166
8.2	Protokollierung und Monitoring.....	167
8.2.1	Audit-Logging (normativ).....	167
8.2.2	Betriebs-Monitoring & SIEM .....	167
8.2.3	Incident-Response & Verwundbarkeiten.....	167
8.2.4	Aufbewahrung.....	168
8.3	Datenschutz und Datensicherheit .....	168
8.3.1	Kryptografie & Speichersicherheit .....	168
8.3.2	Datenminimierung & Zweckbindung.....	168
8.3.3	Löschung, Sperrung, Aufbewahrung.....	168
8.3.4	Sicherer Datenaustausch & Eingabevalidierung.....	169
8.3.5	Backups, Wiederanlauf & Resilienz .....	169
8.3.6	Secure-Development-Lifecycle (SDL).....	169
8.4	Mappingtabelle Kapitel 8 .....	169
9	Anhang A – ID-/Namensraum-Registry .....	171
10	Anhang B – KeyValue-Schlüssel-Registry.....	172
11	Anhang C – XML-Erweiterungen/Namensräume .....	174
12	Anhang D – Feldgrößen-/Zeichen-Master .....	175
13	Anhang E – API-Endpunkte × Scopes-Matrix .....	176
14	Anhang F – Fehler-/Reason-Codes.....	177
15	Anhang G – Buchungsstatus-/Event-Matrix.....	178
16	Anhang H – FareProducts-/Tarif-Katalog.....	179
17	Anhang I – Kostenträger-Datenfelder.....	180
18	Anhang J – TRIAS-Nachrichten-Matrix .....	181
19	Anhang K – Prüf- & Konformitätsmatrix.....	182
20	Anhang L – Beispiel-Feeds & Payloads .....	183
21	Verwaltungs- und Konformitätshinweise (für alle Anhänge).....	184

# 1 Einleitung

## 1.1 Ziel des VSPV-Standards 101

Der VSPV-Standard 101 verfolgt das Ziel, ein technisches Datenmodell für Gelegenheitsverkehre im öffentlichen Personennahverkehr (ÖPNV) Nordrhein-Westfalens zu definieren, das auf dem europäischen NeTEx-Standard basiert und sich an der Systematik und Detaillierung der VDV-Schrift 462 orientiert. Im Fokus stehen dabei Gelegenheitsverkehre nach §47 PBefG (Taxi) und §49 PBefG (Mietwagen) sowie Bedarfsverkehre nach §42/44a PBefG (On-Demand-Angebote) als integrale Bestandteile moderner, flexibler Mobilitätsangebote im ÖPNV.

Mit dem Standard wird eine einheitliche, interoperable und herstellerunabhängige Grundlage für die Modellierung, Bereitstellung und den Austausch aller für Fahrgastinformation und Buchung relevanten Daten geschaffen. Ziel ist es, die Integration von Gelegenheitsverkehren in bestehende und zukünftige ÖPNV-Auskunfts-, Buchungs- und Vertriebssysteme zu ermöglichen und so die Voraussetzungen für eine diskriminierungsfreie, durchgängige und medienbruchfreie Servicekette „Informieren – Buchen – Bezahlen“ zu schaffen.

Der VSPV-Standard 101 adressiert die spezifischen Anforderungen, die sich aus der Vielfalt der Betriebsformen, Dispositionsregeln und technischen Plattformen im Bereich der Gelegenheitsverkehre ergeben. Er legt Profile und technische Konventionen für die Abbildung von Bedienebenen, Buchungsprozessen, Tarifmodellen und die Integration von Taxi- und Mietwagenverkehren fest. Dabei werden die drei in der SDGV-Projektdokumentation beschriebenen Integrationsstufen berücksichtigt:

- **Stufe 1:** Reiner Datenaustausch (statische Netz- und Fahrplandaten für die Fahrgastinformation)
- **Stufe 2:** Datenaustausch mit Buchungslink (Verknüpfung zu externen Buchungssystemen)
- **Stufe 3:** Tiefenintegration (Bereitstellung sämtlicher Buchungsinformationen und Buchungsfunktionen innerhalb des Fahrgastinformationssystems)

Der Standard richtet sich an Systemhersteller, Verkehrsunternehmen, Verbände, Aufgabenträger und Betreiber von Mobilitätsplattformen, die Gelegenheitsverkehre planen, betreiben oder in digitale ÖPNV-Angebote integrieren. Mit der vorliegenden Spezifikation wird die Grundlage geschaffen, die gesetzlichen Anforderungen an die Datenbereitstellung – insbesondere gemäß Mobilitätsdatenverordnung – zu erfüllen und innovative Mobilitätsangebote im Sinne von Mobility-as-a-Service (MaaS) zu realisieren.

Durch die konsequente Orientierung an bestehenden Standards und die Erweiterung um gelegenheitsverkehrsspezifische Anforderungen trägt der VSPV-Standard 101 dazu bei, die Datenhoheit bei den Betreibern zu belassen, die Integration in bestehende Systemlandschaften zu ermöglichen und die Voraussetzungen für eine zukunftsfähige, digitale Vernetzung im ÖPNV zu schaffen.

## 1.2 Ausgangslage

Die Entwicklung des VSPV-Standards 101 erfolgt vor dem Hintergrund einer sich wandelnden Mobilitätslandschaft im öffentlichen Personennahverkehr (ÖPNV) Nordrhein-Westfalens, in der Gelegenheitsverkehre nach §47 PBefG (Taxi) und §49 PBefG (Mietwagen) sowie Bedarfsverkehre nach §42/44a PBefG (On-Demand-Angebote) eine immer größere Bedeutung erlangen. Die Digitalisierung und die zunehmende Vernetzung von Mobilitätsangeboten führen dazu, dass Fahrgäste heute erwarten, alle verfügbaren Verkehrsformen aus einer Hand informiert, buchen und bezahlen zu können. Damit rücken die Integration und Standardisierung von Gelegenheitsverkehren in die bestehenden digitalen ÖPNV-Systeme in den Fokus.

Die bislang etablierten technischen Standards und Schnittstellen – wie etwa die VDV-Schrift 462 auf Basis des europäischen NeTEx-Standards – sind auf den klassischen Linienverkehr mit festen Haltestellen, Fahrplänen und Linienführungen ausgerichtet. Für die Vielzahl an flexiblen, bedarfsorientierten Verkehrsformen, die sich durch dynamische Bedienegebiete, variable Betriebszeiten und unterschiedliche Buchungs- und Dispositionsmodelle auszeichnen, existiert bislang jedoch keine umfassende, interoperable und herstellerunabhängige technische Spezifikation. Dies betrifft insbesondere die Integration von On-Demand-Ridepooling, flexiblen Rufbusangeboten sowie Taxiverkehren, die sowohl im ländlichen Raum als auch in urbanen Gebieten eine wichtige Ergänzung zum Linienverkehr darstellen.

Die aktuelle Praxis in Nordrhein-Westfalen ist durch eine heterogene Systemlandschaft geprägt: Unterschiedliche Verkehrsunternehmen und Aufgabenträger setzen eine Vielzahl von Planungssystemen, Dispositionsplattformen und Auskunftssystemen ein, die jeweils eigene Datenmodelle und Schnittstellen verwenden. Proprietäre Lösungen und Insellösungen erschweren den standardisierten, landes- und bundesweiten Austausch von Fahrgastinformationen, Buchungsoptionen und Betriebsdaten. Die Folge sind Medienbrüche, eingeschränkte Interoperabilität und ein erhöhter manueller Aufwand für die Datenpflege und -integration.

Vor diesem Hintergrund ist die Entwicklung eines einheitlichen, auf bestehenden Standards aufbauenden technischen Datenmodells für Gelegenheitsverkehre eine zentrale Voraussetzung, um die gesetzlichen Anforderungen an die Datenbereitstellung – etwa nach Mobilitätsdatenverordnung – zu erfüllen und innovative, kundenorientierte Mobilitätsangebote im Sinne von Mobility-as-a-Service (MaaS) zu ermöglichen.

Die Analyse der bestehenden Systemlandschaft in Nordrhein-Westfalen zeigt deutlich, dass die Abbildung und Integration von Gelegenheitsverkehren – insbesondere von flächengebundenen On-Demand-Angeboten und Taxiverkehren – mit erheblichen Herausforderungen verbunden ist. Die eingesetzten Planungssysteme, Dispositionsplattformen und Auskunftssysteme wie IVU.pool, DIVA, HAFAS oder EFA sind historisch auf den klassischen Linienverkehr mit festen Fahrplänen und Haltestellenfolgen ausgerichtet. Sie bieten zwar erste Möglichkeiten, flexible Bedienformen zu modellieren, stoßen aber bei der Abbildung dynamischer Bedienegebiete, unscharfer Fahrpläne und komplexer Dispositionsregeln schnell an ihre Grenzen.

Die derzeitige Praxis ist geprägt von einer Vielzahl proprietärer Datenmodelle und Schnittstellen, die jeweils spezifisch auf die Anforderungen einzelner Betreiber oder Systemhersteller zugeschnitten sind. Dies führt zu einer fragmentierten Datenlandschaft, in der der Austausch von fahrgastrelevanten Informationen, Buchungsoptionen und Betriebsdaten zwischen den beteiligten Systemen nur eingeschränkt möglich ist. Medienbrüche, manuelle Nacharbeiten und eine eingeschränkte Aktualität der bereitgestellten Daten sind die Folge. Insbesondere die Pflege und Aktualisierung von Bedienebenen, Buchungsregeln und Tarifinformationen erfolgt häufig außerhalb der etablierten Fahrplandatenprozesse und ist mit erheblichem personellen Aufwand verbunden.

Ein weiteres zentrales Problem besteht darin, dass die Datenhoheit und -verantwortung für Gelegenheitsverkehre vielfach nicht beim Betreiber, sondern bei Datenintegratoren oder Auskunftssystemen liegt. Dadurch können betriebliche Änderungen, etwa bei Dispositionsregeln oder Bedienzeiten, nicht zeitnah und konsistent in die digitalen Auskunfts- und Buchungssysteme übernommen werden. Die Folge ist, dass Fahrgäste nicht immer auf aktuelle und vollständige Informationen zugreifen können und innovative Mobilitätsangebote ihr Potenzial nicht voll entfalten können.

Zudem existieren bislang keine einheitlichen, herstellerübergreifenden Standards für die Beschreibung und den Austausch der für Gelegenheitsverkehre erforderlichen Daten. Zwar bieten Formate wie VDV 462/NeTeX, GTFS-flex oder TRIAS einzelne Lösungsansätze, doch decken sie die Vielfalt der Betriebsformen, Dispositionsregeln und Integrationsstufen bislang nur unvollständig ab. Die SDGV-Projektdokumentation hat diesen Handlungsbedarf systematisch herausgearbeitet und die Notwendigkeit einer standardisierten, interoperablen und zukunftsfähigen technischen Spezifikation für Gelegenheitsverkehre klar benannt.

Vor dem Hintergrund dieser Herausforderungen und der im SDGV-Projekt dokumentierten Anforderungen wurde deutlich, dass ein einheitlicher, herstellerunabhängiger Standard für den Datenaustausch von Gelegenheitsverkehren bislang fehlt, der den vielfältigen betrieblichen Realitäten und Integrationsstufen gerecht wird. Die Analyse bestehender Standards und Schnittstellen – darunter VDV 462/NeTeX, GTFS-flex, TRIAS und weitere – zeigt, dass zwar viele Einzellösungen existieren, jedoch kein Standard alle relevanten Aspekte von Gelegenheitsverkehren, insbesondere die Integration von Buchungsfunktionen und Echtzeitdaten, vollständig abdeckt.

Das SDGV-Projekt hat deshalb ein Stufenmodell für den Datenaustausch entwickelt, das die unterschiedlichen Integrationsgrade von Gelegenheitsverkehren in Auskunfts- und Buchungssysteme abbildet:

Stufe 1: Reiner Datenaustausch für die Fahrgastinformation (statische und dynamische Betriebsdaten)

Stufe 2: Austausch von Buchungslinks, um einen medienbruchfreien Absprung in externe Buchungssysteme zu ermöglichen

Stufe 3: Tiefenintegration, bei der sämtliche Buchungsinformationen und -funktionen direkt im Auskunfts- und Vertriebssystem bereitgestellt werden.

Ziel des VSPV-Standards 101 ist es, auf Basis der VDV-Schrift 462 und der SDGV-Ergebnisse ein technisches Datenmodell zu schaffen, das diese Integrationsstufen abbildet und die Betreiber in die Lage versetzt, alle für die Fahrgastinformation und Buchung relevanten Daten standardisiert und interoperabel bereitzustellen. Die Spezifikation legt dabei besonderen Wert auf die Datenhoheit der Betreiber, die Abbildung der Vielfalt betrieblicher Modelle und die Kompatibilität mit bestehenden und künftigen Anforderungen der Mobilitätsdatenverordnung. Damit wird die Grundlage geschaffen, Gelegenheitsverkehre – einschließlich Taxiverkehren – als gleichberechtigten Bestandteil des ÖPNV in digitale Mobilitätsplattformen und MaaS-Ökosysteme zu integrieren und die Servicekette „Informieren – Buchen – Bezahlen“ durchgängig und medienbruchfrei zu gestalten.

### 1.3 Funktionserweiterung der NeTEx-Schnittstelle

Die VDV-Schrift 462 bildet mit ihrer Implementierung des europäischen NeTEx-Standards einen etablierten Rahmen für den Austausch von Netz- und Fahrplandaten im klassischen Linienverkehr des ÖPNV. In den letzten Jahren hat sich jedoch die Angebotslandschaft im öffentlichen Verkehr deutlich erweitert: Gelegenheitsverkehre nach §47 PBefG (Taxi) und §49 PBefG (Mietwagen), Bedarfsverkehre nach §42/44a PBefG (On-Demand- und Ridepooling-Angebote) gewinnen zunehmend an Bedeutung und stellen neue Anforderungen an die technische Datenhaltung und den interoperablen Datenaustausch.

Die Analyse der System- und Datenlandschaft in Nordrhein-Westfalen, wie sie im Rahmen des SDGV-Projekts vorgenommen wurde, zeigt, dass die etablierten Standards und Schnittstellen – darunter auch die VDV 462 – auf die festen Strukturen des Linienverkehrs ausgerichtet sind. Sie bieten bislang keine umfassende Abbildungsmöglichkeit für die spezifischen Merkmale und Prozesse von Gelegenheitsverkehren. Wesentliche Herausforderungen sind unter anderem:

- die Modellierung dynamischer Bedienegebiete (z. B. als Polygone oder flexible Sammelpunkte),
- die Abbildung unscharfer Fahrpläne und variabler Betriebszeiten,
- die Integration von Buchungsprozessen und Echtzeitinformationen,
- sowie die Einbindung von Taxiverkehren als gleichberechtigte Mobilitätsform.

Im Bestand existieren zahlreiche proprietäre Lösungen und Insellösungen, die den Datenaustausch zwischen Planung, Disposition, Auskunft und Buchung erschweren und eine medienbruchfreie Servicekette verhindern. Die Mobilitätsdatenverordnung und die Anforderungen an Mobility-as-a-Service (MaaS) verlangen jedoch eine standardisierte, interoperable und diskriminierungsfreie Datenbasis, die alle relevanten Verkehrsformen einschließt.

Vor diesem Hintergrund ist eine gezielte Erweiterung des NeTEx-Standards notwendig, um die fachlichen und technischen Anforderungen der Gelegenheitsverkehre abzudecken. Ziel des VSPV-Standards 101 ist es daher, aufbauend auf der VDV-Schrift 462 und unter

Einbeziehung der SDGV-Ergebnisse, die bestehenden NeTEx-Profile um die für Gelegenheitsverkehre erforderlichen Funktionalitäten zu ergänzen. Damit soll ein durchgängiger, systemübergreifender und zukunftsfähiger Austausch von Informationen – von der Fahrgastinformation bis zur Buchung – ermöglicht werden.

Die Integration von Gelegenheitsverkehren – insbesondere flächengebundener On-Demand-Angebote und Taxiverkehre – in bestehende NeTEx/VDV-462-Systemlandschaften erfordert die gezielte Erweiterung und Anpassung des bestehenden Datenmodells. Während der klassische Linienverkehr durch feste Haltestellen, Linienwege und Fahrpläne gekennzeichnet ist, zeichnen sich Gelegenheitsverkehre durch flexible Bediengebiete, variable Betriebszeiten, unterschiedliche Buchungslogiken und eine Vielzahl von Dispositionsregeln aus. Diese Merkmale sind im bisherigen NeTEx-Standard und den zugehörigen VDV-Profilen nur teilweise oder gar nicht abbildbar und machen eine Erweiterung um spezifische Datenstrukturen und Attribute erforderlich.

### **Zentrale Erweiterungen und Anpassungen im Datenmodell betreffen insbesondere:**

- **Flexible Bediengebiete:**

Die Abbildung von Bediengebieten als Polygone, Sammelpunkte oder Adressbereiche ist für flächengebundene On-Demand-Verkehre essenziell. Hierzu wird das bestehende Element FlexibleStopPlace genutzt und um Attribute für Geometrien (z. B. WGS84-Polygon), zulässige Ein-/Ausstiegsarten (Haltestelle, Adresse, Sammelpunkt) und optionale Dispositionsregeln erweitert.

*Beispiel:*

```
<FlexibleStopPlace id="VSPV:StopPlace:NRW-1001" version="1.0">
  <Polygon>
    <!-- WGS84-Koordinaten -->
  </Polygon>
  <keyList>
    <KeyValue>
      <Key>BEDIENART</Key>
      <Value>Adresse, Haltestelle</Value>
    </KeyValue>
  </keyList>
</FlexibleStopPlace>
```

- **Unschärfe Fahrpläne und Zeitfenster:**

Für Gelegenheitsverkehre sind feste Fahrpläne oft nicht anwendbar. Stattdessen werden Zeitfenster mit flexiblen Abfahrtszeiten und Frequenzen benötigt. Das Datenmodell muss daher die Möglichkeit bieten, Bedienzeiten, Buchungsfristen und unscharfe Fahrten (z. B. „Fahrt möglich zwischen 18:00 und 22:00 Uhr, alle 30 Minuten nach Bedarf“) darzustellen.

*Beispiel:*

```
<FlexibleServiceJourney id="VSPV:Journey:2025-Q3-501" version="1.0">
  <BookingProcess>
    <latestBookingTime>PT30M</latestBookingTime>
```

```

    <bookingMethods>mobileApp,telephone</bookingMethods>
  </BookingProcess>
  <ValidBetween>
    <FromDate>2025-06-01</FromDate>
    <ToDate>2025-12-31</ToDate>
  </ValidBetween>
</FlexibleServiceJourney>

```

- **Buchungs- und Dispositionsregeln:**

Gelegenheitsverkehre erfordern die Beschreibung von Buchungswegen (App, Telefon, Web), Vorlaufzeiten, Mindestfahrweiten, Ausschlussregeln (z. B. keine Innerortsfahrt) und Kapazitätsgrenzen. Diese Informationen werden als zusätzliche Attribute, meist im keyList- oder Extensions-Bereich, modelliert.

*Beispiel:*

```

<keyList>
  <KeyValue>
    <Key>MINDESTFAHRWEITE</Key>
    <Value>3km</Value>
  </KeyValue>
  <KeyValue>
    <Key>INNERORTSVERBOT</Key>
    <Value>>true</Value>
  </KeyValue>
  <KeyValue>
    <Key>MAX_KAPAZITAET</Key>
    <Value>8</Value>
  </KeyValue>
</keyList>

```

- **Taxi-Integration:**

Für eigenwirtschaftliche Taxiverkehre nach § 47 PBefG werden spezifische Attribute benötigt, etwa zur Angabe des Taxitarifs nach § 51 Abs. 1 PBefG oder gemeinwirtschaftlicher Tarifsstitution nach VO 1370/2007, der Kapazität und der Buchungsmodalitäten. Die Integration erfolgt über das Element FlexibleLine mit spezifischen Attributen für die Taxi-Charakteristika.

*Beispiel:*

```

<FlexibleLine id="VSPV:TaxiLine:T-500" transportMode="taxi">
  <keyList>
    <KeyValue>
      <Key>BEFÖRDERUNGSART</Key>
      <Value>PBefG§47</Value>
    </KeyValue>
    <KeyValue>
      <Key>TARIFREF</Key>
      <Value>TaxiVO_Hagen_2025</Value>
    </KeyValue>
  </keyList>

```

```
</keyList>
<PassengerCapacity>4</PassengerCapacity>
</FlexibleLine>
```

- **Abbildung von Integrationsstufen:**

Die drei Integrationsstufen (reiner Datenaustausch, Buchungslink, Tiefenintegration) werden durch zusätzliche Strukturelemente, beispielsweise für Deeplinks (BookingLink) oder Echtzeitdatenreferenzen (realTimeDataRef), abgebildet.

*Beispiel:*

```
<Extensions>
<BookingLink>https://booking.vspv.de?journey=201</BookingLink>
<realTimeDataRef>RTD_VSPV_301</realTimeDataRef>
</Extensions>
```

Diese Erweiterungen werden so gestaltet, dass sie mit den bestehenden Mechanismen des NeTEx-Standards kompatibel sind und die Interoperabilität mit klassischen Linienverkehrsdaten gewährleisten. Die Modellierung erfolgt bevorzugt durch Nutzung vorhandener Strukturen, ergänzt um gezielte Erweiterungen mittels keyList und Extensions, um die spezifischen Anforderungen der Gelegenheitsverkehre abzubilden, ohne die Kompatibilität zu bestehenden VDV-462-Profilen zu beeinträchtigen.

Die Integration von Gelegenheitsverkehren in das bestehende NeTEx/VDV-462-Datenmodell stößt bei bestimmten Anforderungen an die Grenzen der im Standard vorgesehenen Strukturen. Um dennoch eine vollständige und interoperable Abbildung betrieblicher Realitäten zu ermöglichen, sieht der NeTEx-Standard verschiedene Mechanismen für proprietäre Erweiterungen vor, die auch in der VDV-Schrift 462 detailliert beschrieben sind.

### 1. Nutzung von Extensions

NeTEx erlaubt es, an nahezu jedem Datenobjekt einen Bereich <Extensions> anzufügen, in dem zusätzliche, nicht im Standard beschriebene Strukturen hinterlegt werden können. Dies ist insbesondere dann erforderlich, wenn neue Attribute oder komplexe Objekte abgebildet werden müssen, die für Gelegenheitsverkehre spezifisch sind, etwa für die Tiefenintegration von Buchungsprozessen oder für die dynamische Preisbildung bei Taxiverkehren.

*Beispiel:*

```
<FlexibleServiceJourney id="VSPV:Journey:2025-Q3-501" version="1.0">
<Extensions>

<vspv:BookingLink>https://booking.vspv.de?journey=501</vspv:BookingLink>
<vspv:RealTimeDataRef>RTD_VSPV_501</vspv:RealTimeDataRef>
</Extensions>
</FlexibleServiceJourney>
```

Die Nutzung von Extensions sollte dokumentiert und zwischen den beteiligten Systempartnern abgestimmt werden, um die Interoperabilität zu sichern. Die VDV

462 empfiehlt, Extensions möglichst restriktiv und nur dort einzusetzen, wo eine Abbildung mit Standardmitteln nicht möglich ist.

## 2. Verwendung von Key-Value-Paaren in keyList

Für die Übertragung zusätzlicher Identifikatoren oder einfacher Attribute, die nicht im Standardmodell vorgesehen sind, sieht NeTEx die Verwendung von Key-Value-Paaren in einer keyList vor. Dies eignet sich besonders für Zusatzinformationen wie Mindestfahrweiten, Ausschlussregeln oder interne Referenzen.

*Beispiel:*

```
<keyList>
  <KeyValue>
    <Key>MINDESTFAHRWEITE</Key>
    <Value>3km</Value>
  </KeyValue>
  <KeyValue>
    <Key>INNERORTSVERBOT</Key>
    <Value>>true</Value>
  </KeyValue>
</keyList>
```

Die VDV-Schrift 462 beschreibt dieses Prinzip explizit für die Übertragung zusätzlicher IDs und empfiehlt, die Schlüsselbezeichnungen eindeutig und dokumentiert zu wählen.

## 3. Erweiterung für zusätzliche Datenstrukturen

Wenn Anforderungen bestehen, die über einzelne Attribute hinausgehen – etwa die Modellierung komplexer Buchungs- oder Preisinformationen –, kann innerhalb von <Extensions> eine beliebige XML-Struktur eingefügt werden. Die VDV 462 weist darauf hin, dass solche Erweiterungen stets mit den beteiligten Partnern abzustimmen sind und idealerweise in eine künftige Standardisierung einfließen sollten.

*Beispiel:*

```
<Extensions>
  <vspv:TaxiTariff>
    <vspv:BaseFare>4.50</vspv:BaseFare>
    <vspv:PerKm>2.20</vspv:PerKm>
    <vspv:NightSurcharge>1.00</vspv:NightSurcharge>
  </vspv:TaxiTariff>
</Extensions>
```

## 4. Empfehlungen zur Interoperabilität

Die VDV 462 empfiehlt, Erweiterungen zunächst mit den vorhandenen Standardmechanismen (KeyValue, Extensions) zu realisieren und diese in der Schnittstellendokumentation klar zu beschreiben. Proprietäre Lösungen sollten so gestaltet werden, dass sie die Verarbeitung durch Standard-NeTEx-Parser nicht

behindern. Darüber hinaus sollte eine kontinuierliche Abstimmung mit dem VDV und den relevanten Arbeitsgruppen erfolgen, um perspektivisch eine Aufnahme in den offiziellen Standard zu ermöglichen.

## **5. Beispiele und Mappingtabellen**

Zur Unterstützung der Implementierung sollten für alle proprietären Erweiterungen Beispiel-XMLs und Mappingtabellen bereitgestellt werden, die die Zuordnung der neuen Elemente zu den fachlichen Anforderungen und den betroffenen NeTEx-Objekten dokumentieren. Die VDV 462 sieht vor, dass solche Mappingtabellen Bestandteil der technischen Spezifikation sind und regelmäßig aktualisiert werden.

### **Fazit:**

Durch die gezielte Nutzung von Extensions und Key-Value-Mechanismen können auch komplexe und bislang nicht im Standard abbildbare Anforderungen der Gelegenheitsverkehre interoperabel und nachvollziehbar in NeTEx/VDV-462-Systemlandschaften integriert werden, ohne die Kompatibilität mit bestehenden Anwendungen zu gefährden.

## **1.4 Erläuterungen zum Dokument**

### **1.4.1 Überblick über die Dokumentbestandteile**

Der VSPV-Standard 101 ist modular aufgebaut und besteht aus mehreren, aufeinander abgestimmten Dokument- und Arbeitsbestandteilen. Diese Struktur gewährleistet, dass sowohl konzeptionelle als auch technische, praxisorientierte und prüfbare Aspekte des Standards abgedeckt werden. Die einzelnen Bestandteile sind im Folgenden beschrieben:

#### **Handbuch (vorliegendes Dokument):**

Das Handbuch bildet das zentrale Referenzdokument des VSPV-Standards 101. Es beschreibt die Zielsetzung, die fachlichen und technischen Grundlagen, die Modellierungskonzepte sowie die Anwendungskonventionen für Gelegenheitsverkehre und Taxi-Integration. Die Gliederung orientiert sich an der VDV-Schrift 462 und folgt einer systematischen Darstellung der relevanten Anwendungsbereiche, Datenstrukturen und Prozesse. Das Handbuch ist als fortlaufend gepflegtes Dokument konzipiert und wird bei Bedarf um neue Anwendungsfälle und technische Entwicklungen ergänzt.

#### **Mappingtabellen:**

Ein zentrales Element des Standards sind die Mappingtabellen, die als Excel- oder CSV-Dateien bereitgestellt werden. Sie dokumentieren die Zuordnung aller im VSPV-Standard 101 beschriebenen fachlichen Datenelemente zu den entsprechenden Elementen des NeTEx-XSD-Schemas. Die Mappingtabellen enthalten für jedes Datenelement den vollständigen Pfad im NeTEx-Modell, Hinweise zur Kardinalität, zu Pflichtfeldern sowie zu

spezifischen Erweiterungen oder Einschränkungen. Sie dienen als verbindliche Referenz für die technische Implementierung und Validierung.

### **Beispiel-XMLs:**

Für alle zentralen Anwendungsfälle – wie etwa die Modellierung eines Bedienegebiets, die Abbildung eines Buchungslincks oder die Integration von Taxi-Tarifen – werden validierte Beispiel-XML-Dateien zur Verfügung gestellt. Diese Beispiele illustrieren die praktische Umsetzung der im Standard beschriebenen Anforderungen und dienen als Vorlage für die Implementierung in operativen Systemen. Sie unterstützen zudem die Qualitätssicherung und erleichtern die Kommunikation zwischen den beteiligten Systempartnern.

### **XSD-Schema:**

Der Standard referenziert das für die Validierung zu verwendende XML-Schema (NeTeX/VDV 462). Das Schema ist maßgeblich für die technische Prüfung der Datenlieferungen und stellt sicher, dass alle gelieferten Daten den strukturellen Vorgaben entsprechen. Für Erweiterungen oder spezifische Anpassungen werden ergänzende Schematron- oder SHACL-Definitionen bereitgestellt, sofern dies für die Validierung notwendig ist.

### **Ergänzende Materialien:**

Je nach Bedarf werden weitere unterstützende Dokumente – etwa Implementierungshinweise, technische Leitfäden, SHACL-Constraints für die semantische Validierung oder Erläuterungen zu spezifischen Use Cases – bereitgestellt. Diese Materialien dienen der Vertiefung einzelner Aspekte und unterstützen die Anwender bei der Umsetzung und Weiterentwicklung des Standards.

### **Verbindlichkeit und Zusammenspiel:**

Alle genannten Bestandteile sind integraler Teil des VSPV-Standards 101. Im Falle von Abweichungen oder Unklarheiten zwischen den einzelnen Dokumenten gilt die Mappingtabelle als maßgeblich für die technische Umsetzung. Das Zusammenspiel dieser Komponenten gewährleistet eine konsistente, nachvollziehbare und praxistaugliche Anwendung des Standards und unterstützt alle beteiligten Akteure bei der Einführung und Pflege interoperabler Datenstrukturen für Gelegenheitsverkehre im ÖPNV.

## **1.4.2 Verweis auf zugrundeliegende Normen und externe Dokumente**

Der VSPV-Standard 101 ist nicht als isolierte Spezifikation zu verstehen, sondern baut konsequent auf bestehenden, national und international anerkannten Normen, Standards und Projektdokumentationen auf. Die Einbindung dieser externen Referenzen gewährleistet, dass die im VSPV-Standard 101 beschriebenen Datenstrukturen, Prozesse und Konventionen sowohl den aktuellen rechtlichen Anforderungen als auch dem Stand

der Technik entsprechen und eine hohe Interoperabilität mit bestehenden und zukünftigen Systemlandschaften ermöglichen.

#### **Zentrale Referenzdokumente sind insbesondere:**

- **CEN/TS 16614 (NeTEx):**

Der europäische Standard NeTEx (Network Timetable Exchange) definiert ein umfassendes Datenmodell und ein XML-basiertes Austauschformat für Netz-, Fahrplan- und Tarifdaten im öffentlichen Verkehr. NeTEx ist die technische Grundlage für die strukturierte und interoperable Bereitstellung von ÖPNV-Daten in Europa und bildet das Fundament für die Modellierung im VSPV-Standard 101.

- **VDV-Schrift 462:**

Die VDV 462 konkretisiert die Anwendung von NeTEx im deutschen ÖPNV-Kontext und legt Profile, Konventionen und technische Details für den Austausch von Linien- und Bedarfsverkehren fest. Sie ist der zentrale Bezugsrahmen für die Struktur, Terminologie und die technischen Konventionen des VSPV-Standards 101.

- **SDGV-Projektdokumentation:**

Die im Rahmen des SDGV-Projekts erarbeiteten fachlichen Anforderungen, Integrationsstufen und Use Cases für Gelegenheitsverkehre in Nordrhein-Westfalen bilden die inhaltliche Grundlage für die Erweiterungen des VSPV-Standards 101. Die dort entwickelten fachlichen Datenmodelle und Integrationsstufen werden im technischen Modell konsequent umgesetzt.

- **Mobilitätsdatenverordnung (MDV):**

Die Mobilitätsdatenverordnung regelt die Verpflichtung zur Bereitstellung von Mobilitätsdaten für den Nationalen Zugangspunkt (NAP) und legt Anforderungen an Datenformate, Aktualität und Umfang der zu übermittelnden Informationen fest. Der VSPV-Standard 101 trägt dazu bei, die daraus resultierenden gesetzlichen Vorgaben für Gelegenheitsverkehre und Taxiangebote zu erfüllen.

- **Weitere relevante Standards und Leitfäden:**

Bei Bedarf werden Querverweise auf weitere Standards wie VDV 431 (TRIAS), GTFS-flex, SIRI oder spezifische technische Leitfäden gegeben, sofern diese für die Umsetzung einzelner Aspekte (z. B. Echtzeitdaten, Buchungsintegration) heranzuziehen sind.

#### **Anwendung und Priorisierung:**

Die genannten Referenzdokumente sind bei der Implementierung und Auslegung des VSPV-Standards 101 stets zu berücksichtigen. Bei Auslegungsfragen oder Widersprüchen zwischen diesem Standard und den externen Dokumenten ist der VSPV-Standard 101 für Gelegenheitsverkehre in Nordrhein-Westfalen maßgeblich. Weitergehende Anforderungen oder Empfehlungen aus den externen Dokumenten

sind ergänzend heranzuziehen, sofern sie die Interoperabilität und Zukunftsfähigkeit der Systeme unterstützen.

### 1.4.3 Konventionen bei Begriffen, Schreibweise und Beispielen

Die Verständlichkeit, Nachvollziehbarkeit und technische Umsetzbarkeit des VSPV-Standards 101 wird durch die konsequente Anwendung klar definierter Begriffs-, Schreib- und Darstellungsregeln unterstützt. Diese Konventionen orientieren sich an den Gepflogenheiten der VDV-Schrift 462 und des NeTeX-Standards und dienen dazu, Missverständnisse zu vermeiden und die Konsistenz zwischen Dokumentation, Mappingtabellen und Beispieldateien sicherzustellen.

#### **Begriffe und Terminologie:**

- Für alle Datenstrukturen, Objekte und Attribute werden die im NeTeX-Standard und der VDV 462 definierten englischen Originalbegriffe verwendet (z. B. FlexibleStopPlace, BookingProcess, ServiceJourney).
- Deutsche Begriffe werden zur erläuternden Beschreibung, Kommentierung und in Überschriften genutzt, sofern keine eindeutige englische Entsprechung existiert oder zur besseren Verständlichkeit im deutschen Kontext.
- Fachbegriffe aus der SDGV-Projektdokumentation werden übernommen und, sofern erforderlich, im Glossar erläutert.

#### **Schreibweise und Formatierung:**

- Elemente und Bezeichnungen aus NeTeX werden in **Aptos** und in der originalen CamelCase-Schreibweise dargestellt (z. B. FlexibleStopPlace, TarifZone).
- XML-Elemente und Codebeispiele werden als eingerückte Codeblöcke dargestellt, um die technische Struktur und Hierarchie klar zu visualisieren.
- Pfadangaben und Mappingtabellen folgen der Notation des NeTeX-XSD und geben stets den vollständigen Pfad zum jeweiligen Element an.

#### **Beispiele und Illustrationen:**

- Für zentrale Anwendungsfälle werden validierte XML-Beispiele bereitgestellt, die die praktische Umsetzung der beschriebenen Anforderungen illustrieren. Diese Beispiele enthalten nur die für das jeweilige Thema relevanten Ausschnitte, um die Lesbarkeit zu erhöhen.
- Mappingtabellen und Beispiel-XMLs sind eng aufeinander abgestimmt und werden regelmäßig aktualisiert, um Änderungen im Standard oder in den zugrundeliegenden Normen zeitnah zu berücksichtigen.
- In den Beispielen werden Platzhalter (z. B. VSPV:StopPlace:NRW-1001) verwendet, die in der Praxis durch die jeweils gültigen Identifikatoren zu ersetzen sind.

## **Verweise und Querverbindungen:**

- Innerhalb des Dokuments wird durchgängig auf relevante Kapitel, Tabellen, Beispiele und externe Referenzen verwiesen, um die Navigation und das Verständnis zu erleichtern.
- Begriffe, die im Glossar erläutert werden, sind entsprechend gekennzeichnet.

Diese Konventionen sind für die Erstellung, Pflege und Anwendung des VSPV-Standards 101 verbindlich. Sie gewährleisten eine konsistente Dokumentation, erleichtern die technische Implementierung und fördern die Interoperabilität zwischen unterschiedlichen Systemen und Akteuren.

### **1.4.4 Hinweise zu Implementationshinweisen und Mappingtabellen**

Die technische Umsetzung und fortlaufende Pflege des VSPV-Standards 101 wird durch gezielte Implementationshinweise sowie durch den systematischen Einsatz von Mappingtabellen unterstützt. Beide Instrumente dienen dazu, die Übertragbarkeit der fachlichen Anforderungen in technische Strukturen zu gewährleisten und die praktische Anwendung des Standards zu erleichtern.

#### **Implementationshinweise:**

- Innerhalb des Handbuchs sind spezielle Abschnitte als „Hinweis“ oder „Implementationshinweis“ gekennzeichnet. Diese Abschnitte enthalten praxisorientierte Empfehlungen, Erläuterungen zu typischen Stolpersteinen, Hinweise auf bewährte Vorgehensweisen oder auf Besonderheiten bei der Anwendung bestimmter Datenstrukturen.
- Implementationshinweise sind nicht Teil der verbindlichen Spezifikation, sondern dienen als Hilfestellung für Entwickler, Systemintegratoren und Datenverantwortliche. Sie sollen die technische Umsetzung beschleunigen, Fehlerquellen minimieren und die Interoperabilität fördern.
- Wo erforderlich, verweisen Implementationshinweise direkt auf relevante Mappingtabellen, Beispiel-XMLs oder externe Dokumente.

#### **Mappingtabellen:**

- Die Mappingtabellen sind ein zentrales Werkzeug zur Übertragung der im VSPV-Standard 101 beschriebenen fachlichen Anforderungen auf die technische Ebene des NeTeX-Datenmodells. Sie dokumentieren für jedes relevante Datenelement die Zuordnung zum entsprechenden NeTeX-Element, inklusive Pfadangabe, Datentyp, Kardinalität und etwaigen Besonderheiten.
- Die Tabellen enthalten zudem Hinweise zu Pflichtfeldern, zulässigen Wertebereichen, empfohlenen Erweiterungen (z. B. Nutzung von keyList oder Extensions) sowie zu spezifischen Validierungsregeln.
- Mappingtabellen werden als Excel- oder CSV-Datei bereitgestellt und sind integraler Bestandteil des Standards. Sie werden regelmäßig gepflegt und bei

Weiterentwicklungen des Standards oder der zugrundeliegenden Normen aktualisiert.

- Im Falle von Abweichungen oder Unklarheiten zwischen Handbuch, Beispielen und Mappingtabellen gilt die Mappingtabelle als maßgeblich für die technische Implementierung.

#### **Zusammenspiel und Nutzung:**

- Die enge Verzahnung von Handbuch, Implementationshinweisen, Mappingtabellen und Beispiel-XMLs stellt sicher, dass der VSPV-Standard 101 sowohl konzeptionell klar als auch technisch eindeutig und praxistauglich ist.
- Es wird empfohlen, bei der Implementierung stets die aktuelle Version der Mappingtabellen und der Beispiel-XMLs zu verwenden und bei Unsicherheiten Rücksprache mit dem Standardisierungsgremium oder den Herausgebern des VSPV-Standards 101 zu halten.

#### **1.4.5 Umgang mit Erweiterungen und individuellen Anpassungen**

Der VSPV-Standard 101 ist darauf ausgelegt, eine einheitliche und interoperable Grundlage für den Datenaustausch im Bereich der Gelegenheitsverkehre zu schaffen. Gleichzeitig ist anerkannt, dass in der Praxis spezifische betriebliche Anforderungen oder regionale Besonderheiten auftreten können, die über den im Standard vorgesehenen Funktionsumfang hinausgehen. Für diese Fälle sieht der Standard definierte Mechanismen zur Erweiterung und individuellen Anpassung vor, die die Kompatibilität und Zukunftsfähigkeit der Datenstrukturen sicherstellen sollen.

#### **Mechanismen für Erweiterungen:**

- **KeyValue-Paare in keyList:**

Für die Aufnahme zusätzlicher, nicht im Standard definierter Attribute empfiehlt der VSPV-Standard 101 die Nutzung von KeyValue-Paaren innerhalb des Elements keyList. Diese Methode eignet sich insbesondere für die Übertragung einfacher Zusatzinformationen, wie etwa betriebsinterne Kennungen, besondere Buchungsregeln oder regionale Einschränkungen.

- **Extensions-Bereich:**

Für komplexere oder strukturierte Erweiterungen, die nicht durch einzelne KeyValue-Paare abbildbar sind, steht der Extensions-Bereich zur Verfügung. Hier können auch umfangreichere proprietäre Datenstrukturen als XML-Elemente eingefügt werden, beispielsweise für spezielle Tarifmodelle, dynamische Preisinformationen oder zusätzliche Buchungsfunktionen.

### **Vorgehen bei individuellen Anpassungen:**

- Alle Erweiterungen und individuellen Anpassungen sind in der jeweiligen Schnittstellendokumentation eindeutig zu beschreiben und zwischen den beteiligten Systempartnern verbindlich abzustimmen.
- Es ist sicherzustellen, dass Erweiterungen die Verarbeitung durch Standard-NeTeX-Parser nicht beeinträchtigen. Die Einhaltung der XSD-Struktur und die Validierbarkeit der Datenlieferungen bleiben zwingende Voraussetzung.
- Erweiterungen sollten so gestaltet sein, dass sie perspektivisch in zukünftige Versionen des VSPV-Standards 101 oder in weiterentwickelte nationale bzw. internationale Standards übernommen werden können. Eine enge Abstimmung mit dem Standardisierungsgremium wird empfohlen.

### **Dokumentation und Transparenz:**

- Erweiterungen und individuelle Anpassungen sind in den Mappingtabellen und in der Schnittstellenbeschreibung explizit zu kennzeichnen und zu dokumentieren.
- Bei der Weitergabe von Daten an Dritte, insbesondere an zentrale Auskunfts- und Buchungssysteme oder den Nationalen Zugangspunkt, ist sicherzustellen, dass die verwendeten Erweiterungen nachvollziehbar und – soweit möglich – dokumentiert sind.

### **Zielsetzung:**

- Der VSPV-Standard 101 verfolgt das Ziel, die größtmögliche Interoperabilität und Wiederverwendbarkeit der Datenstrukturen zu gewährleisten, ohne die notwendige Flexibilität für betriebliche oder regionale Besonderheiten einzuschränken.
- Individuelle Anpassungen sollen stets im Sinne der Standardkonformität und Zukunftsfähigkeit erfolgen und möglichst in den Standard zurückgeführt werden, sobald sie sich als allgemein relevant erweisen.

## **2 Grundlagenmodellierung**

Die Grundlagenmodellierung bildet das methodische und technische Fundament des VSPV-Standards 101. Sie legt die Prinzipien, Strukturen und Konventionen fest, nach denen sämtliche Daten zu Gelegenheitsverkehren und Taxiverkehren modelliert, bereitgestellt und ausgetauscht werden. Die Ausgestaltung orientiert sich konsequent an der VDV-Schrift 462, die als deutsche Umsetzung des europäischen NeTeX-Standards (CEN/TS 16614) etabliert ist und sich im Bereich des Linienverkehrs als interoperabler und praxistauglicher Rahmen bewährt hat.

Mit der Ausweitung des ÖPNV-Angebots um flexible, flächengebundene On-Demand- und Taxiverkehre entstehen neue Anforderungen an die Datenmodellierung: Dynamische Bedienegebiete, variable Fahrpläne, unterschiedliche Buchungs- und Dispositionsregeln sowie die Integration von Echtzeit- und Buchungsinformationen erfordern eine

Erweiterung und Präzisierung der bisherigen Modellierungsansätze. Der VSPV-Standard 101 greift diese Herausforderungen auf und entwickelt die Grundlagenmodellierung der VDV 462 gezielt weiter, um die spezifischen Bedürfnisse von Gelegenheitsverkehren abzubilden und gleichzeitig die Kompatibilität mit bestehenden Systemen und Prozessen zu wahren.

Kapitel 2 beschreibt zunächst die Zielsetzung und die Rolle des NeTEx-Standards als methodisches Fundament. Es erläutert die zentralen Modellierungsprinzipien und Grundelemente, auf denen das technische Datenmodell für Gelegenheitsverkehre aufbaut. Darüber hinaus werden die für den VSPV-Standard 101 maßgeblichen technischen Konventionen, Erweiterungsmechanismen und Validierungsgrundlagen dargestellt. Abschließend erfolgt eine Einordnung und Abgrenzung zu weiteren relevanten Standards und Austauschformaten.

Ziel dieses Kapitels ist es, allen Anwendern – von Systemherstellern über Verkehrsunternehmen bis zu Datenintegratoren – ein einheitliches Verständnis der Modellierungsgrundlagen zu vermitteln und die Basis für die darauf aufbauenden, spezifischen Modellierungsabschnitte zu schaffen.

#### Zielsetzung und Rolle von NeTEx

Die Grundlage des VSPV-Standards 101 bildet der europäische Standard NeTEx (Network Timetable Exchange, CEN/TS 16614), der als umfassendes, objektorientiertes Datenmodell für den Austausch von Netz-, Fahrplan- und Tarifdaten im öffentlichen Verkehr entwickelt wurde. Die Zielsetzung von NeTEx ist es, einen europaweit einheitlichen, interoperablen Rahmen für die strukturierte Bereitstellung und den Austausch von Mobilitätsdaten zu schaffen, der sowohl den Anforderungen klassischer Linienverkehre als auch moderner, flexibler Bedienformen gerecht wird.

Im Kontext des VSPV-Standards 101 übernimmt NeTEx eine zentrale Rolle als methodisches und technisches Fundament für die Modellierung sämtlicher relevanter Daten zu Gelegenheitsverkehren. Die Entscheidung für NeTEx als Basis beruht auf mehreren fachlichen und strategischen Überlegungen:

- **Interoperabilität und Standardkonformität:**

NeTEx ist als CEN-Standard europaweit anerkannt und wird in zahlreichen nationalen und regionalen Projekten eingesetzt. Die Nutzung von NeTEx gewährleistet die Kompatibilität mit bestehenden und zukünftigen Systemlandschaften – insbesondere mit den in Deutschland etablierten VDV-462-Profilen – und ermöglicht den Austausch von Mobilitätsdaten über System- und Ländergrenzen hinweg.

- **Flexibilität und Erweiterbarkeit:**

Das NeTEx-Datenmodell ist modular und objektorientiert aufgebaut. Es erlaubt die Abbildung unterschiedlichster Betriebsformen, von klassischen Linienverkehren bis hin zu flexiblen On-Demand-Angeboten und Taxiverkehren. Durch die Möglichkeit, bestehende Strukturen zu erweitern und um spezifische Attribute

oder Mechanismen zu ergänzen, können auch neue oder bislang nicht standardisierte Anforderungen integriert werden, ohne die Kompatibilität zum Gesamtsystem zu gefährden.

- **Zukunftsfähigkeit und Harmonisierung:**

Die kontinuierliche Weiterentwicklung von NeTEx durch europäische und nationale Standardisierungsgremien stellt sicher, dass der Standard auch zukünftigen Anforderungen an Mobilitätsdaten und deren Austausch gerecht wird. Die Orientierung an NeTEx erleichtert zudem die Harmonisierung mit anderen europäischen Initiativen und die Integration in nationale Zugangspunkte (z. B. den Nationalen Zugangspunkt für Mobilitätsdaten in Deutschland).

- **Abdeckung der gesamten Mobilitätskette:**

NeTEx ist darauf ausgelegt, sämtliche relevanten Aspekte der Mobilitätskette – von der Netzbeschreibung über Fahrpläne und Tarife bis hin zu Ressourcen, Betriebszeiten und Echtzeitinformationen – in einem konsistenten Datenmodell abzubilden. Für die Integration von Gelegenheitsverkehren bedeutet dies, dass alle für Fahrgastinformation, Buchung und Betrieb erforderlichen Datenstrukturen in einem durchgängigen Rahmen modelliert werden können.

Im VSPV-Standard 101 wird NeTEx nicht nur als technisches Austauschformat genutzt, sondern als methodisches Leitbild für die gesamte Modellierungskonzeption übernommen. Die Profile, Konventionen und Erweiterungsmechanismen der VDV-Schrift 462 werden gezielt weiterentwickelt, um die spezifischen Anforderungen von Gelegenheitsverkehren und Taxiverkehren zu adressieren und gleichzeitig die Anschlussfähigkeit an bestehende und künftige Standards zu sichern.

Damit bildet NeTEx die unverzichtbare Grundlage für eine zukunftsfähige, interoperable und erweiterbare Datenhaltung und -austausch im Bereich der Gelegenheitsverkehre in Nordrhein-Westfalen.

Der europäische Standard NeTEx (Network Timetable Exchange, CEN/TS 16614) wurde entwickelt, um den Austausch von Netz-, Fahrplan- und Tarifdaten im öffentlichen Verkehr europaweit zu harmonisieren. Ziel ist es, eine gemeinsame, interoperable Datenbasis zu schaffen, die die unterschiedlichen Anforderungen der nationalen und regionalen Mobilitätssysteme abdeckt und zugleich die Grundlage für eine grenzüberschreitende Integration von Mobilitätsangeboten bildet. NeTEx ist das Ergebnis langjähriger Standardisierungsarbeit unter Beteiligung zahlreicher europäischer Verkehrsunternehmen, Systemhersteller und Verbände. Er ist als CEN Technical Specification veröffentlicht und bildet damit die formale Grundlage für die nationale und regionale Ausgestaltung technischer Austauschformate im öffentlichen Verkehr.

Die Bedeutung von NeTEx liegt vor allem in seiner Fähigkeit, die Vielfalt der europäischen Mobilitätslandschaften in einem einzigen, flexiblen Datenmodell abzubilden. Während in einzelnen Ländern unterschiedliche technische Systeme, Datenformate und betriebliche Anforderungen bestehen, ermöglicht NeTEx eine standardisierte Beschreibung aller relevanten Aspekte des öffentlichen Verkehrs – von der Netzstruktur über Fahrpläne und

Tarife bis hin zu Ressourcen, Betriebszeiten und Echtzeitinformationen. Damit ist NeTEx nicht nur ein Austauschformat, sondern ein umfassendes Referenzmodell für die digitale Abbildung der gesamten Mobilitätskette.

Ein zentrales Merkmal von NeTEx ist seine Modularität und Objektorientierung. Das Datenmodell ist in logisch abgegrenzte Bereiche – sogenannte Frames – unterteilt, die jeweils spezifische Aspekte des Verkehrs abbilden. So gibt es beispielsweise eigene Frames für das Liniennetz (NetworkFrame), für Fahrplandaten (TimetableFrame), für Tarifinformationen (FareFrame) und für Ressourcen (ResourceFrame). Innerhalb der Frames werden die einzelnen Objekte durch eindeutige Identifikatoren, Versionierungen und Gültigkeitszeiträume beschrieben, was eine präzise Steuerung von Aktualisierungen, Änderungen und historischen Datenständen ermöglicht.

Ein weiteres wesentliches Prinzip von NeTEx ist die strikte Trennung und zugleich die klare Verknüpfbarkeit von Netz-, Fahrplan- und Tarifdaten. Dies erlaubt es, komplexe Zusammenhänge – etwa zwischen einem dynamisch veränderlichen Bediengebiet, den zugehörigen Fahrten und den jeweils gültigen Tarifregeln – konsistent und nachvollziehbar abzubilden. Die Referenzierung zwischen den Objekten erfolgt über stabile, persistente IDs, die eine eindeutige Zuordnung und Wiederverwendbarkeit der Daten über System- und Organisationsgrenzen hinweg gewährleisten.

Ein besonderer Vorteil von NeTEx liegt in seiner Fähigkeit, mit der zunehmenden Komplexität und Dynamik moderner Mobilitätsangebote Schritt zu halten. Während frühere Austauschformate oft auf den klassischen Linienverkehr mit festen Haltestellen und Fahrplänen ausgerichtet waren, berücksichtigt NeTEx von Beginn an die Notwendigkeit, auch flexible und bedarfsorientierte Verkehrsformen – wie On-Demand-Angebote, Rufbusse oder Taxiverkehre – systematisch abzubilden. Das Datenmodell sieht hierfür spezielle Objekttypen wie FlexibleStopPlace für dynamische Haltepunkte oder FlexibleServiceJourney für unscharfe, nicht liniengebundene Fahrten vor. Diese Erweiterungen ermöglichen es, die betriebliche Realität von Gelegenheitsverkehren mit ihren variablen Bediengebieten, unterschiedlichen Buchungs- und Dispositionsregeln sowie wechselnden Betriebszeiten präzise und standardkonform zu modellieren.

Für den VSPV-Standard 101 ist die Übernahme von NeTEx als methodisches und technisches Fundament deshalb von zentraler Bedeutung. Die in NeTEx angelegte Flexibilität und Erweiterbarkeit erlaubt es, die spezifischen Anforderungen der Gelegenheitsverkehre in Nordrhein-Westfalen – wie sie in der SDGV-Projektdokumentation und den Vorgaben der Mobilitätsdatenverordnung beschrieben sind – systematisch und zukunftssicher umzusetzen. Dies betrifft sowohl die Abbildung dynamischer Bediengebiete und unscharfer Fahrpläne als auch die Integration von Buchungsprozessen, Tarifmodellen und Echtzeitinformationen.

Ein weiterer wesentlicher Aspekt ist die Interoperabilität, die durch die Nutzung von NeTEx erreicht wird. Da NeTEx als europaweiter Standard von zahlreichen Systemherstellern, Verkehrsunternehmen und Plattformbetreibern unterstützt wird, ist sichergestellt, dass die mit dem VSPV-Standard 101 erzeugten Daten nicht nur innerhalb Nordrhein-Westfalens, sondern auch bundesweit und international verarbeitet und genutzt werden

können. Dies ist insbesondere für die Anbindung an den Nationalen Zugangspunkt für Mobilitätsdaten (NAP) und für die Integration in nationale und europäische MaaS-Plattformen von großer Bedeutung.

Darüber hinaus bietet NeTEx durch seine ausgefeilten Mechanismen zur Versionierung und Gültigkeitspflege die Möglichkeit, Änderungen im Netz, bei den Fahrplänen oder bei den Tarifbedingungen nachvollziehbar und transparent zu verwalten. Dies ist gerade für Gelegenheitsverkehre mit ihren häufigen Anpassungen und betrieblichen Besonderheiten ein entscheidender Vorteil. Die eindeutige Identifikation und Historisierung aller relevanten Objekte erleichtert die Pflege der Daten und ermöglicht eine konsistente und aktuelle Fahrgastinformation sowie eine zuverlässige Integration in Buchungs- und Vertriebssysteme.

Die Entscheidung, NeTEx als technisches Rückgrat für den VSPV-Standard 101 zu nutzen, ist auch vor dem Hintergrund der deutschen Systemlandschaft und der bestehenden IT-Infrastruktur im ÖPNV zu sehen. Mit der VDV-Schrift 462 existiert bereits eine umfassende nationale Profilspezifikation, die die Anwendung und Ausgestaltung von NeTEx für den deutschen Markt regelt. Sie legt verbindliche Profile, Konventionen und technische Details für den Austausch von Linien- und Bedarfsverkehren fest und wird von allen maßgeblichen Systemherstellern und Verkehrsunternehmen in Deutschland unterstützt. Der VSPV-Standard 101 knüpft an diese etablierten Strukturen an und erweitert sie gezielt um die für Gelegenheitsverkehre und Taxiverkehre notwendigen Funktionalitäten.

Die enge Anlehnung an die VDV 462 stellt sicher, dass bestehende Prozesse, Werkzeuge und Validierungsmechanismen weiterverwendet werden können und die Integration von Gelegenheitsverkehren in bestehende Auskunft-, Buchungs- und Vertriebssysteme ohne grundlegende Systemumstellungen möglich ist. Gleichzeitig wird durch die gezielte Erweiterung des NeTEx-Modells – etwa durch die Nutzung von Extensions, keyList-Mechanismen oder die Definition zusätzlicher Profile – die notwendige Flexibilität geschaffen, um auch zukünftige betriebliche und regulatorische Anforderungen abbilden zu können.

NeTEx bietet zudem umfangreiche Möglichkeiten zur Dokumentation und Beschreibung von betrieblichen Prozessen, Dispositionsregeln und Schnittstellen zu anderen Systemen. Durch die strukturierte Modellierung von Buchungsprozessen, Kapazitätsgrenzen, Vorlaufzeiten, Ausschlussregeln und weiteren betrieblichen Parametern können die Besonderheiten der Gelegenheitsverkehre detailliert und nachvollziehbar abgebildet werden. Dies erleichtert nicht nur die technische Implementierung, sondern auch die Abstimmung und Zusammenarbeit zwischen den beteiligten Akteuren – von Verkehrsunternehmen über Systemhersteller bis hin zu Aufgabenträgern und Plattformbetreibern.

Schließlich trägt die Nutzung von NeTEx auch zur Rechtssicherheit und zur Erfüllung gesetzlicher Vorgaben bei. Die Mobilitätsdatenverordnung und die Anforderungen an den Nationalen Zugangspunkt verlangen eine standardisierte, maschinenlesbare und interoperable Bereitstellung aller relevanten Mobilitätsdaten – einschließlich der Daten zu Gelegenheitsverkehren und Taxiverkehren. Mit NeTEx und den darauf aufbauenden

VDV- und VSPV-Profilen steht ein erprobtes und anerkanntes Instrumentarium zur Verfügung, das die Einhaltung dieser Vorgaben zuverlässig unterstützt.

Zusammenfassend lässt sich festhalten, dass NeTEx im Rahmen des VSPV-Standards 101 weit mehr ist als ein technisches Austauschformat. Es bildet das methodische Rückgrat für die strukturierte, nachvollziehbare und zukunftsfähige Modellierung sämtlicher Daten zu Gelegenheitsverkehren und Taxiverkehren. Die Wahl von NeTEx als Basis gewährleistet, dass die im VSPV-Standard 101 beschriebenen Datenstrukturen sowohl mit den bestehenden deutschen und europäischen Systemlandschaften als auch mit den Anforderungen moderner, digitaler Mobilitätsplattformen kompatibel sind.

Die konsequente Orientierung an NeTEx ermöglicht es, die Vielfalt und Dynamik der Gelegenheitsverkehre in Nordrhein-Westfalen flexibel und präzise abzubilden, ohne dabei die Interoperabilität, die Datenhoheit der Betreiber oder die Anschlussfähigkeit an nationale und internationale Entwicklungen aus dem Blick zu verlieren. Durch die Nutzung etablierter Mechanismen wie Frames, Versionierung, Validitätszeiträume und Erweiterungsoptionen (z. B. Extensions und keyList) wird sichergestellt, dass auch künftige betriebliche und regulatorische Anforderungen effizient und standardkonform umgesetzt werden können.

Mit dieser methodischen und technischen Grundlage ist der VSPV-Standard 101 in der Lage, eine durchgängige, diskriminierungsfreie und medienbruchfreie Servicekette von der Fahrgastinformation über die Buchung bis hin zur Abrechnung und zum Reporting zu unterstützen. Die Rolle von NeTEx als Fundament für die Grundlagenmodellierung ist damit zentral für den nachhaltigen Erfolg der Digitalisierung und Standardisierung von Gelegenheitsverkehren im ÖPNV Nordrhein-Westfalens und darüber hinaus.

## 2.1 Zentrale Modellierungsprinzipien und Grundelemente

Ein zentrales Grundprinzip des NeTEx-Standards – und damit auch des VSPV-Standards 101 – ist die konsequente Modularisierung der Datenstruktur. Die Komplexität moderner Mobilitätsangebote und die Vielzahl der zu modellierenden Informationen machen es erforderlich, die Daten logisch und technisch in klar abgegrenzte, wiederverwendbare Module zu gliedern. Im NeTEx-Datenmodell werden diese Module als sogenannte **Frames** bezeichnet.

**Frames** sind eigenständige Container für zusammengehörige Datenobjekte eines bestimmten Themenbereichs. Sie strukturieren das Gesamtdatenmodell in logisch abgegrenzte Bereiche, die unabhängig voneinander gepflegt, ausgetauscht und versioniert werden können. Die wichtigsten Frame-Typen im Kontext des VSPV-Standards 101 sind:

- **NetworkFrame:** Enthält die grundlegenden Netzstrukturdaten, wie Haltestellen, Haltepunkte, Betriebshöfe, Fußwege, Strecken, Linienverläufe und Netzknoten.
- **ServiceFrame:** Modelliert die betrieblichen und fahrgastrelevanten Aspekte des Verkehrsangebots, insbesondere Linien, Fahrten, flexible Bedienformen, Bedienegebiete und zugehörige Dispositionsregeln.

- **TimetableFrame:** Beinhaltet die eigentlichen Fahrplandaten, also die zeitliche Abfolge von Fahrten, Umläufen, Kursen, Fahrzeit- und Haltezeitinformationen sowie Betriebskalender und Tagesarten.
- **FareFrame:** (optional, aber relevant für Gelegenheitsverkehre mit variablen Tarifen) Enthält Tarifdaten, Preisregeln, Tarifzonen und Zuschläge.
- **ResourceFrame:** (optional) Dient der Modellierung von Ressourcen wie Fahrzeugen, Personal, Betriebsmitteln.

Jeder Frame ist in sich konsistent und kann unabhängig von anderen Frames gepflegt und aktualisiert werden. Dadurch wird die **Wartbarkeit** des Gesamtsystems erheblich verbessert. Änderungen an einzelnen Aspekten – etwa die Anpassung eines Bedienegebiets oder die Einführung eines neuen Tarifs – können gezielt in dem jeweils betroffenen Frame vorgenommen werden, ohne dass das gesamte Datenmodell neu aufgebaut werden muss.

Ein weiterer Vorteil des Frame-Konzepts ist die **Wiederverwendbarkeit:** Ein Frame, der beispielsweise ein bestimmtes Bedienegebiet oder eine Tarifzone beschreibt, kann in mehreren Kontexten referenziert und genutzt werden. Dies fördert die Konsistenz der Daten und reduziert Redundanzen. Frames sind zudem die zentrale Einheit für den Datenaustausch zwischen Systemen: Sie können einzeln exportiert, importiert und versioniert werden, was die Integration in bestehende Systemlandschaften und die Zusammenarbeit zwischen verschiedenen Akteuren erleichtert.

Im VSPV-Standard 101 wird das Frame-Prinzip genutzt, um die spezifischen Anforderungen von Gelegenheitsverkehren – etwa die Abbildung dynamischer Bedienegebiete oder die Integration von Taxitarifen – klar von den klassischen Linienverkehrsdaten zu trennen und dennoch eine konsistente Gesamtsicht zu gewährleisten. Die Modularisierung bildet somit das Rückgrat der technischen Architektur und ist Voraussetzung für die Flexibilität, Skalierbarkeit und Zukunftsfähigkeit des Datenmodells.

Zur Vermeidung von Unschärfen werden die zentralen fachlichen Entitäten im Folgenden klar definiert: Eine Fahrt ist die Durchführung einer Beförderung von einem Start- zu einem Zielpunkt (gegebenenfalls mit Zwischenhalten) im Gelegenheitsverkehr; sie entspricht im NeTEx-Datenmodell einem ServiceJourney. Eine Buchung bezeichnet den Vorgang bzw. Datensatz, mit dem ein Fahrgast eine solche Fahrt anfragt oder reserviert – inklusive aller relevanten Angaben wie Abhol- und Zielort, Uhrzeit sowie Fahrgast- und ggf. Zahlungsinformationen. Ein Fahrzeug ist das eingesetzte Transportmittel (z. B. Taxi, Mietwagen oder Kleinbus) mit seinen spezifischen Eigenschaften wie Kapazität, Ausstattung und Verfügbarkeit, das zur Durchführung von Fahrten bereitgestellt wird. Ein Haltepunkt ist ein Ein- oder Ausstiegspunkt für Fahrgäste und kann entweder eine physische Haltestelle des ÖPNV oder ein virtueller Haltepunkt (wie eine definierte Adresse oder Koordinate innerhalb eines Bedienegebiets) sein. Als Kostenträger wird eine Institution oder Stelle bezeichnet, die die Kosten einer Fahrt übernimmt oder bezuschusst (z. B. eine Krankenkasse bei verordneten Krankenfahrten oder ein öffentlicher Aufgabenträger im Rahmen besonderer Verkehrsangebote). Die klare

Trennung und Definition dieser Entitäten gewährleistet eine konsistente und verständliche Modellierung im gesamten Standard.

Ein weiteres zentrales Prinzip des NeTEx-Standards und damit auch des VSPV-Standards 101 ist die **Objektorientierung**. Das Datenmodell basiert auf der Definition klar abgegrenzter, typisierter Objekte, die jeweils spezifische Eigenschaften und Beziehungen besitzen. Jedes Objekt repräsentiert eine reale oder betriebliche Entität des öffentlichen Verkehrs – etwa eine Haltestelle, eine Fahrt, ein Bediengebiet, eine Tarifzone oder einen Buchungsprozess.

#### **Objekthierarchien und Vererbung:**

Die Objektorientierung zeigt sich insbesondere in der Hierarchie und Vererbung von Eigenschaften. Oberklassen wie „Place“ oder „Service“ definieren gemeinsame Attribute und Methoden, die von spezialisierten Unterklassen wie „StopPlace“, „FlexibleStopPlace“ oder „ServiceJourney“ übernommen und bei Bedarf erweitert werden. Dadurch lassen sich sowohl generische als auch spezifische Anforderungen effizient und konsistent abbilden. Zum Beispiel erbt das Objekt FlexibleStopPlace alle grundlegenden Eigenschaften einer Haltestelle (StopPlace), ergänzt diese aber um zusätzliche Attribute wie Polygone für Bediengebiete oder spezielle Dispositionsregeln.

#### **Referenzierung und Beziehungen zwischen Objekten:**

Die Objekte im NeTEx-Modell stehen in vielfältigen Beziehungen zueinander. Diese Beziehungen werden durch explizite Referenzen abgebildet, die auf eindeutigen, persistenten Identifikatoren (IDs) basieren. So verweist beispielsweise eine Fahrt (ServiceJourney) auf die zugehörige Linie (Line), die bedienten Haltestellen (StopPlace oder FlexibleStopPlace), den Betriebskalender (OperatingDay) und ggf. auf Tarifinformationen oder Buchungsprozesse. Durch diese Referenzierung entsteht ein engmaschiges Netz von Beziehungen, das die komplexen Zusammenhänge im öffentlichen Verkehr präzise widerspiegelt.

#### **Vorteile für Wiederverwendbarkeit und Konsistenz:**

Die konsequente Nutzung von Referenzen und Objektorientierung bietet erhebliche Vorteile für die Wiederverwendbarkeit und Konsistenz der Daten. Ein einmal definiertes Objekt – etwa ein Bediengebiet oder eine Tarifzone – kann in verschiedenen Kontexten referenziert werden, ohne dass die Informationen mehrfach vorgehalten werden müssen. Änderungen an einem Objekt wirken sich automatisch auf alle referenzierenden Strukturen aus, was die Datenpflege vereinfacht und Fehlerquellen minimiert.

#### **Beispielhafte Anwendung:**

Ein FlexibleServiceJourney (flexible Fahrt) kann auf mehrere FlexibleStopPlace-Objekte (dynamische Haltepunkte) verweisen, die wiederum in unterschiedlichen Bediengebieten verwendet werden. Ein BookingProcess-Objekt kann von verschiedenen

Fahrten referenziert werden, um einheitliche Buchungsregeln für unterschiedliche Angebote zu gewährleisten.

**Fazit:**

Die Objektorientierung und die explizite Referenzierung zwischen den Objekten sind entscheidend für die Skalierbarkeit, Wartbarkeit und Erweiterbarkeit des Datenmodells. Sie ermöglichen es, auch komplexe und dynamische Verkehrsangebote wie Gelegenheitsverkehre und Taxiverkehre strukturiert, konsistent und interoperabel abzubilden.

Zur Integration von bedarfsorientierten Flächenverkehren nutzt der VSPV-Standard 101 zusätzliche NeTEx-Rahmen und -Objekte. So kommt ein StopPlaceFrame zum Einsatz, in dem Haltepunkte – einschließlich virtueller Einstiegspunkte und adressbasierter Haltepunkte – verwaltet und strukturiert hinterlegt werden können. Flexible geografische Bedienegebiete werden über TopographicPlace-Objekte mit Polygon-Geometrie abgebildet, was die Definition polygonaler Bedienegebietsgrenzen innerhalb des Datenmodells ermöglicht. Darüber hinaus wird für die spezifischen Elemente des Bedarfsverkehrs ein eigener DemandResponseFrame vorgesehen. In diesem können alle für On-Demand-Angebote relevanten Objekte zusammengefasst werden (z. B. Buchungsregeln, variable Betriebszeiten, Pooling-Parameter), um sie sauber vom klassischen Linienverkehr zu trennen. Durch die Einführung dieser Strukturen – in Anlehnung an die SDGV-Definitionen – können flächenhafte Verkehre vollständig im Rahmen des NeTEx/VDV-462-Profiles abgebildet und zwischen allen Beteiligten interoperabel ausgetauscht werden.

Ein zentrales Element der Grundlagenmodellierung ist die eindeutige Identifikation, Versionierung und Gültigkeitssteuerung sämtlicher Objekte im Datenmodell. Jedes Objekt – sei es eine Haltestelle, ein Bedienegebiet, eine Fahrt oder ein Tarif – erhält einen persistenten, systemweit eindeutigen Identifikator (ID). Diese IDs sind so gestaltet, dass sie auch bei Systemwechseln, Datenmigrationen oder im Austausch zwischen verschiedenen Akteuren ihre Eindeutigkeit und Referenzierbarkeit behalten. Die VDV 462 und damit auch der VSPV-Standard 101 geben hierzu klare Konventionen vor, beispielsweise durch die Verwendung von Namensräumen, Präfixen und standardisierten Schemata (z. B. VSPV:StopPlace:NRW-1001).

Die Versionierung von Objekten ist ein weiteres wesentliches Prinzip. Sie ermöglicht es, Änderungen an einzelnen Datenobjekten – etwa die Anpassung eines Bedienegebiets, die Aktualisierung eines Fahrplans oder die Einführung neuer Tarifregeln – gezielt und nachvollziehbar zu dokumentieren. Jede Änderung an einem Objekt führt zu einer neuen Version, die mit einem eigenen Zeitstempel und Gültigkeitszeitraum versehen wird. Frühere Versionen bleiben erhalten und können bei Bedarf für historische Analysen oder zur Konsistenzprüfung herangezogen werden. Dieses Prinzip der Versionierung und Historisierung unterstützt die Transparenz und Nachvollziehbarkeit aller betrieblichen Änderungen und ist insbesondere für die Qualitätssicherung und die revisionssichere Datenhaltung von großer Bedeutung.

Die Gültigkeit von Objekten wird im NeTeX-Modell explizit durch Gültigkeitszeiträume (ValidBetween) gesteuert. Dadurch ist es möglich, sowohl zukünftige als auch zurückliegende Zustände des Netzes, der Fahrpläne oder der Tariflandschaft präzise abzubilden. Für Gelegenheitsverkehre mit häufigen Anpassungen – etwa bei der temporären Einrichtung neuer Bedienegebiete, der saisonalen Anpassung von Betriebszeiten oder der Einführung von Sondertarifen – ist diese Funktionalität unverzichtbar. Sie gewährleistet, dass Fahrgäste und Systeme stets auf aktuelle und gültige Informationen zugreifen können, während gleichzeitig die Historie und Entwicklung der Angebote lückenlos dokumentiert bleibt.

Die Trennung und zugleich die gezielte Verknüpfung von Netz-, Fahrplan- und Tarifdaten ist ein weiteres zentrales Merkmal des Modells. Netzstrukturdaten (Haltestellen, Linien, Bedienegebiete) werden unabhängig von den zeitlichen Fahrplandaten und den zugehörigen Tarifinformationen modelliert. Über Referenzen werden diese Bereiche jedoch logisch miteinander verbunden: Eine Fahrt verweist auf die bedienten Haltestellen und das zugrundeliegende Netz, ein Tarif verweist auf die zugehörigen Tarifzonen oder Bedienegebiete, und ein Buchungsprozess kann sowohl mit Fahrten als auch mit bestimmten Netzbereichen verknüpft werden. Diese Architektur erlaubt es, Änderungen in einem Bereich (z. B. Anpassung eines Tarifs) vorzunehmen, ohne dass zwangsläufig alle anderen Bereiche angepasst werden müssen, und sichert dennoch die Konsistenz und Nachvollziehbarkeit der gesamten Datenbasis.

Ein weiterer wichtiger Aspekt ist die Unterstützung von Mehrsprachigkeit und Internationalisierung. Im NeTeX- und VSPV-Modell können alle relevanten Texte, Labels und Fahrgastinformationen in mehreren Sprachen hinterlegt werden. Dies ist insbesondere für die barrierefreie Fahrgastinformation und für die Integration in internationale Plattformen von Bedeutung. Die Standardisierung der Sprachkennzeichnung und die Möglichkeit, für jedes Textelement mehrere Sprachversionen bereitzustellen, erleichtern die Nutzung der Daten durch unterschiedliche Zielgruppen und Systeme.

Schließlich sieht das Modell verschiedene Mechanismen für Erweiterungen und die Abbildung individueller Anforderungen vor. Neben den im Standard definierten Attributen können zusätzliche Informationen über sogenannte Extensions oder Key-Value-Paare (keyList) eingebunden werden. Diese Mechanismen ermöglichen es, betriebliche Besonderheiten, regionale Vorgaben oder innovative Angebotsformen zu modellieren, ohne die Kompatibilität mit dem Gesamtsystem zu gefährden. Die Erweiterungsmechanismen sind so gestaltet, dass sie die Validierbarkeit und Interoperabilität der Daten nicht beeinträchtigen und perspektivisch in zukünftige Standardversionen übernommen werden können.

Die Qualitätssicherung und Validierung der Daten erfolgt auf mehreren Ebenen. Technische Prüfungen stellen sicher, dass die gelieferten Daten den strukturellen Vorgaben des XML-Schemas entsprechen. Ergänzend kommen semantische Validierungen zum Einsatz, die beispielsweise die Konsistenz von Referenzen, die Einhaltung von Pflichtfeldern oder die Gültigkeit von Zeiträumen überprüfen. Mappingtabellen, die die Zuordnung der fachlichen Anforderungen zu den technischen

Datenstrukturen dokumentieren, sind ein zentrales Werkzeug für die Implementierung und Pflege des Modells. Sie dienen als Referenz für Entwickler, Systemintegratoren und Datenverantwortliche und sichern die Nachvollziehbarkeit und Wiederverwendbarkeit der Modellierung.

Insgesamt schafft die Kombination aus Modularität, Objektorientierung, klaren Identifikations- und Versionierungsregeln, flexiblen Erweiterungsmechanismen und umfassenden Validierungsverfahren ein robustes, skalierbares und zukunftsfähiges Fundament für die Modellierung von Gelegenheitsverkehren im Rahmen des VSPV-Standards 101. Dieses Fundament ist die Voraussetzung dafür, dass die spezifischen Anforderungen flexibler Mobilitätsangebote nahtlos in bestehende und zukünftige ÖPNV-Systemlandschaften integriert werden können.

Ein besonderes Augenmerk verdient die Art und Weise, wie NeTeX und der VSPV-Standard 101 die Erweiterbarkeit und Anpassungsfähigkeit des Modells an betriebliche und regionale Anforderungen sicherstellen. Während das Grundmodell bereits eine breite Palette von Verkehrsformen und betrieblichen Prozessen abdeckt, ist es unvermeidlich, dass in der Praxis zusätzliche Informationen benötigt werden, die im Standard nicht explizit vorgesehen sind. Um diese Anforderungen zu erfüllen, stehen zwei zentrale Erweiterungsmechanismen zur Verfügung: der Einsatz von Key-Value-Paaren in sogenannten keyList-Strukturen und die Nutzung des Extensions-Elements.

Key-Value-Paare bieten eine einfache Möglichkeit, zusätzliche Attribute an beliebigen Objekten zu hinterlegen, ohne die Struktur des Grundmodells zu verändern. Sie eignen sich insbesondere für die Übertragung von Zusatzinformationen wie internen Kennungen, speziellen Buchungsregeln, regionalen Einschränkungen oder temporären Betriebsmerkmalen. Die Verwendung von Key-Value-Paaren ist in der VDV 462 und im VSPV-Standard 101 klar geregelt: Schlüsselbezeichnungen müssen eindeutig, dokumentiert und zwischen den beteiligten Partnern abgestimmt sein, um die Interoperabilität zu gewährleisten.

Für komplexere oder strukturierte Erweiterungen, die über einzelne Attribute hinausgehen, steht das Extensions-Element zur Verfügung. Hier können auch umfangreichere proprietäre Datenstrukturen als XML-Elemente eingefügt werden, beispielsweise für spezielle Tarifmodelle, dynamische Preisinformationen oder zusätzliche Buchungsfunktionen. Die Nutzung von Extensions erfordert eine sorgfältige Dokumentation und Abstimmung, um die Validierbarkeit und Kompatibilität der Daten zu sichern. Ziel ist es, Erweiterungen so zu gestalten, dass sie perspektivisch in zukünftige Versionen des Standards übernommen werden können, wenn sie sich als allgemein relevant erweisen.

Die Integration dieser Erweiterungsmechanismen in das Grundmodell ermöglicht es, Innovationen und betriebliche Besonderheiten flexibel und ohne Systembruch abzubilden. Gleichzeitig bleibt das Datenmodell robust, validierbar und interoperabel, da alle Erweiterungen klar gekennzeichnet und dokumentiert werden. Dies fördert die Akzeptanz des Standards bei unterschiedlichen Akteuren und erleichtert die kontinuierliche Weiterentwicklung im Dialog zwischen Praxis und Standardisierung.

Ein weiterer wichtiger Baustein der Grundlagenmodellierung ist die technische und semantische Qualitätssicherung. Neben der Prüfung auf Schema-Konformität (z. B. mittels XSD-Validierung) werden semantische Prüfungen eingesetzt, um die inhaltliche Konsistenz und Vollständigkeit der Daten zu gewährleisten. Hierzu zählen unter anderem die Überprüfung der Referenzintegrität, die Einhaltung von Pflichtfeldern, die Konsistenz von Zeiträumen und die Plausibilität von Attributwerten. Die Mappingtabellen, die im VSPV-Standard 101 als verbindliches Werkzeug bereitgestellt werden, spielen hierbei eine zentrale Rolle: Sie dokumentieren die Zuordnung aller fachlichen Anforderungen zu den technischen Datenstrukturen, unterstützen die Implementierung und dienen als Referenz für die Qualitätssicherung.

Die Kombination aus modularer Architektur, objektorientierter Modellierung, klaren Erweiterungsmechanismen und umfassender Qualitätssicherung schafft ein leistungsfähiges und zukunftsfähiges Fundament für die digitale Abbildung von Gelegenheitsverkehren. Sie stellt sicher, dass neue Mobilitätsangebote, betriebliche Innovationen und regulatorische Anforderungen flexibel und ohne Medienbruch in die bestehende Systemlandschaft integriert werden können – ein entscheidender Erfolgsfaktor für die nachhaltige Weiterentwicklung des ÖPNV in Nordrhein-Westfalen und darüber hinaus.

Ein weiterer Aspekt, der die Praxistauglichkeit und Zukunftsfähigkeit der Grundlagenmodellierung im VSPV-Standard 101 unterstreicht, ist die konsequente Unterstützung von Mehrsprachigkeit und Internationalisierung. Gerade im Kontext wachsender Mobilitätsplattformen, touristischer Angebote und der zunehmenden Bedeutung grenzüberschreitender Fahrgastinformation ist es unerlässlich, alle relevanten Texte, Bezeichnungen und Hinweise in mehreren Sprachen bereitstellen zu können. Das Datenmodell sieht hierfür standardisierte Mechanismen vor: Für jedes Textelement – etwa Haltestellennamen, Linienbezeichnungen, Bedienhinweise oder Buchungsinformationen – können beliebig viele Sprachversionen hinterlegt werden, jeweils eindeutig durch Sprachcodes (z. B. „de“ für Deutsch, „en“ für Englisch) gekennzeichnet. Diese Mehrsprachigkeit zieht sich durch sämtliche Ebenen des Modells und ermöglicht es, die Daten ohne Redundanzen oder Medienbrüche in verschiedenen Sprachen auszugeben und zu verarbeiten.

Die Internationalisierung betrifft nicht nur die Fahrgastinformation, sondern auch die technische Interoperabilität. Durch die Orientierung an europaweit anerkannten Standards wie NeTeX und die Übernahme international gebräuchlicher Begriffe und Strukturen wird sichergestellt, dass die im VSPV-Standard 101 modellierten Daten ohne Anpassungsaufwand in internationale Plattformen, nationale Zugangspunkte oder grenzüberschreitende Auskunftssysteme integriert werden können. Dies ist insbesondere für die Zukunftsfähigkeit und Skalierbarkeit des Standards von zentraler Bedeutung, da Mobilitätsdaten zunehmend als Teil eines europäischen oder globalen Ökosystems betrachtet werden.

Die Grundlagenmodellierung legt zudem Wert auf eine klare Trennung zwischen fachlicher und technischer Ebene. Während die fachliche Modellierung die betrieblichen Anforderungen, Prozesse und Regeln beschreibt, erfolgt die technische Umsetzung in

Form von XML-Strukturen, Mappingtabellen und Validierungsregeln. Diese Trennung ermöglicht es, fachliche Weiterentwicklungen – etwa die Einführung neuer Bedienformen, Tarifmodelle oder Buchungsprozesse – unabhängig von technischen Details zu konzipieren und anschließend gezielt in die technische Modellierung zu überführen. Die Mappingtabellen bilden dabei die Brücke zwischen beiden Ebenen und sichern die Nachvollziehbarkeit und Konsistenz der Umsetzung.

Nicht zuletzt ist die Grundlagenmodellierung darauf ausgelegt, den gesamten Lebenszyklus der Daten – von der Ersterfassung über die Pflege und Aktualisierung bis hin zur Archivierung und Auswertung – systematisch zu unterstützen. Die Kombination aus eindeutigen Identifikatoren, Versionierung, Gültigkeitszeiträumen und Historisierung ermöglicht es, alle Änderungen an Netz, Fahrplänen, Tarifen oder Buchungsregeln lückenlos zu dokumentieren und bei Bedarf reproduzierbar zu machen. Dies ist nicht nur für die Qualitätssicherung und das Berichtswesen wichtig, sondern auch für die Erfüllung regulatorischer Anforderungen, etwa im Rahmen der Mobilitätsdatenverordnung oder bei der Bereitstellung von Nachweisen gegenüber Aufgabenträgern und Behörden.

Insgesamt schafft die Grundlagenmodellierung im VSPV-Standard 101 ein belastbares, flexibles und zukunftssicheres Fundament, das sowohl den aktuellen als auch den absehbaren zukünftigen Anforderungen an die digitale Abbildung von Gelegenheitsverkehren gerecht wird. Sie ermöglicht es allen Beteiligten – von Verkehrsunternehmen über Systemhersteller bis zu Plattformbetreibern und Aufgabenträgern –, innovative Mobilitätsangebote effizient, interoperabel und medienbruchfrei in die bestehende und künftige ÖPNV-Landschaft zu integrieren.

Ein zukunftsfähiges Datenmodell muss nicht nur die heutigen Anforderungen abbilden, sondern auch offen für Weiterentwicklungen und Innovationen bleiben. Die im VSPV-Standard 101 verankerten Prinzipien der Erweiterbarkeit und Modularität gewährleisten, dass neue Verkehrsformen, technische Innovationen oder regulatorische Anpassungen ohne grundlegende Systembrüche integriert werden können. Dies betrifft beispielsweise die künftige Einbindung von autonomen Fahrzeugen in den Gelegenheitsverkehr, die Integration von Sharing-Angeboten oder die Anbindung an neue Buchungs- und Zahlungsplattformen. Durch die klare Strukturierung der Daten in Frames und die Nutzung von Erweiterungsmechanismen wie Extensions und KeyValue-Paaren bleibt das Modell flexibel und anpassbar, ohne die Validierbarkeit und Interoperabilität zu gefährden.

Die Grundlagenmodellierung legt zudem besonderen Wert auf die Datenhoheit und die Nachvollziehbarkeit aller Änderungen. Jeder Akteur – sei es ein Verkehrsunternehmen, ein Betreiber eines On-Demand-Systems oder ein Aufgabenträger – kann seine eigenen Datenbereiche pflegen und aktualisieren, während die konsistente Verknüpfung über Referenzen und IDs sichergestellt bleibt. Die Versionierung und Historisierung aller Objekte erlaubt es, jederzeit auf frühere Zustände zurückzugreifen, Änderungen transparent zu dokumentieren und bei Bedarf auch komplexe Rollback-Szenarien oder Datenanalysen durchzuführen.

Ein weiterer Vorteil der gewählten Modellierungsprinzipien ist die Unterstützung für automatisierte Prozesse und Schnittstellen. Die klare Definition von Objektstrukturen, Attributen, Gültigkeiten und Beziehungen ermöglicht es, Datenflüsse zwischen unterschiedlichen Systemen zu automatisieren, Qualitätssicherungsprozesse zu standardisieren und die Integration neuer Partner oder Plattformen zu beschleunigen. Mappingtabellen und Validierungsregeln dienen dabei als technische Referenz und unterstützen die Implementierung von Import- und Exportschnittstellen, Datenkonvertern oder Monitoring-Lösungen.

Nicht zuletzt fördert die Grundlagenmodellierung auch die Transparenz und Vergleichbarkeit von Mobilitätsangeboten. Durch die standardisierte Beschreibung von Bedienegebieten, Fahrplänen, Buchungsregeln und Tarifen können Fahrgäste, Aufgabenträger und andere Stakeholder verschiedene Angebote objektiv vergleichen, bewerten und weiterentwickeln. Die Möglichkeit, alle relevanten Informationen maschinenlesbar und interoperabel bereitzustellen, ist eine wesentliche Voraussetzung für die Entwicklung offener Mobilitätsplattformen, die Integration in MaaS-Ökosysteme und die Erfüllung gesetzlicher Transparenzanforderungen.

Damit bildet die Grundlagenmodellierung im VSPV-Standard 101 das stabile Rückgrat für alle weiteren Kapitel und spezifischen Modellierungsabschnitte. Sie schafft die Voraussetzung, dass die besonderen Anforderungen der Gelegenheitsverkehre – von dynamischen Bedienegebieten über flexible Fahrpläne bis hin zu innovativen Tarif- und Buchungsmodellen – nahtlos, effizient und zukunftssicher in die digitale Infrastruktur des ÖPNV integriert werden können.

Die in der Grundlagenmodellierung verankerten Prinzipien wirken sich auch unmittelbar auf die tägliche Praxis der Datenpflege, Systemintegration und Qualitätssicherung aus. Durch die konsequente Trennung von fachlicher und technischer Modellierung können Verkehrsunternehmen und Systembetreiber ihre betrieblichen Anforderungen und Prozesse unabhängig von technischen Detailfragen beschreiben und weiterentwickeln. Die technische Umsetzung erfolgt dann auf Basis der im VSPV-Standard 101 festgelegten Strukturen, Konventionen und Validierungsregeln. Diese Arbeitsteilung fördert die Effizienz, reduziert Fehlerquellen und erleichtert die Zusammenarbeit zwischen unterschiedlichen Akteuren, etwa bei der Einführung neuer Angebote, der Migration von Altsystemen oder der Integration externer Partner.

Ein praktisches Beispiel für die Vorteile dieses Ansatzes ist die Einführung eines neuen On-Demand-Angebots in einer Kommune. Die fachliche Beschreibung umfasst das gewünschte Bedienegebiet, die Betriebszeiten, Buchungsregeln, Kapazitätsgrenzen und Tarifmodelle. Die technische Umsetzung erfolgt durch die Modellierung des Bedienegebiets als FlexibleStopPlace mit Polygon-Geometrie, die Definition von Betriebszeiten und Buchungsregeln im ServiceFrame, die Zuordnung von Fahrten im TimetableFrame und die Hinterlegung der Tarifinformationen im FareFrame. Alle Objekte erhalten eindeutige IDs, werden versioniert und mit Gültigkeitszeiträumen versehen. Änderungen – etwa die Erweiterung des Bedienegebiets oder die Anpassung der Buchungsregeln – können gezielt und nachvollziehbar vorgenommen werden, ohne die Integrität des Gesamtsystems zu gefährden.

Die Validierung und Qualitätssicherung erfolgt automatisiert auf Basis der bereitgestellten Mappingtabellen und Validierungsregeln. Technische Prüfungen stellen sicher, dass die Daten den strukturellen Vorgaben entsprechen, während semantische Prüfungen die inhaltliche Konsistenz und Vollständigkeit kontrollieren. Fehler oder Inkonsistenzen werden frühzeitig erkannt und können gezielt behoben werden. Die enge Verzahnung von fachlicher und technischer Modellierung, unterstützt durch klare Rollen und Verantwortlichkeiten, ermöglicht eine effiziente und nachhaltige Pflege der Datenbestände.

Darüber hinaus schafft die Grundlagenmodellierung die Voraussetzung für eine kontinuierliche Weiterentwicklung und Innovation. Neue Anforderungen – etwa aus der Weiterentwicklung der Mobilitätsdatenverordnung, aus der Einführung neuer Technologien oder aus veränderten Kundenbedürfnissen – können flexibel in das bestehende Modell integriert werden. Durch die Nutzung von Erweiterungsmechanismen, die klare Dokumentation und die regelmäßige Pflege der Mappingtabellen bleibt das Modell stets aktuell und anschlussfähig. Dies sichert die Investitionen in die digitale Infrastruktur und fördert die nachhaltige Entwicklung des ÖPNV und der Gelegenheitsverkehre in Nordrhein-Westfalen.

Insgesamt zeigt sich, dass die im VSPV-Standard 101 verankerte Grundlagenmodellierung weit mehr ist als ein technisches Detail: Sie ist das Fundament für eine moderne, effiziente und zukunftsfähige Mobilitätsdateninfrastruktur, die den vielfältigen Anforderungen von Verkehrsunternehmen, Aufgabenträgern, Systemherstellern und Fahrgästen gleichermaßen gerecht wird.

Die in diesem Kapitel dargelegten Grundlagenprinzipien – Modularität, Objektorientierung, eindeutige Identifikation, Versionierung, Gültigkeitssteuerung, Mehrsprachigkeit, Erweiterungsmechanismen und Qualitätssicherung – bilden das stabile Fundament des VSPV-Standards 101. Sie gewährleisten, dass sämtliche Daten zu Gelegenheitsverkehren, Taxiverkehren und flexiblen Betriebsformen in einer einheitlichen, konsistenten und interoperablen Struktur abgebildet werden können. Die konsequente Anwendung dieser Prinzipien ermöglicht es, die Vielfalt und Dynamik moderner Mobilitätsangebote effizient zu modellieren, zu pflegen und in bestehende wie zukünftige ÖPNV-Systemlandschaften zu integrieren.

Für die nachfolgenden Kapitel des VSPV-Standards 101 sind diese Grundlagen von zentraler Bedeutung: Sie bestimmen die Art und Weise, wie spezifische Elemente wie Bedienegebiete, flexible Fahrten, Buchungs- und Dispositionsregeln, Tarife und Taximodelle technisch umgesetzt werden. Die in der Grundlagenmodellierung verankerten Mechanismen sorgen dafür, dass auch komplexe und sich wandelnde Anforderungen – wie etwa die Einführung neuer Bedienformen, die Anpassung von Tarifstrukturen oder die Integration innovativer Buchungsprozesse – ohne Systembrüche, Medienbrüche oder Inkonsistenzen abgebildet werden können. Damit wird die Voraussetzung geschaffen, dass Gelegenheitsverkehre als gleichwertiger und voll integrierter Bestandteil des ÖPNV in Nordrhein-Westfalen und darüber hinaus etabliert werden können.

Die Grundlagenmodellierung legt die methodische und technische Basis für das gesamte Datenmodell, erhebt jedoch nicht den Anspruch, sämtliche betrieblichen, regulatorischen oder technologischen Anforderungen im Detail zu adressieren. Insbesondere Aspekte der Echtzeitkommunikation, der operativen Disposition, der Fahrgastinteraktion außerhalb der strukturierten Datenhaltung (z. B. Live-Chat, Push-Nachrichten) oder der Abrechnung und Zahlungsabwicklung werden im Rahmen der Grundlagenmodellierung nicht abgedeckt, sondern in darauf aufbauenden technischen oder betrieblichen Spezifikationen behandelt.

Ebenso werden spezifische Validierungs- und Monitoringprozesse, die über die Prüfung der Datenstruktur und -konsistenz hinausgehen, in späteren Kapiteln oder in ergänzenden Leitfäden beschrieben. Die Grundlagenmodellierung ist somit als Rahmen und Werkzeugkasten zu verstehen, der die Grundlage für alle weiteren Modellierungsabschnitte bildet.

In den folgenden Kapiteln des VSPV-Standards 101 werden die hier beschriebenen Prinzipien konkretisiert und auf die spezifischen Anforderungen der Gelegenheitsverkehre angewendet. Dabei werden die Modellierung von Netzpunkten und Bediengebieten, die Abbildung flexibler Fahrten und Buchungsprozesse, die Integration von Tarifen und Taxiangeboten sowie die Mechanismen für den Datenaustausch und die Interoperabilität im Detail beschrieben. Die Grundlagenmodellierung bleibt dabei stets der verbindliche Bezugsrahmen für alle weiteren Spezifikationen und Implementierungen.

Neben den in diesem Kapitel dargestellten Strukturprinzipien für Fahrten, Haltepunkte und Bediengebiete umfasst das technische Modell des VSPV-Standards 101 auch Elemente, die auf die Nutzerinteraktion und Abrechnungslogik bezogen sind. Dazu gehören insbesondere Informationen über Buchungsvorgänge, Zahlungsprozesse und besondere Anforderungen einzelner Fahrgäste, etwa im medizinischen Kontext. Diese Komponenten sind erforderlich, um die vollständige Prozesskette von der Fahrthanfrage bis zur Abrechnung modellseitig abzubilden und insbesondere den Anforderungen des SDGV-Projekts an eine tiefe Integration gerecht zu werden.

## 2.2 Technische Konventionen und Erweiterungsmechanismen

Ein zentrales Element der technischen Konventionen im VSPV-Standard 101 ist die eindeutige und konsistente Vergabe von Identifikatoren für sämtliche Objekte im Datenmodell. Jede Entität – von Haltestellen und Bediengebieten über Fahrten und Tarife bis hin zu Buchungsregeln – erhält einen persistenten, systemweit eindeutigen Identifikator. Diese IDs sind so konzipiert, dass sie auch beim Austausch zwischen unterschiedlichen Systemen, bei Datenmigrationen oder in föderierten Datenlandschaften ihre Eindeutigkeit behalten. Die Struktur der Identifikatoren folgt klaren Regeln, die sich an etablierten Standards wie der VDV 462 und NeTeX orientieren: Ein Namensraum-Präfix kennzeichnet die Organisation, das System oder die Region (beispielsweise „VSPV:“, „NRW:“, „DIVA:“), gefolgt von einer Kennzeichnung des Objekttyps und einer laufenden Nummer oder einem eindeutigen Schlüssel.

Ein Beispiel für eine solche ID ist

```
<FlexibleStopPlace id="VSPV:StopPlace:NRW-1001" version="1.0">  
  <!-- weitere Attribute -->  
</FlexibleStopPlace>
```

für einen dynamischen Haltepunkt oder

```
<ServiceJourney id="VSPV:ServiceJourney:OB-2025-001" version="2.0">  
  <!-- weitere Attribute -->  
</ServiceJourney>
```

für eine Fahrt.

Diese IDs sind dauerhaft und stabil zu vergeben. Einmal vergebene Identifikatoren dürfen nicht wiederverwendet oder nachträglich verändert werden. Änderungen am Objekt, wie etwa die Anpassung eines Bedienegebiets oder die Aktualisierung von Fahrplandaten, führen zur Vergabe einer neuen Version, nicht zu einer neuen ID. Die Versionierung erfolgt über ein separates Attribut, sodass die Historie und Entwicklung jeder Entität lückenlos nachvollziehbar bleibt.

Für die Pflege und Vergabe der Identifikatoren empfiehlt sich die zentrale Dokumentation, idealerweise in einer organisationsweiten ID-Registry. Dies erleichtert die Verwaltung, verhindert Kollisionen und unterstützt die Harmonisierung bei der Integration externer Datenquellen. Bei der Integration von Fremddaten ist darauf zu achten, dass Namensräume und Präfixe abgestimmt werden, um Mehrdeutigkeiten auszuschließen.

Die konsequente Anwendung dieser Namens- und ID-Konventionen ist die Grundlage für die Referenzierbarkeit, Wiederverwendbarkeit und Interoperabilität der Daten im gesamten VSPV-Standard 101. Sie ermöglicht es, komplexe Beziehungen zwischen Objekten zuverlässig abzubilden und ist ein unverzichtbares Fundament für alle weiteren technischen und fachlichen Prozesse der Modellierung und des Datenaustauschs.

Der VSPV-Standard 101 ergänzt die vorhandenen NeTeX-Profilen um weitere Attribute und Mechanismen, wie sie im SDGV-Datenmodell vorgesehen sind. So wird das Objekt FlexibleStopPlace um zusätzliche Eigenschaften erweitert (z. B. Angabe von Geokoordinaten-Polygonen zur exakten Definition eines Bedienegebiets sowie Kennzeichnung des Haltepunkttyps – Haltestelle, Adresse oder Sammelpunkt). Flexible Linien und Fahrten (etwa FlexibleLine und FlexibleServiceJourney) erhalten die Möglichkeit, Buchungslinks für externe Buchungssysteme sowie variable Zeitfenster für die Durchführung von Fahrten zu definieren. Darüber hinaus kommen standardisierte KeyValue-Attribute zum Einsatz, um spezielle betriebliche Parameter abzubilden – etwa Mindestfahrweiten, Fahrtkategorien (Regelfahrt, Krankenfahrt etc.) oder Ausschlussregeln –, wie sie im SDGV-Konzept beschrieben sind. All diese Erweiterungen sind so gestaltet, dass sie nahtlos in die bestehende NeTeX-Struktur integriert werden können. Sie folgen den Konventionen der VDV 462 (German Profile für NeTeX) und wurden zwischen den Projektbeteiligten abgestimmt, um die Interoperabilität und zukünftige Überführbarkeit in offizielle Standards zu gewährleisten.

Ein weiterer zentraler Aspekt der technischen Konventionen im VSPV-Standard 101 betrifft die Vorgaben für Zeichensätze, Feldgrößen und die Behandlung von Texten. Um die Interoperabilität zwischen unterschiedlichen Systemen und die internationale Nutzbarkeit der Daten zu gewährleisten, ist die Verwendung des Unicode-Zeichensatzes UTF-8 verbindlich vorgeschrieben. Dies ermöglicht die fehlerfreie Darstellung und Verarbeitung aller in Europa und darüber hinaus gebräuchlichen Schriftzeichen, einschließlich Sonderzeichen, diakritischer Zeichen und nicht-lateinischer Alphabete.

Für alle textuellen Felder im Datenmodell gelten klar definierte Längenbeschränkungen, die sich an den Empfehlungen der VDV 462 orientieren. So sind beispielsweise Haltestellennamen in der Regel auf 70 Zeichen begrenzt, Linienbezeichnungen auf 30 Zeichen und Kurzttexte für Bedienhinweise oder Buchungsregeln auf 255 Zeichen. Diese Begrenzungen dienen der Kompatibilität mit bestehenden Auskunftssystemen und verhindern Darstellungsprobleme in Endgeräten oder Schnittstellen. Bei der Modellierung und Pflege der Daten ist darauf zu achten, dass diese Feldlängen nicht überschritten werden; längere Texte sind gegebenenfalls zu kürzen oder in separate Langtextfelder auszulagern.

Die Behandlung von Texten umfasst auch die konsequente Unterstützung von Mehrsprachigkeit. Für jedes relevante Textelement – etwa Haltestellenbezeichnungen, Linienamen, Bedienhinweise oder Buchungsinformationen – können und sollten mehrere Sprachversionen hinterlegt werden. Dies erfolgt durch die Angabe des jeweiligen Sprachcodes nach ISO 639-1 (z. B. „de“ für Deutsch, „en“ für Englisch, „tr“ für Türkisch). Ein Beispiel für einen mehrsprachigen Haltestellennamen im XML-Format sieht folgendermaßen aus:

```
<Name>
  <Text language="de">Hauptbahnhof</Text>
  <Text language="en">Central Station</Text>
  <Text language="tr">Merkez İstasyonu</Text>
</Name>
```

Die Mehrsprachigkeit ist nicht nur für die barrierefreie Fahrgastinformation relevant, sondern auch für die Integration in internationale Plattformen und die Einhaltung gesetzlicher Vorgaben zur Gleichstellung und Teilhabe. Bei der Pflege der Daten ist darauf zu achten, dass alle relevanten Sprachen konsistent und aktuell gehalten werden und dass keine Informationen verloren gehen, wenn neue Sprachversionen hinzugefügt oder bestehende geändert werden.

Sonderzeichen und Formatierungen sind nur insoweit zulässig, als sie mit dem UTF-8-Zeichensatz und den Vorgaben der VDV 462 kompatibel sind. Nicht darstellbare oder systemkritische Zeichen (z. B. Steuerzeichen, nicht druckbare Zeichen) sind zu vermeiden. Für spezielle Anforderungen – etwa die Darstellung von mathematischen Symbolen in Tarifregeln oder die Einbindung von Piktogrammen – sind standardisierte Unicode-Zeichen zu verwenden oder entsprechende Alternativen zu definieren.

Die Einhaltung dieser Zeichensatz-, Feldgrößen- und Textkonventionen ist eine wesentliche Voraussetzung für die technische Validierbarkeit, die Interoperabilität und

die Nutzerfreundlichkeit der im VSPV-Standard 101 modellierten Daten. Sie erleichtert die Integration in bestehende und zukünftige Systemlandschaften und stellt sicher, dass die Informationen korrekt und verständlich an alle Zielgruppen – von Fahrgästen über Systemhersteller bis zu internationalen Plattformen – weitergegeben werden können.

Für den Austausch von Daten im Rahmen des VSPV-Standards 101 ist das XML-Format auf Basis des NeTeX-Schemas verbindlich vorgeschrieben. XML (Extensible Markup Language) bietet die notwendige Strukturierung, Lesbarkeit und Validierbarkeit, um komplexe Mobilitätsdaten systemübergreifend und langfristig nutzbar zu machen. Das zugrundeliegende NeTeX-XSD-Schema definiert die erlaubten Strukturen, Datentypen und Beziehungen und stellt damit sicher, dass alle ausgetauschten Daten den formalen Anforderungen des Standards entsprechen. Die Einhaltung des Schemas wird durch technische Validierungsprozesse überprüft, sodass fehlerhafte oder unvollständige Daten frühzeitig erkannt und korrigiert werden können.

Der Austausch der XML-Daten kann entweder dateibasiert (z. B. per SFTP, Download-Link, Cloud-Speicher) oder über Webservices erfolgen. Im Falle von Webservices werden in der Regel REST- oder SOAP-Schnittstellen genutzt, die eine automatisierte, zeitgesteuerte oder ereignisbasierte Übertragung von Daten zwischen den beteiligten Systemen ermöglichen. Die genaue Ausgestaltung der Schnittstelle – etwa Authentifizierungsverfahren, Endpunkt-Definitionen, Übertragungsintervalle und Fehlerbehandlung – wird zwischen den beteiligten Partnern abgestimmt und in einer Schnittstellenbeschreibung dokumentiert.

Ein typischer XML-Austauschdatensatz beginnt mit einem <PublicationDelivery>-Element, das als Container für die einzelnen Frames (z. B. NetworkFrame, ServiceFrame, TimetableFrame) dient. Innerhalb dieser Frames werden die jeweiligen Objekte und deren Beziehungen abgebildet. Ein einfaches Beispiel für einen solchen Aufbau sieht wie folgt aus:

```
<PublicationDelivery>
  <ServiceFrame id="VSPV:ServiceFrame:2025-01" version="1.0">
    <FlexibleStopPlace id="VSPV:StopPlace:NRW-1001" version="1.0">
      <!-- weitere Attribute -->
    </FlexibleStopPlace>
    <!-- weitere Objekte -->
  </ServiceFrame>
</PublicationDelivery>
```

Die Struktur der XML-Dateien ist so gestaltet, dass sie sowohl für vollständige Datenlieferungen (Initial- oder Komplettimporte) als auch für inkrementelle Updates (z. B. nur geänderte oder neue Objekte) geeignet ist. Die Versionierung und Gültigkeitszeiträume der einzelnen Objekte ermöglichen es, Änderungen gezielt und nachvollziehbar zu übertragen, ohne dass ganze Datenbestände ausgetauscht werden müssen.

Für die Integration in bestehende Systemlandschaften und die Anbindung an externe Plattformen – etwa den Nationalen Zugangspunkt für Mobilitätsdaten – ist die Einhaltung

der NeTeX-Spezifikation und der im VSPV-Standard 101 festgelegten Profile verbindlich. Abweichungen oder proprietäre Erweiterungen sind explizit zu kennzeichnen und zu dokumentieren, um die Interoperabilität und Validierbarkeit der Daten zu gewährleisten.

Die Nutzung standardisierter Austauschformate und klar definierter Schnittstellen ist eine zentrale Voraussetzung für die Automatisierung von Datenflüssen, die Qualitätssicherung und die nachhaltige Pflege der Mobilitätsdaten. Sie ermöglicht es, neue Partner, Systeme oder Plattformen effizient anzubinden und die Datenbasis kontinuierlich aktuell und konsistent zu halten.

Trotz des umfangreichen und flexiblen Datenmodells von NeTeX und des VSPV-Standards 101 gibt es immer wieder Anforderungen, die über die im Standard vorgesehenen Strukturen hinausgehen. Um auch betriebliche Besonderheiten, regionale Vorgaben oder innovative Angebotsformen abbilden zu können, stehen zwei zentrale Erweiterungsmechanismen zur Verfügung: die Verwendung von KeyValue-Paaren und das Extensions-Element.

KeyValue-Paare ermöglichen es, beliebigen Objekten zusätzliche Attribute zuzuweisen, ohne die zugrundeliegende Struktur verändern zu müssen. Jedes KeyValue-Paar besteht aus einem Schlüssel (Key) und einem zugehörigen Wert (Value). Die Schlüssel sollten eindeutig, sprechend und zwischen allen beteiligten Partnern abgestimmt sein, um Missverständnisse und Inkonsistenzen zu vermeiden. Typische Anwendungsfälle sind die Übertragung von internen Kennungen, spezifischen Buchungsregeln, regionalen Einschränkungen oder temporären Betriebsmerkmalen. Ein Beispiel für die Verwendung von KeyValue-Paaren in einer keyList sieht wie folgt aus:

```
<FlexibleStopPlace id="VSPV:StopPlace:NRW-1001" version="1.0">
  <keyList>
    <KeyValue>
      <Key>MINDESTFAHRWEITE</Key>
      <Value>3km</Value>
    </KeyValue>
    <KeyValue>
      <Key>INNERORTSVERBOT</Key>
      <Value>>true</Value>
    </KeyValue>
  </keyList>
  <!-- weitere Attribute -->
</FlexibleStopPlace>
```

Für komplexere oder strukturierte Erweiterungen, die sich nicht durch einzelne KeyValue-Paare abbilden lassen, steht das Extensions-Element zur Verfügung. Innerhalb von <Extensions> können beliebige XML-Strukturen eingefügt werden, um beispielsweise spezielle Tarifmodelle, dynamische Preisinformationen, zusätzliche Buchungsfunktionen oder neue betriebliche Prozesse zu beschreiben. Die Nutzung von Extensions erfordert eine sorgfältige Dokumentation und Abstimmung, um die Validierbarkeit und Kompatibilität der Daten zu sichern. Ein Beispiel für die Verwendung von Extensions könnte folgendermaßen aussehen:

```
<FlexibleServiceJourney id="VSPV:Journey:2025-Q3-501" version="1.0">
  <Extensions>
    <vspv:BookingLink>https://booking.vspv.de?journey=501</vspv:BookingLink>
    <vspv:RealTimeDataRef>RTD_VSPV_501</vspv:RealTimeDataRef>
  </Extensions>
  <!-- weitere Attribute -->
</FlexibleServiceJourney>
```

Beide Erweiterungsmechanismen sind so gestaltet, dass sie die Validierbarkeit und Interoperabilität des Datenmodells nicht beeinträchtigen. Sie ermöglichen es, Innovationen und betriebliche Besonderheiten flexibel und ohne Systembruch zu integrieren, während das Grundmodell stabil und kompatibel bleibt. Wichtig ist, dass alle Erweiterungen klar dokumentiert und zwischen den beteiligten Systempartnern abgestimmt werden, um eine reibungslose Verarbeitung und Weiterentwicklung der Daten zu gewährleisten. Idealerweise werden bewährte Erweiterungen perspektivisch in zukünftige Versionen des Standards übernommen.

Die konsequente Nutzung von KeyValue-Paaren und Extensions macht den VSPV-Standard 101 zu einem lebendigen und anpassungsfähigen Werkzeug, das sowohl die aktuellen als auch die künftigen Anforderungen des Gelegenheitsverkehrs und der Digitalisierung im ÖPNV flexibel abbilden kann.

Ein grundlegender Bestandteil der technischen Konventionen im VSPV-Standard 101 ist der Umgang mit Versionierung, Gültigkeit und Historisierung von Datenobjekten. Da sich Mobilitätsangebote, Bediengebiete, Fahrpläne und betriebliche Regeln im Laufe der Zeit regelmäßig ändern, ist es unerlässlich, jede Änderung nachvollziehbar und konsistent im Datenmodell abzubilden. Jedes Objekt im Modell – sei es eine Haltestelle, ein Bediengebiet, eine Fahrt oder ein Tarif – trägt daher neben seiner eindeutigen ID auch eine Versionsnummer und Angaben zum Gültigkeitszeitraum.

Die Versionierung erfolgt über das Attribut „version“. Bei jeder inhaltlichen Änderung eines Objekts – etwa der Anpassung eines Bediengebiets, der Aktualisierung von Buchungsregeln oder der Einführung eines neuen Tarifs – wird die Versionsnummer inkrementiert. Frühere Versionen bleiben im Datenbestand erhalten, sodass die Entwicklung und Historie jedes Objekts lückenlos nachvollzogen werden kann. Dies ist nicht nur für die Qualitätssicherung und das Berichtswesen wichtig, sondern auch für die Reproduzierbarkeit vergangener Betriebszustände und die Erfüllung regulatorischer Anforderungen.

Die Gültigkeit von Objekten wird durch explizite Zeitangaben gesteuert, in der Regel durch die Elemente <ValidBetween>, <FromDate> und <ToDate>. Dadurch kann für jedes Objekt präzise festgelegt werden, in welchem Zeitraum es gültig ist. So lassen sich beispielsweise saisonale Bediengebiete, temporäre Fahrpläne oder befristete Tarifangebote sauber und ohne Medienbruch abbilden. Bei Überschneidungen oder Änderungen werden die jeweiligen Gültigkeitszeiträume angepasst, sodass zu jedem Zeitpunkt eindeutig bestimmt werden kann, welche Version eines Objekts aktuell gültig ist.

Ein Beispiel für die Kombination von Versionierung und Gültigkeit in einer XML-Struktur sieht folgendermaßen aus:

```
<FlexibleStopPlace id="VSPV:StopPlace:NRW-1001" version="2.0">
  <ValidBetween>
    <FromDate>2025-06-01</FromDate>
    <ToDate>2025-12-31</ToDate>
  </ValidBetween>
  <!-- weitere Attribute -->
</FlexibleStopPlace>
```

Die Historisierung aller Änderungen ist ein zentrales Prinzip im VSPV-Standard 101. Sie ermöglicht es, jederzeit auf frühere Zustände zurückzugreifen, Änderungen transparent zu dokumentieren und bei Bedarf auch komplexe Rollback-Szenarien oder Datenanalysen durchzuführen. Gerade für Gelegenheitsverkehre, die häufigen betrieblichen Anpassungen unterliegen, ist diese Funktionalität unverzichtbar.

Die konsequente Umsetzung von Versionierung, Gültigkeit und Historisierung stellt sicher, dass alle Daten im VSPV-Standard 101 nicht nur aktuell und valide, sondern auch revisionsicher und nachvollziehbar sind. Dies schafft Vertrauen bei allen Beteiligten und bildet die Grundlage für eine nachhaltige, zukunftssichere Datenhaltung im ÖPNV.

Die Sicherstellung der Datenqualität ist ein zentrales Anliegen des VSPV-Standards 101 und zieht sich durch alle technischen Konventionen und Modellierungsprinzipien. Damit die im Datenmodell abgebildeten Informationen zuverlässig, konsistent und interoperabel genutzt werden können, sind mehrstufige Validierungs- und Qualitätssicherungsprozesse vorgesehen.

Die erste Stufe bildet die technische Validierung gegen das zugrundeliegende NeTeX-XSD-Schema. Jede Datenlieferung muss formal korrekt aufgebaut sein und alle im Schema definierten Strukturen, Datentypen und Beziehungen einhalten. Diese Prüfung erfolgt automatisiert und stellt sicher, dass die XML-Dateien maschinenlesbar und systemübergreifend nutzbar sind. Fehlerhafte oder unvollständige Daten werden bereits bei der Annahme abgewiesen und müssen vor einer erneuten Übermittlung korrigiert werden.

Ergänzend zur technischen Prüfung erfolgt eine semantische Validierung, die die inhaltliche Konsistenz und Plausibilität der gelieferten Daten überprüft. Hierzu zählen unter anderem die Kontrolle der Referenzintegrität (z. B. ob alle referenzierten Objekte tatsächlich existieren), die Einhaltung von Pflichtfeldern, die Überprüfung von Wertebereichen und die Konsistenz von Gültigkeitszeiträumen. Auch die Einhaltung der in den Mappingtabellen dokumentierten Zuordnungen und Konventionen wird geprüft. Diese Tabellen dienen als zentrales Werkzeug für die Qualitätssicherung und bilden die Brücke zwischen fachlichen Anforderungen und technischer Umsetzung.

Ein weiteres Qualitätsmerkmal ist die regelmäßige Pflege und Aktualisierung der Mappingtabellen und Beispiel-XMLs. Sie stellen sicher, dass alle Neuerungen, Erweiterungen oder Anpassungen im Datenmodell zeitnah dokumentiert und in der Praxis nachvollziehbar umgesetzt werden. Die Bereitstellung von validierten Beispiel-XMLs für

typische Anwendungsfälle unterstützt Entwickler und Systemintegratoren bei der korrekten Implementierung und erleichtert die Fehlersuche im Betrieb.

Die Qualitätssicherung endet nicht mit der initialen Datenlieferung. Auch im laufenden Betrieb sind regelmäßige Prüfungen und Monitoringprozesse vorgesehen, um die Aktualität, Konsistenz und Vollständigkeit der Daten dauerhaft zu gewährleisten. Abweichungen oder Fehler werden dokumentiert, analysiert und zeitnah behoben. Die enge Verzahnung von technischer und fachlicher Qualitätssicherung sorgt dafür, dass der VSPV-Standard 101 als verlässliche Grundlage für die Digitalisierung und Standardisierung der Gelegenheitsverkehre im ÖPNV dient.

Durch diese mehrstufigen Validierungs- und Qualitätssicherungsprozesse wird sichergestellt, dass alle im Rahmen des VSPV-Standards 101 bereitgestellten Daten höchsten Ansprüchen an Korrektheit, Nachvollziehbarkeit und Interoperabilität genügen – eine unverzichtbare Voraussetzung für den erfolgreichen Betrieb und die kontinuierliche Weiterentwicklung moderner Mobilitätsangebote.

## 2.3 Einordnung und Abgrenzung zu anderen Standards

Ein zentraler Bezugspunkt für die Einordnung des VSPV-Standards 101 ist das international weit verbreitete Format GTFS (General Transit Feed Specification) und dessen Erweiterung GTFS-flex. GTFS hat sich als de-facto-Standard für die Bereitstellung von Fahrplandaten für digitale Fahrgastinformationssysteme, insbesondere für Routing- und Navigationsdienste wie Google Maps, etabliert. Die große Verbreitung von GTFS beruht auf seiner einfachen, tabellenbasierten Struktur, die eine schnelle Implementierung und breite Unterstützung durch zahlreiche Softwarelösungen weltweit ermöglicht. GTFS-Daten werden in Form von CSV-Dateien bereitgestellt, die über klar definierte Tabellen für Haltestellen, Linien, Fahrpläne, Fahrten und Verbindungen verfügen.

Allerdings ist das klassische GTFS-Format primär auf den Linienverkehr mit festen Haltestellen und Fahrplänen ausgelegt. Es eignet sich hervorragend für die statische Fahrgastinformation in Systemen, die keine komplexen, flexiblen oder on-demand-basierten Verkehrsangebote abbilden müssen. Die Modellierungstiefe in Bezug auf Tarife, Buchungsprozesse, dynamische Bedienegebiete oder betriebliche Besonderheiten ist im klassischen GTFS stark eingeschränkt.

Mit GTFS-flex wurde ein Versuch unternommen, diese Lücke zu schließen und auch flexible Bedienformen, Rufbusse und On-Demand-Angebote zu integrieren. GTFS-flex erweitert das Grundmodell von GTFS um zusätzliche Tabellen und Felder, mit denen beispielsweise flexible Haltepunkte, Buchungsfenster und Servicegebiete beschrieben werden können. Dennoch bleibt die semantische Präzision und die Tiefe der Modellierung hinter den Möglichkeiten von NeTEx zurück. Insbesondere für komplexe Anwendungsfälle – etwa die Integration von Taxiverkehren, die Abbildung variabler Tarife, die Verknüpfung mit Buchungssystemen oder die Modellierung von Dispositionsregeln – stößt GTFS-flex schnell an seine Grenzen. Auch die Unterstützung für Versionierung, Historisierung und mehrsprachige Fahrgastinformation ist in GTFS und GTFS-flex nur rudimentär vorhanden.

Ein weiterer Unterschied betrifft die Interoperabilität mit europäischen und nationalen IT-Systemen: Während GTFS vor allem für die Integration in internationale Routing- und Kartenplattformen optimiert ist, ist NeTEx als europäischer CEN-Standard speziell auf die Anforderungen öffentlicher Verkehrsunternehmen, Aufgabenträger und nationale Zugangspunkte zugeschnitten. Die Einhaltung regulatorischer Vorgaben, die Unterstützung für komplexe Tarifstrukturen und die nahtlose Integration in nationale und regionale Dateninfrastrukturen sind zentrale Stärken von NeTEx und damit des VSPV-Standards 101.

Trotz dieser Unterschiede ist die Kompatibilität mit GTFS und GTFS-flex für den VSPV-Standard 101 ein wichtiges Ziel. Über Konverter und Mappingtabellen können die in NeTEx/VDV 462 modellierten Daten in GTFS-Formate überführt werden, um die Sichtbarkeit und Nutzbarkeit der Angebote auf internationalen Plattformen sicherzustellen. Die klare Dokumentation der Zuordnungen und die Pflege von Konvertierungswerkzeugen gewährleisten, dass keine relevanten Informationen verloren gehen und die Daten konsistent zwischen den Systemen übertragen werden können. Damit bleibt der VSPV-Standard 101 anschlussfähig an die internationale Datenlandschaft, ohne die Vorteile der tieferen und flexibleren Modellierung von NeTEx aufzugeben.

Neben GTFS und GTFS-flex sind im europäischen und deutschen Kontext insbesondere die Standards TRIAS (Travellers Real-time Information and Advisory Standard) und SIRI (Service Interface for Real Time Information) von Bedeutung. Beide Formate wurden mit dem Ziel entwickelt, Echtzeitdaten im öffentlichen Verkehr strukturiert und interoperabel auszutauschen, unterscheiden sich jedoch in ihrer Ausrichtung und technischen Umsetzung von NeTEx und dem VSPV-Standard 101.

TRIAS ist ein XML-basierter Standard, der in Deutschland und weiteren europäischen Ländern für die Kommunikation zwischen Fahrplanauskunftssystemen, Mobilitätsplattformen und Echtzeit-APIs eingesetzt wird. Er ermöglicht die Übertragung von Echtzeitinformationen wie Prognosen zu Ankunfts- und Abfahrtszeiten, Verspätungen, Störungen, Anschlussbeziehungen und Fahrgastinformationen. TRIAS ist besonders leistungsfähig, wenn es um die dynamische Bereitstellung von Fahrgastinformationen und die Integration verschiedener Verkehrsträger in Echtzeitauskünfte geht. Allerdings ist TRIAS nicht für die langfristige, strukturierte Modellierung und Pflege von Netz-, Fahrplan- und Tarifdaten konzipiert. Vielmehr ergänzt TRIAS die statischen Datenmodelle wie NeTEx oder GTFS, indem es die dynamische Komponente der Fahrgastinformation abbildet.

SIRI, ein weiterer international etablierter Standard, fokussiert sich ebenfalls auf den Austausch von Echtzeitdaten im öffentlichen Verkehr. SIRI wird insbesondere für die Übertragung von Fahrzeugpositionen, Prognosen, Störungsmeldungen, Anschlussinformationen und Fahrgastzählungen genutzt. Der Standard ist modular aufgebaut und deckt verschiedene Anwendungsfälle ab, darunter Linienverfolgung, Störungsmanagement, Anschlussmanagement und Fahrgastinformation. Wie TRIAS ist SIRI kein Ersatz für umfassende Datenmodelle wie NeTEx, sondern eine spezialisierte

Schnittstelle für die Kommunikation von Echtzeitereignissen zwischen unterschiedlichen Systemen.

Die Abgrenzung zwischen NeTEx (bzw. dem VSPV-Standard 101) und den Echtzeitstandards TRIAS und SIRI ist somit klar: Während NeTEx die strukturierte, langfristige und revisionssichere Modellierung von Netz-, Fahrplan- und Tarifdaten ermöglicht, sind TRIAS und SIRI darauf ausgelegt, aktuelle Betriebszustände, Prognosen und Ereignisse in Echtzeit zu übertragen. In der Praxis werden beide Ansätze kombiniert: Die statischen Grunddaten werden in NeTEx gepflegt und bereitgestellt, während TRIAS oder SIRI für die Echtzeitkommunikation und die laufende Aktualisierung von Fahrgastinformationen genutzt werden.

Für den VSPV-Standard 101 bedeutet dies, dass die Modellierung von Gelegenheitsverkehren, Bediengebieten, Fahrten, Tarifen und Buchungsregeln auf der NeTEx-Basis erfolgt, während Echtzeitdaten – wie Fahrzeugpositionen, aktuelle Verfügbarkeiten oder kurzfristige Änderungen – über ergänzende TRIAS- oder SIRI-Schnittstellen bereitgestellt werden. Die klare Trennung der Verantwortlichkeiten und die definierte Integration der verschiedenen Standards stellen sicher, dass sowohl die langfristige Datenqualität als auch die Aktualität und Reaktionsfähigkeit der Systeme gewährleistet sind.

Im direkten Vergleich mit anderen Datenstandards zeigen sich die besonderen Stärken, aber auch die Grenzen von NeTEx und der darauf aufbauenden VDV 462 – und damit des VSPV-Standards 101. Die größte Stärke von NeTEx liegt in seiner enormen Modellierungstiefe und Flexibilität. Das Datenmodell ist objektorientiert, modular aufgebaut und ermöglicht die präzise Abbildung nahezu aller Aspekte des öffentlichen Verkehrs: von der Netzstruktur über Fahrpläne und Ressourcen bis hin zu komplexen Tarifmodellen, Buchungsregeln, Bediengebieten und betrieblichen Prozessen. Durch die Möglichkeit, eigene Profile, Erweiterungen und spezifische Attribute zu definieren, können auch regionale oder betriebliche Besonderheiten nahtlos integriert werden, ohne die Kompatibilität zum Gesamtsystem zu verlieren.

Ein weiterer Vorteil ist die Unterstützung für Versionierung, Gültigkeitszeiträume und Historisierung. Jede Änderung an einem Objekt – sei es eine Haltestelle, ein Bediengebiet oder ein Tarif – wird lückenlos dokumentiert und kann zeitlich genau nachvollzogen werden. Dies ist insbesondere für die Qualitätssicherung, die Revisionssicherheit und die Einhaltung regulatorischer Vorgaben von großer Bedeutung. Die Mehrsprachigkeit, die Möglichkeit zur Modellierung von Beziehungen und Referenzen sowie die klare Trennung und Verknüpfung von Netz-, Fahrplan- und Tarifdaten machen NeTEx zu einem leistungsfähigen Werkzeug für die Digitalisierung und Standardisierung des ÖPNV.

Die VDV 462 als deutsche Profilspezifikation von NeTEx trägt dazu bei, die Komplexität des europäischen Standards auf praxistaugliche und interoperable Anwendungen für den deutschen Markt zu fokussieren. Sie legt verbindliche Konventionen, Profile und technische Regeln fest, die die Einführung und Pflege von NeTEx-basierten Systemen erleichtern und die Zusammenarbeit zwischen Verkehrsunternehmen, Systemherstellern und Aufgabenträgern fördern.

Allerdings bringt diese Modellierungstiefe auch eine gewisse Komplexität mit sich. Die Implementierung und Pflege von NeTEx/VDV 462 erfordert ein hohes Maß an technischem Verständnis, sorgfältiger Planung und kontinuierlicher Qualitätssicherung. Für kleinere Verkehrsunternehmen oder Projekte mit sehr begrenztem Anwendungsumfang kann der initiale Aufwand höher sein als bei einfacheren Formaten wie GTFS. Zudem ist die Integration in internationale Plattformen und Routingdienste, die primär auf GTFS setzen, mit zusätzlichem Aufwand für Konvertierung und Mapping verbunden.

Ein weiterer Aspekt ist die Fokussierung von NeTEx auf die strukturierte Modellierung von statischen und semi-dynamischen Daten. Für die Übertragung und Verarbeitung von Echtzeitinformationen sind ergänzende Standards wie TRIAS oder SIRI notwendig. Die Interoperabilität zwischen den verschiedenen Standards ist zwar technisch möglich, erfordert aber eine klare Schnittstellendefinition und fortlaufende Abstimmung zwischen den beteiligten Systemen.

Insgesamt überwiegen jedoch die Vorteile: Die Tiefe, Flexibilität und Zukunftsfähigkeit von NeTEx/VDV 462 machen den VSPV-Standard 101 zu einer tragfähigen Grundlage für die Digitalisierung, Standardisierung und Integration moderner Gelegenheitsverkehre im ÖPNV – insbesondere, wenn Wert auf Interoperabilität, Erweiterbarkeit und die Einhaltung nationaler sowie europäischer Vorgaben gelegt wird.

Die Entscheidung, NeTEx als methodisches und technisches Fundament für den VSPV-Standard 101 zu wählen, ist das Ergebnis einer strategischen Abwägung zwischen Interoperabilität, Zukunftsfähigkeit, regulatorischen Anforderungen und der praktischen Umsetzbarkeit in der deutschen und europäischen Systemlandschaft. Einer der wichtigsten Gründe ist die europaweite Anerkennung von NeTEx als CEN-Standard, der von der Europäischen Kommission und nationalen Behörden als Referenz für die Bereitstellung von Mobilitätsdaten empfohlen wird. Damit ist sichergestellt, dass der VSPV-Standard 101 nicht nur den aktuellen gesetzlichen Vorgaben – etwa der deutschen Mobilitätsdatenverordnung und den Anforderungen an den Nationalen Zugangspunkt (NAP) – entspricht, sondern auch langfristig anschlussfähig an künftige europäische Entwicklungen bleibt.

Der VSPV-Standard 101 unterscheidet klar zwischen Erweiterungen innerhalb des definierten Profils und individuellen NeTEx-Extensionselementen. KeyValue-Profile fassen alle mittels keyList übertragbaren Zusatzattribute zusammen, die nicht im Kernmodell definiert, aber zwischen allen Beteiligten vereinbart sind. Die Schlüssel solcher KeyValue-Paare sind eindeutig festgelegt und dokumentiert, sodass zusätzliche Informationen (z. B. betriebliche Parameter) ohne Strukturbruch übermittelt werden können, ohne den Standard zu verlassen. Demgegenüber werden <Extensions>-Elemente nur eingesetzt, wenn eine erforderliche Information weder durch Standardfelder noch durch das vereinbarte KeyValue-Profil darstellbar ist. In solchen Fällen sind die neuen XML-Elemente mit einem eigenen Namensraum-Präfix (z. B. „vspv:“) zu versehen und in der Schnittstellendokumentation präzise zu beschreiben. Diese klare Abgrenzung stellt sicher, dass profilkonforme Erweiterungen (KeyValue) und individuelle Erweiterungen (Extensions) konsistent eingesetzt werden. Damit bleibt die

VDV-462-Konformität gewahrt, und gleichzeitig besteht die nötige Flexibilität, um innovative Anforderungen abzubilden, ohne die Interoperabilität zu gefährden.

Die enge Verzahnung mit der VDV 462 als nationalem Profil von NeTEx stellt sicher, dass die spezifischen Anforderungen des deutschen Marktes – etwa hinsichtlich Liniennummerierung, Tarifstrukturen, Betriebsdatenerfassung oder Fahrgastinformation – optimal berücksichtigt werden. Gleichzeitig erleichtert die Kompatibilität mit VDV 462 die Integration in bestehende IT-Systeme, Auskunfts- und Buchungsplattformen sowie die Zusammenarbeit mit Systemherstellern und Aufgabenträgern.

Ein weiterer strategischer Aspekt ist die Möglichkeit, Daten aus dem VSPV-Standard 101 über Konverter und Mappingtabellen in andere Formate wie GTFS oder GTFS-flex zu überführen. Dies gewährleistet, dass die im VSPV-Standard 101 modellierten Angebote nicht nur in nationalen und europäischen Plattformen, sondern auch in internationalen Routingdiensten, Kartenanwendungen und MaaS-Ökosystemen sichtbar und nutzbar sind. Die Pflege von Konvertierungswerkzeugen und die laufende Dokumentation der Zuordnungen sichern die Nachhaltigkeit und Reichweite der Daten.

Nicht zuletzt trägt die Wahl von NeTEx dazu bei, die Innovationsfähigkeit und Zukunftssicherheit des VSPV-Standards 101 zu gewährleisten. Die kontinuierliche Weiterentwicklung des Standards durch europäische und nationale Gremien, die Einbindung neuer Verkehrsformen, Technologien und regulatorischer Anforderungen sowie die Möglichkeit, bewährte Erweiterungen in zukünftige Standardversionen zu überführen, machen NeTEx zu einer tragfähigen Basis für die Digitalisierung und Standardisierung des Gelegenheitsverkehrs im ÖPNV.

Zusammenfassend zeigt sich, dass der VSPV-Standard 101 mit der Wahl von NeTEx und der VDV 462 als methodischem und technischem Fundament eine umfassende, zukunftsfähige und interoperable Lösung für die digitale Abbildung von Gelegenheitsverkehren im ÖPNV schafft. Die Stärken von NeTEx – insbesondere die hohe Modellierungstiefe, die Flexibilität, die Unterstützung für Versionierung, Mehrsprachigkeit und Erweiterungsmechanismen sowie die enge Verzahnung mit nationalen und europäischen Vorgaben – ermöglichen es, sowohl klassische Linienverkehre als auch moderne, flexible und on-demand-basierte Angebote präzise und konsistent abzubilden.

Gleichzeitig bleibt der VSPV-Standard 101 anschlussfähig an internationale Datenstandards wie GTFS und GTFS-flex, indem er die Möglichkeit zur Konvertierung und zum Mapping in diese Formate vorsieht. Damit wird sichergestellt, dass die in Nordrhein-Westfalen modellierten und bereitgestellten Mobilitätsdaten nicht nur im nationalen und europäischen Kontext, sondern auch auf globalen Plattformen und in internationalen Routingdiensten genutzt werden können. Die klare Abgrenzung zu Echtzeitstandards wie TRIAS und SIRI sorgt dafür, dass sowohl die langfristige Datenqualität als auch die Aktualität und Reaktionsfähigkeit der Systeme gewährleistet sind.

Der VSPV-Standard 101 nutzt die Synergien zwischen den verschiedenen Standards, indem er die Stärken von NeTEx/VDV 462 für die strukturierte, langfristige Modellierung und Pflege von Mobilitätsdaten mit den Vorteilen von GTFS für die internationale Sichtbarkeit und von TRIAS/SIRI für die Echtzeitkommunikation verbindet. Wo notwendig,

werden bewusste Abgrenzungen vorgenommen, um die Komplexität beherrschbar zu halten und die Interoperabilität zu sichern.

Mit dieser strategischen Ausrichtung ist der VSPV-Standard 101 bestens gerüstet, um den aktuellen und zukünftigen Anforderungen an die Digitalisierung, Standardisierung und Integration von Gelegenheitsverkehren im ÖPNV gerecht zu werden – sowohl auf regionaler als auch auf nationaler und internationaler Ebene.

## 3 Technische Konventionen

### 3.1 Einleitung und Zielsetzung

Kapitel 3 definiert die verbindlichen technischen Regeln für die Implementierung, Pflege und den Austausch der Daten im Rahmen des VSPV-Standards 101. Während Kapitel 2 die fachlichen Modelle und konzeptionellen Grundlagen beschreibt, werden in Kapitel 3 konkrete Vorgaben und Konventionen festgelegt. Diese technischen Konventionen stellen sicher, dass das entwickelte Datenmodell in der Praxis interoperabel, validierbar und betrieblich einsetzbar ist. Insbesondere wird darauf geachtet, dass alle relevanten Informationen zu Gelegenheitsverkehren konsistent und eindeutig abgebildet werden, um eine nahtlose Integration in verschiedene Systeme zu ermöglichen.

Zielsetzung:

- Sicherstellung der technischen Konsistenz und Interoperabilität aller Datenlieferungen im VSPV-Standard.
- Festlegung verbindlicher Regeln für IDs, Feldlängen, Zeichensätze, Austauschformate und Erweiterungen des Datenmodells.
- Klare Vorgaben zur Versionierung, zur Verwaltung von Gültigkeitszeiträumen und zur Qualitätssicherung der Daten.
- Erleichterung der Integration in bestehende Systemlandschaften sowie Anbindung externer Plattformen (z. B. Auskunftssysteme und Buchungssysteme).

Dieses Kapitel dient als technische Referenz für Entwickler, Systemintegratoren und Datenverantwortliche. Es bildet die Brücke zwischen der fachlichen Modellierung aus den vorherigen Kapiteln und der operativen Umsetzung in IT-Systemen. In den folgenden Unterabschnitten werden alle notwendigen technischen Konventionen detailliert behandelt – von der eindeutigen Identifizierung und Strukturierung der Daten über die Modellierung flexibler Bedienformen (virtuelle Haltestellen, Bediengebiete, Zeitfenster) und die Einbindung von Buchungs- und Zahlungsprozessen bis hin zu Erweiterungsmechanismen und Schnittstellen zu Echtzeitsystemen. Durch die konsequente Anwendung dieser Regeln wird gewährleistet, dass VSPV-konforme Implementierungen robust, zukunftsfähig und vollständig interoperabel sind.

### 3.2 Persistente Identifikatoren und ID-Schemata

Die eindeutige Identifikation sämtlicher Datenobjekte ist eine der zentralen technischen Anforderungen im VSPV-Standard 101. Nur durch konsistente, unverwechselbare und dauerhaft stabile Identifikatoren (IDs) kann sichergestellt werden, dass alle Systeme –

von der Datenerzeugung über die Verarbeitung bis hin zum Austausch mit Dritten – zuverlässig und interoperabel auf dieselben Objekte referenzieren. Die Vergabe und der Aufbau dieser IDs folgt daher strengen Konventionen, die sich an den Vorgaben der VDV 462 und des NeTEx-Standards orientieren.

Jedes Objekt, das im Datenmodell abgebildet wird – sei es eine Haltestelle, ein Bediengebiet, eine Fahrt, ein Tarif oder ein Buchungsprozess – erhält einen persistenten, systemweit eindeutigen Identifikator. Diese IDs sind so gestaltet, dass sie auch beim Austausch zwischen unterschiedlichen Systemen, bei Migrationen oder in föderierten Datenlandschaften ihre Eindeutigkeit behalten. Die Struktur der Identifikatoren folgt einem festen Schema, das aus mehreren Komponenten besteht:

- **Namensraum-Präfix:**

Das Präfix kennzeichnet die Organisation, das System oder die Region, die für die Vergabe der ID verantwortlich ist. Beispiele sind „VSPV:“, „NRW:“, „DIVA:“, „SDGV:“. Der Namensraum ist zentral zu dokumentieren und mit allen beteiligten Partnern abzustimmen, um Kollisionen zu vermeiden. Um die globale Eindeutigkeit zu gewährleisten, sollte das Präfix IFOPT-konform gestaltet sein (z. B. inklusive Ländercode wie „DE“ für deutsche IDs)

- **Objekttyp-Kennung:**

Direkt nach dem Präfix folgt eine Kennzeichnung des Objekttyps, etwa „StopPlace“, „FlexibleServiceJourney“, „TariffZone“, „BookingProcess“. Diese Kennung macht die ID sprechend und erleichtert die manuelle Nachverfolgung und Fehlersuche.

- **Laufende Nummer oder eindeutiger Schlüssel:**

Der dritte Bestandteil ist eine fortlaufende Nummer, ein eindeutiger Schlüssel oder eine Kombination aus beidem. Sie sorgt dafür, dass jedes Objekt innerhalb seines Namensraums und Objekttyps eindeutig identifiziert werden kann.

Ein typisches Beispiel für eine solche ID ist

```
<FlexibleStopPlace id="VSPV:StopPlace:NRW-1001" version="1.0">
  <!-- weitere Attribute -->
</FlexibleStopPlace>
```

für einen dynamischen Haltepunkt oder

```
<ServiceJourney id="VSPV:ServiceJourney:OB-2025-001" version="2.0">
  <!-- weitere Attribute -->
</ServiceJourney>
```

für eine Fahrt.

Gemäß VDV 462 ist zwischen einer Haltestelle (StopPlace) und den konkreten Einstiegspunkten/Steigen (Quays) zu unterscheiden, die jeweils eigene eindeutige Identifikatoren erhalten.

Die IDs sind dauerhaft und stabil zu vergeben. Einmal vergebene Identifikatoren dürfen nicht wiederverwendet oder nachträglich verändert werden. Änderungen am Objekt, wie

etwa die Anpassung eines Bedienegebiets oder die Aktualisierung von Fahrplandaten, führen zur Vergabe einer neuen Version, nicht zu einer neuen ID. Die Versionierung erfolgt über ein separates Attribut, sodass die Historie und Entwicklung jeder Entität lückenlos nachvollziehbar bleibt.

Für die Pflege und Vergabe der Identifikatoren empfiehlt sich die zentrale Dokumentation, idealerweise in einer organisationsweiten ID-Registry. Dies erleichtert die Verwaltung, verhindert Kollisionen und unterstützt die Harmonisierung bei der Integration externer Datenquellen. Bei der Integration von Fremddaten ist darauf zu achten, dass Namensräume und Präfixe abgestimmt werden, um Mehrdeutigkeiten auszuschließen.

Die konsequente Anwendung dieser Namens- und ID-Konventionen ist die Grundlage für die Referenzierbarkeit, Wiederverwendbarkeit und Interoperabilität der Daten im gesamten VSPV-Standard 101. Sie ermöglicht es, komplexe Beziehungen zwischen Objekten zuverlässig abzubilden und ist ein unverzichtbares Fundament für alle weiteren technischen und fachlichen Prozesse der Modellierung und des Datenaustauschs.

Die korrekte Vergabe und Pflege von Identifikatoren ist nicht nur für die technische Interoperabilität, sondern auch für die langfristige Wartbarkeit und Erweiterbarkeit des Datenmodells von zentraler Bedeutung. In der Praxis treten dabei verschiedene Sonderfälle und Herausforderungen auf, die im VSPV-Standard 101 durch zusätzliche Konventionen und Empfehlungen adressiert werden.

Ein häufiges Problem ist die Integration von Daten aus unterschiedlichen Quellen oder Systemen, etwa wenn mehrere Verkehrsunternehmen, Aufgabenträger oder externe Dienstleister gemeinsam an einem Mobilitätsangebot arbeiten. Um Konflikte und Überschneidungen bei der Vergabe von IDs zu vermeiden, ist es unerlässlich, dass alle beteiligten Partner ihre Namensräume und Präfixe frühzeitig abstimmen und zentral dokumentieren. Für größere Projekte empfiehlt sich die Einrichtung einer gemeinsamen ID-Registry, in der alle vergebenen Identifikatoren, Namensräume und Objekttypen eindeutig erfasst und gepflegt werden. So lassen sich Kollisionen und Inkonsistenzen bereits im Vorfeld vermeiden.

Ein weiterer Spezialfall betrifft die Migration von Bestandsdaten aus Altsystemen oder anderen Standards (z. B. VDV 452, DIVA, GTFS). Hier ist darauf zu achten, dass bestehende IDs – sofern sie bereits eindeutig und stabil vergeben wurden – beibehalten oder eindeutig in das neue Schemata überführt werden. Falls dies nicht möglich ist, müssen neue IDs nach dem VSPV-Schema vergeben und die Zuordnung zu den alten Identifikatoren dokumentiert werden. Für die Nachverfolgbarkeit empfiehlt es sich, alte IDs als zusätzliche Attribute (z. B. `legacyId`) im Datenmodell zu hinterlegen. Hinweis: Alte IDs sollten in NeTEx-konformer Weise hinterlegt werden, z. B. als `PrivateCode` oder als Schlüssel-Wert-Eintrag im `KeyList`-Abschnitt des Objekts, um die Schema-Konformität zu wahren.

Auch bei der Versionierung von Objekten sind einige Besonderheiten zu beachten. Änderungen am Objekt, die lediglich technische Korrekturen (z. B. Tippfehler in der Beschreibung) betreffen, können in der Regel durch eine neue Version abgebildet werden, ohne dass die ID geändert wird. Bei grundlegenden Änderungen – etwa der Neudefinition

eines Bedienegebiets oder der Zusammenlegung von Haltestellen – ist hingegen die Vergabe einer neuen ID erforderlich, um die Integrität der Referenzen zu wahren. In solchen Fällen sollte die Beziehung zwischen alten und neuen Objekten durch entsprechende Verweise (z. B. replaces, isReplacedBy) dokumentiert werden.

Best Practices für die Vergabe und Pflege von IDs umfassen:

- Die konsequente Nutzung sprechender, nachvollziehbarer Namensbestandteile (z. B. Region, Objekttyp, laufende Nummer).
- Die zentrale Dokumentation und Pflege aller vergebenen IDs, idealerweise in einer maschinenlesbaren Registry.
- Die regelmäßige Überprüfung auf Kollisionen, Dubletten und verwaiste Referenzen.
- Die transparente Dokumentation von Änderungen, Migrationen und Versionierungen.
- Die Schulung aller beteiligten Akteure im Umgang mit den ID-Konventionen und der Registry.

Fehlerquellen, die es zu vermeiden gilt, sind unter anderem:

- Die mehrfache Vergabe derselben ID an unterschiedliche Objekte.
- Die nachträgliche Änderung oder Wiederverwendung bereits vergebener IDs.
- Die Nutzung nicht abgestimmter Präfixe oder Namensräume.
- Die unvollständige Dokumentation von Änderungen, Migrationen oder Löschungen.

Durch die konsequente Anwendung dieser Best Practices und Konventionen wird sichergestellt, dass das Datenmodell des VSPV-Standards 101 auch bei wachsender Komplexität, zunehmender Anzahl von Partnern und fortlaufender Weiterentwicklung stabil, interoperabel und wartbar bleibt.

### **Spezialfälle bei der ID-Vergabe**

Ein typischer Spezialfall ist die Übernahme von Daten aus unterschiedlichen, historisch gewachsenen Systemen, bei denen keine durchgängig einheitlichen oder sprechenden IDs existieren. In solchen Fällen empfiehlt es sich, einen eigenen Migrations-Namensraum zu definieren (z. B. „VSPV:MIG:“), um die Herkunft der Objekte im Datenmodell sichtbar zu machen. Zusätzlich sollte die ursprüngliche, im Altsystem verwendete ID als separates Attribut (z. B. <legacyId>) mitgeführt werden. So bleibt die Nachverfolgbarkeit auch nach der Migration erhalten und Rückfragen oder Zuordnungen zu Altbeständen sind jederzeit möglich.

Ein weiteres Beispiel betrifft die situationsbedingte Zusammenlegung oder Aufteilung von Objekten, etwa wenn zwei Bedienegebiete zu einem neuen zusammengeführt werden oder ein großes Bedienegebiet in mehrere kleinere Einheiten aufgeteilt wird. In diesen Fällen ist

für jedes neue Objekt eine neue, eindeutige ID zu vergeben. Zur Dokumentation der Beziehung zwischen alten und neuen Objekten empfiehlt sich die Nutzung von Referenzattributen wie <replaces> oder <isReplacedBy>. Damit können auch externe Systeme oder nachgelagerte Prozesse nachvollziehen, wie sich die Objektlandschaft im Zeitverlauf verändert hat.

Zur technischen Umsetzung solcher Verweise sollte das NeTeX-Extensionsschema genutzt werden. Beispielsweise kann das nachfolgende Objekt im <Extensions>-Block einen Verweis auf die ID des Vorgänger-Objekts enthalten (entsprechend replaces/isReplacedBy). Dadurch bleibt die Ablöse-Beziehung auch im Datenaustausch eindeutig nachvollziehbar.

### **Fehlerbehandlung und Konfliktlösung**

Tritt bei der Datenintegration oder im laufenden Betrieb ein Konflikt durch doppelt vergebene oder widersprüchliche IDs auf, sollte das System die Datenlieferung automatisiert ablehnen und einen Fehlerbericht mit Angabe der betroffenen IDs und Objekte erzeugen. Die Korrektur erfolgt dann durch die verantwortliche Organisation, indem die Kollision aufgelöst und die betroffenen Objekte mit neuen, eindeutigen IDs versehen werden. Eine regelmäßige, automatisierte Prüfung auf ID-Kollisionen und verwaiste Referenzen ist als Teil der Qualitätssicherung zu etablieren.

Ein weiteres Fehlerbild ist die versehentliche Löschung oder Deaktivierung von Objekten, auf die noch von anderen Datenstrukturen referenziert wird. In solchen Fällen muss das System entweder die Löschung verhindern oder die betroffenen Referenzen automatisch anpassen (z. B. durch einen Verweis auf einen Nachfolger oder eine Ersatzstruktur). Die Historisierung und Versionierung aller Objekte erlaubt es, versehentlich gelöschte oder geänderte Datenstände wiederherzustellen und die Datenkonsistenz zu sichern.

### **Praxisbeispiel 1: Migration von Haltestellen aus Altsystemen**

Angenommen, ein Verkehrsunternehmen migriert seine Haltestellen aus einem Altsystem mit numerischen IDs nach VSPV-Standard. Die ursprüngliche ID „4711“ wird im neuen Modell als

```
<StopPlace id="VSPV:StopPlace:NRW-4711" version="1.0">  
  <legacyId>4711</legacyId>  
  <!-- weitere Attribute -->  
</StopPlace>
```

abgebildet. Sollte diese Haltestelle später mit einer anderen zusammengelegt werden, erhält das neue Objekt eine frische ID, und die alten IDs werden im Attribut <replaces> dokumentiert.

### **Praxisbeispiel 2: Konfliktlösung bei ID-Kollision**

Wird bei der Integration von Daten aus zwei unterschiedlichen Partnerregionen versehentlich die ID „VSPV:FlexibleServiceJourney:1001“ doppelt vergeben, erkennt das System dies beim Import und gibt eine Fehlermeldung aus. Die verantwortlichen Partner

stimmen dann ab, welcher Datensatz eine neue, eindeutige ID erhält, und dokumentieren die Änderung in der zentralen Registry.

### Praxisbeispiel 3: Historisierung und Nachverfolgbarkeit

Ein Bediengebiet wird zum 1. Januar 2026 erweitert. Die ursprüngliche Version bleibt mit

```
<FlexibleStopPlace id="VSPV:StopPlace:NRW-1001" version="1.0">
  <ValidBetween>
    <FromDate>2025-06-01</FromDate>
    <ToDate>2025-12-31</ToDate>
  </ValidBetween>
  <!-- weitere Attribute -->
</FlexibleStopPlace>
```

im Datenbestand erhalten. Die erweiterte Version wird als

```
<FlexibleStopPlace id="VSPV:StopPlace:NRW-1001" version="2.0">
  <ValidBetween>
    <FromDate>2026-01-01</FromDate>
  </ValidBetween>
  <!-- neue Geometrie, weitere Attribute -->
</FlexibleStopPlace>
```

abgelegt. So bleibt die Entwicklung des Objekts über die Zeit nachvollziehbar und auskunftsfähig.

Weitere Gelegenheitsverkehrsspezifische Modellierungen in beispielhafter Form:

#### 1. Flexibles Bediengebiet (Flughafen Köln/Bonn)

Ein Taxibetrieb bedient den Flughafen Köln/Bonn als dynamisches Bediengebiet. Das Gebiet wird als Polygon mit Koordinaten abgebildet.

```
<FlexibleStopPlace id="VSPV:StopPlace:CGN-Airport-Taxi" version="1.0">
  <Name>
    <Text language="de">Taxi-Bediengebiet Flughafen Köln/Bonn</Text>
  </Name>
  <Polygon>
    <posList>
      50.8658 7.1427 50.8662 7.1435 50.8655 7.1450 <!-- Weitere
Koordinaten -->
    </posList>
  </Polygon>
  <ValidBetween>
    <FromDate>2025-01-01</FromDate>
  </ValidBetween>
  <keyList>
    <KeyValue>
      <Key>BEFÖRDERUNGSART</Key>
      <Value>PBefG§47</Value>
    </KeyValue>
  </keyList>
</FlexibleStopPlace>
```

```
</KeyValue>
</keyList></FlexibleStopPlace>
```

## 2. Buchungsprozess mit Vorlaufzeit und Authentifizierung

Taxi-Buchung über App (OAuth2) oder Telefon, mind. 30 Minuten Vorlaufzeit:

```
<BookingProcess id="VSPV:BookingProcess:Taxi-CGN-1" version="1.0">
  <Name>
    <Text language="de">Taxibuchung Flughafen Köln/Bonn</Text>
  </Name>
  <latestBookingTime>PT30M</latestBookingTime>
  <bookingMethods>mobileApp</bookingMethods>
  <bookingAccess>
    <MobileAppAccess>
      <AuthenticationMethod>OAuth2.0</AuthenticationMethod>
      <Url>https://taxi-cgn.de/oauth</Url>
    </MobileAppAccess>
  </bookingAccess>
  <keyList>
    <KeyValue>
      <Key>BEFÖRDERUNGSART</Key>
      <Value>PBefG$47</Value>
    </KeyValue>
  </keyList>
</BookingProcess>
```

## 3. Dynamischer Tarif mit Festpreisoption

Tarif für eine Taxifahrt vom Flughafen ins Stadtzentrum (Festpreis 35 € oder Taxameter):

```
<FareStructure id="VSPV:Fare:CGN-City-Fixed" version="1.0">
  <Name>
    <Text language="de">Festpreis Flughafen - Innenstadt</Text>
  </Name>
  <TypeOfTariffRef ref="VSPV:Tariff:Taxi-Fixed"/>
  <PriceGroups>
    <PriceGroup id="VSPV:PriceGroup:CGN-City">
      <Amount>35.00</Amount>
      <Currency>EUR</Currency>
      <PriceType>fixed</PriceType>
    </PriceGroup>
  </PriceGroups>
  <ValidBetween>
    <FromDate>2025-01-01</FromDate>
```

```

</ValidBetween>
<keyList>
  <KeyValue>
    <Key>BEFÖRDERUNGSART</Key>
    <Value>PBefG847</Value>
  </KeyValue>
</keyList></FareStructure>

```

#### 4. Fahrzeug mit Rollstuhlzugang und E-Auto-Kennzeichnung

```

<VehicleType id="VSPV:VehicleType:WAV-E" version="1.0">
  <Name>
    <Text language="de">Rollstuhlgerechtes E-Taxi</Text>
  </Name>
  <keyList>
    <KeyValue>
      <Key>BEFÖRDERUNGSART</Key>
      <Value>PBefG847</Value>
    </KeyValue>
    <KeyValue>
      <Key>WHEELCHAIR_ACCESSIBLE</Key>
      <Value>>true</Value>
    </KeyValue>
    <KeyValue>
      <Key>VEHICLE_PROPULSION</Key>
      <Value>electric</Value>
    </KeyValue>
  </keyList></VehicleType>

```

#### 5. Ridepooling-Service mit Kapazitätsangabe

```

<FlexibleServiceJourney id="VSPV:ServiceJourney:Pool-2025-001"
version="1.0">
  <Name>
    <Text language="de">Köln Ridepooling (max. 4 Pers.)</Text>
  </Name>
  <TransportMode>taxi</TransportMode>
  <passengerCapacity>4</passengerCapacity>
  <keyList>
    <KeyValue>
      <Key>POOLING_ALLOWED</Key>
      <Value>>true</Value>
    </KeyValue>
    <KeyValue>
      <Key>MAX_DETOUR_TIME</Key>

```

```

    <Value>PT15M</Value> <!-- Max. 15 Minuten Umweg -->
  </KeyValue>
</keyList>
</FlexibleServiceJourney>

```

### Erläuterungen:

- **FlexibleStopPlace:** Definiert das geografische Gebiet, in dem Taxis ohne feste Haltestellen gebucht werden können.
- **BookingProcess:** Legt Buchungsmodalitäten fest (Methoden, Fristen, Authentifizierung).
- **FareStructure:** Bietet Optionen für Festpreise oder dynamische Berechnung (hier: Festpreis).
- **VehicleType:** Kennzeichnet spezielle Fahrzeugmerkmale (Rollstuhlzugang, E-Antrieb).
- **FlexibleServiceJourney:** Modelliert Ridepooling-Dienste mit Kapazitätsgrenzen und Umwegregeln.

Ein typisches Szenario für die Digitalisierung des Gelegenheitsverkehrs im ländlichen Raum Westfalens ist die Taxibuchung als Kioskbuchung einer noch einzurichtenden modernen Variante der Taxirufsäule – hier im Beispiel am Bahnhof Wetter (Ruhr) – und die nahtlose Verknüpfung mit dem Linienbusverkehr. Gerade an kleineren Bahnhöfen, wo nicht dauerhaft Taxis bereitstehen, bietet ein digitaler Kiosk einen niedrighschwelligigen Zugang für alle Fahrgastgruppen, insbesondere für Menschen ohne Smartphone oder mit eingeschränkter digitaler Kompetenz.

Der Kiosk wird im Datenmodell als eigenständiger Haltepunkt abgebildet, der räumlich eindeutig dem Bahnhof Wetter (Ruhr) zugeordnet ist. Die Geokoordinaten werden als Centroid im StopPlace-Objekt hinterlegt. Beispielhaft sieht die Modellierung so aus:

```

<StopPlace id="VSPV:StopPlace:Wetter-Kiosk" version="1.0">
  <Name>
    <Text language="de">Taxikiosk Bahnhof Wetter</Text>
  </Name>
  <Centroid>
    <Location>
      <Longitude>7.3950</Longitude>
      <Latitude>51.3875</Latitude>
    </Location>
  </Centroid>
  <ValidBetween>
    <FromDate>2025-01-01</FromDate>
  </ValidBetween>
</StopPlace>

```

Der zugehörige Buchungsprozess für eine Taxifahrt wird als BookingProcess modelliert. Hierbei werden die Buchungswege – Kiosk und Telefon – sowie die Öffnungszeiten des

Kiosks und die Telefonnummer des lokalen Taxiunternehmens hinterlegt. Es ist kein Vorlauf erforderlich, da die Buchung sofort ausgelöst werden kann:

```
<BookingProcess id="VSPV:BookingProcess:Taxi-Wetter" version="1.0">
  <Name>
    <Text language="de">Taxibuchung am Kiosk Wetter</Text>
  </Name>
  <bookingMethods>kiosk, telephone</bookingMethods>
  <latestBookingTime>PT0M</latestBookingTime>
  <bookingAccess>
    <KioskAccess>
      <StopPlaceRef ref="VSPV:StopPlace:Wetter-Kiosk"/>
      <OpeningTimes>
        <OpeningTime>
          <FromTime>06:00:00</FromTime>
          <ToTime>22:00:00</ToTime>
        </OpeningTime>
      </OpeningTimes>
    </KioskAccess>
    <TelephoneAccess>
      <PhoneNumber>02335 66666</PhoneNumber>
    </TelephoneAccess>
  </bookingAccess>
</BookingProcess>
```

Der eigentliche Fahrtwunsch – beispielsweise von Wetter (Ruhr) nach Volmarstein – wird als FlexibleServiceJourney abgebildet. Hier werden Start- und Zielhaltepunkte, die Kapazität des Fahrzeugs sowie der Festpreis nach regionalem Taxitarif modelliert. Der Festpreis kann als Key-Value-Attribut hinterlegt werden:

```
<FlexibleServiceJourney id="VSPV:ServiceJourney:Taxi-Wetter-
Volmarstein" version="1.0">
  <Name>
    <Text language="de">Taxi Wetter → Volmarstein</Text>
  </Name>
  <TransportMode>taxi</TransportMode>
  <passengerCapacity>4</passengerCapacity>
  <JourneyPattern>
    <StopPointInJourneyPattern>
      <Order>1</Order>
      <StopPointRef ref="VSPV:StopPlace:Wetter-Kiosk"/>
    </StopPointInJourneyPattern>
    <StopPointInJourneyPattern>
      <Order>2</Order>
      <StopPointRef ref="VSPV:StopPlace:Volmarstein-Bushalt"/>
    </StopPointInJourneyPattern>
  </JourneyPattern>
```

```

<keyList>
  <KeyValue>
    <Key>FESTPREIS</Key>
    <Value>27.00</Value>
  </KeyValue>
</keyList>
</FlexibleServiceJourney>

```

Die Verknüpfung mit dem Linienverkehr erfolgt durch die Modellierung der Anschlussfahrt mit der (fiktiven) Buslinie 559, die von Volmarstein nach Hagen Hauptbahnhof führt. Die Bushaltestellen werden als reguläre StopPlace-Objekte geführt, die Fahrt als ServiceJourney:

```

<ServiceJourney id="VSPV:ServiceJourney:Bus-559" version="1.0">
  <Name>
    <Text language="de">Bus 559 Volmarstein → Hagen Hbf</Text>
  </Name>
  <LineRef ref="VSPV:Line:Bus-559"/>
  <DepartureTime>09:30:00</DepartureTime>
  <JourneyPattern>
    <StopPointInJourneyPattern>
      <Order>1</Order>
      <StopPointRef ref="VSPV:StopPlace:Volmarstein-Bushalt"/>
    </StopPointInJourneyPattern>
    <StopPointInJourneyPattern>
      <Order>2</Order>
      <StopPointRef ref="VSPV:StopPlace:Hagen-Hbf"/>
    </StopPointInJourneyPattern>
  </JourneyPattern>
</ServiceJourney>

```

Für die Fahrgäste wird ein durchgehendes Kombi-Ticket angeboten, das beide Fahrtabschnitte – Taxi und Bus – umfasst. Die Tarifstruktur wird als FareStructure modelliert. Der Gesamtpreis (z. B. 30 €) und die Bedingungen (z. B. Gültigkeit nur mit Deutschlandticket) werden als Attribute hinterlegt:

```

<FareStructure id="VSPV:Fare:Taxi-Bus-Combi" version="1.0">
  <Name>
    <Text language="de">Kombi-Ticket Taxi + Bus</Text>
  </Name>
  <TypeOfTariffRef ref="VSPV:Tariff:WestfalenTarif"/>
  <PriceGroups>
    <PriceGroup id="VSPV:PriceGroup:Combi-Wetter">
      <Amount>30.00</Amount>
      <Currency>EUR</Currency>
      <PriceType>fixed</PriceType>
    </PriceGroup>
  </PriceGroups>

```

```

<conditions>
  <Condition>
    <Text language="de">Gilt nur mit gültigem Deutschlandticket</Text>
  </Condition>
</conditions>
<ValidBetween>
  <FromDate>2025-01-01</FromDate>
</ValidBetween>
</FareStructure>

```

Im praktischen Ablauf bucht der Fahrgast am Kiosk am Bahnhof Wetter (Ruhr) eine Fahrt nach Volmarstein, erhält ein Kombi-Ticket, das sowohl für die Taxifahrt als auch für den Anschlussbus gilt, und kann die Reise ohne Medienbruch fortsetzen. Die Integration des Taxibuchungsprozesses, der Fahrten und der Tarifstruktur in einer konsistenten Datenbasis nach VSPV-Standard 101 ermöglicht eine durchgehende digitale Verarbeitung, eine einfache Abrechnung und eine zuverlässige Fahrgastinformation – sowohl für spontane als auch für vorab geplante Reisen im ländlichen Raum.

Ein zentrales Anwendungsbeispiel für die Leistungsfähigkeit des VSPV-Standards 101 ist die durchgehende digitale Abbildung einer Mobilitätskette, bei der verschiedene Verkehrsträger – Taxi, Bus, Regionalexpress und erneut Taxi – in einer einzigen App geplant, gebucht und abgerechnet werden. Im Sinne eines modernen MaaS-Ansatzes (Mobility-as-a-Service) ist es dabei entscheidend, dass der Fahrgast nicht mehrere Apps oder Tickets benötigt, sondern die gesamte Reisekette mit einem einzigen digitalen Vorgang abwickeln kann.

Die folgende Modellierung zeigt, wie eine solche Mobilitätskette von Hagen Bathey über Dortmund Syburg, Dortmund Hörde, Arnsberg Bahnhof bis zum Marktplatz Arnsberg im VSPV-Standard 101 abgebildet wird:

## 1. Modellierung der Haltepunkte (StopPlace)

### a) Hagen Bathey (Startpunkt)

```

<StopPlace id="VSPV:StopPlace:Hagen-Bathey" version="1.0">
  <Name>
    <Text language="de">Hagen Bathey (Gewerbegebiet)</Text>
  </Name>
  <Centroid>
    <Location>
      <Longitude>7.4714</Longitude>
      <Latitude>51.3672</Latitude>
    </Location>
  </Centroid>
</StopPlace>

```

### b) Dortmund Syburg (Taxi-Ziel & Bus-Start)

```
<StopPlace id="VSPV:StopPlace:Dortmund-Syburg" version="1.0">
  <Name>
    <Text language="de">Dortmund-Syburg</Text>
  </Name>
  <Centroid>
    <Location>
      <Longitude>7.4850</Longitude>
      <Latitude>51.4250</Latitude>
    </Location>
  </Centroid>
</StopPlace>
```

c) Dortmund Hörde Bf (Bus-Ziel & Zug-Start)

```
<StopPlace id="VSPV:StopPlace:Dortmund-Hoerde-Bf" version="1.0">
  <Name>
    <Text language="de">Dortmund Hörde Bahnhof</Text>
  </Name>
  <Centroid>
    <Location>
      <Longitude>7.5010</Longitude>
      <Latitude>51.4857</Latitude>
    </Location>
  </Centroid>
</StopPlace>
```

d) Arnsberg Bf (Zug-Ziel & Taxi-Start)

```
<StopPlace id="VSPV:StopPlace:Arnsberg-Bf" version="1.0">
  <Name>
    <Text language="de">Arnsberg Bahnhof</Text>
  </Name>
  <Centroid>
    <Location>
      <Longitude>8.0639</Longitude>
      <Latitude>51.3983</Latitude>
    </Location>
  </Centroid>
</StopPlace>
```

e) Arnsberg Marktplatz (Taxi-Ziel)

```
<StopPlace id="VSPV:StopPlace:Arnsberg-Marktplatz" version="1.0">
  <Name>
    <Text language="de">Arnsberg Innenstadt (Marktplatz)</Text>
  </Name>
```

```

<Centroid>
  <Location>
    <Longitude>8.0645</Longitude>
    <Latitude>51.3967</Latitude>
  </Location>
</Centroid>
</StopPlace>

```

## 2. Einheitlicher Buchungsprozess für die gesamte Kette

Alle Teilstrecken – unabhängig vom Verkehrsträger – referenzieren denselben Buchungsprozess, der die MaaS-App beschreibt. Die Attribute legen fest, dass die Buchung per App erfolgt, und verweisen auf die entsprechende Plattform.

```

<BookingProcess id="VSPV:BookingProcess:MaaS-Westfalen"
version="1.0">
  <Name>
    <Text language="de">WestfalenMobil-App</Text>
  </Name>
  <bookingMethods>mobileApp</bookingMethods>
  <bookingAccess>
    <MobileAppAccess>
      <AuthenticationMethod>OAuth2.0</AuthenticationMethod>
      <Url>https://westfalenmobil.de/app</Url>
    </MobileAppAccess>
  </bookingAccess>
  <latestBookingTime>PT0M</latestBookingTime>
</BookingProcess>

```

## 3. Modellierung der einzelnen Teilstrecken

a) Taxi: Hagen Bathey → Dortmund Syburg

```

<FlexibleServiceJourney id="VSPV:ServiceJourney:Taxi-Hagen-Syburg"
version="1.0">
  <Name>
    <Text language="de">Taxi Hagen Bathey → Dortmund Syburg</Text>
  </Name>
  <TransportMode>taxi</TransportMode>
  <JourneyPattern>
    <StopPointInJourneyPattern>
      <Order>1</Order>
      <StopPointRef ref="VSPV:StopPlace:Hagen-Bathey"/>
    </StopPointInJourneyPattern>
    <StopPointInJourneyPattern>
      <Order>2</Order>
      <StopPointRef ref="VSPV:StopPlace:Dortmund-Syburg"/>
    </StopPointInJourneyPattern>
  </JourneyPattern>
</FlexibleServiceJourney>

```

```

</StopPointInJourneyPattern>
</JourneyPattern>
<BookingProcessRef ref="VSPV:BookingProcess:MaaS-Westfalen"/>
<keyList>
  <KeyValue>
    <Key>FESTPREIS</Key>
    <Value>40.00</Value>
  </KeyValue>
</keyList>
</FlexibleServiceJourney>

```

b) Bus: Dortmund Syburg → Dortmund Hörde Bf (Linie 432)

```

<ServiceJourney id="VSPV:ServiceJourney:Bus-432" version="1.0">
  <Name>
    <Text language="de">Bus 432 Syburg → Hörde Bf</Text>
  </Name>
  <LineRef ref="VSPV:Line:Bus-432"/>
  <JourneyPattern>
    <StopPointInJourneyPattern>
      <Order>1</Order>
      <StopPointRef ref="VSPV:StopPlace:Dortmund-Syburg"/>
    </StopPointInJourneyPattern>
    <StopPointInJourneyPattern>
      <Order>2</Order>
      <StopPointRef ref="VSPV:StopPlace:Dortmund-Hoerde-Bf"/>
    </StopPointInJourneyPattern>
  </JourneyPattern>
  <BookingProcessRef ref="VSPV:BookingProcess:MaaS-Westfalen"/>
</ServiceJourney>

```

c) Zug: Dortmund Hörde Bf → Arnsberg Bf (RE57)

```

<ServiceJourney id="VSPV:ServiceJourney:RE57" version="1.0">
  <Name>
    <Text language="de">RE57 Dortmund Hörde → Arnsberg</Text>
  </Name>
  <LineRef ref="VSPV:Line:RE57"/>
  <JourneyPattern>
    <StopPointInJourneyPattern>
      <Order>1</Order>
      <StopPointRef ref="VSPV:StopPlace:Dortmund-Hoerde-Bf"/>
    </StopPointInJourneyPattern>
    <StopPointInJourneyPattern>
      <Order>2</Order>

```

```

    <StopPointRef ref="VSPV:StopPlace:Arnsberg-Bf"/>
  </StopPointInJourneyPattern>
</JourneyPattern>
<BookingProcessRef ref="VSPV:BookingProcess:MaaS-Westfalen"/>
</ServiceJourney>

```

#### d) Taxi: Arnsberg Bf → Arnsberg Marktplatz

```

<FlexibleServiceJourney id="VSPV:ServiceJourney:Taxi-Arnsberg-Markt"
version="1.0">
  <Name>
    <Text language="de">Taxi Arnsberg Bf → Marktplatz</Text>
  </Name>
  <TransportMode>taxi</TransportMode>
  <JourneyPattern>
    <StopPointInJourneyPattern>
      <Order>1</Order>
      <StopPointRef ref="VSPV:StopPlace:Arnsberg-Bf"/>
    </StopPointInJourneyPattern>
    <StopPointInJourneyPattern>
      <Order>2</Order>
      <StopPointRef ref="VSPV:StopPlace:Arnsberg-Marktplatz"/>
    </StopPointInJourneyPattern>
  </JourneyPattern>
  <BookingProcessRef ref="VSPV:BookingProcess:MaaS-Westfalen"/>
  <keyList>
    <KeyValue>
      <Key>FESTPREIS</Key>
      <Value>8.00</Value>
    </KeyValue>
  </keyList>
</FlexibleServiceJourney>

```

#### 4. Kombierter Tarif für die gesamte Mobilitätskette

Die Tarifstruktur kann als Kombi-Tarif für die gesamte Strecke abgebildet werden. Die Preisgruppen und Komponenten werden auf die einzelnen Teilstrecken referenziert.

```

<FareStructure id="VSPV:Fare:Kombi-Hagen-Arnsberg" version="1.0">
  <Name>
    <Text language="de">Kombi-Tarif Hagen Bathey → Arnsberg
Marktplatz</Text>
  </Name>
  <TypeOfTariffRef ref="VSPV:Tariff:WestfalenMaaS"/>
  <PriceGroups>

```

```

<PriceGroup id="VSPV:PriceGroup:Gesamtpreis">
  <Amount>69.00</Amount>
  <Currency>EUR</Currency>
  <PriceType>fixed</PriceType>
</PriceGroup>
</PriceGroups>
<components>
  <Component>
    <ServiceJourneyRef ref="VSPV:ServiceJourney:Taxi-Hagen-Syburg"/>
  </Component>
  <Component>
    <ServiceJourneyRef ref="VSPV:ServiceJourney:Bus-432"/>
  </Component>
  <Component>
    <ServiceJourneyRef ref="VSPV:ServiceJourney:RE57"/>
  </Component>
  <Component>
    <ServiceJourneyRef ref="VSPV:ServiceJourney:Taxi-Arnsberg-Markt"/>
  </Component>
</components>
</FareStructure>

```

## 5. Erläuterung des Nutzererlebnisses

Der Fahrgast plant und bucht die gesamte Reisekette von Hagen Bathey bis zum Marktplatz Arnsberg in einer einzigen App. Die App zeigt die optimale Verbindung an, reserviert alle Teilstrecken, stellt ein durchgehendes Ticket aus und wickelt die Bezahlung für die gesamte Mobilitätskette ab. Während der Reise erhält der Fahrgast Echtzeitinformationen zu Anschlusszeiten und Fahrzeugpositionen. Im Hintergrund sorgt der VSPV-Standard 101 für die Interoperabilität, die konsistente Datenhaltung und die Abrechnung zwischen den beteiligten Verkehrsunternehmen und Taxi-Anbietern.

## Mappingtabelle für Kapitel 3.2

Fachliche Anforderung	Technische Datenstruktur	Beispiel-XML	Konventionen
Buchung einer Taxifahrt via MaaS-App	BookingProcess mit mobileApp-Methode	<BookingProcess id="VSPV:BookingProcess:MaaS-Westfalen">...</BookingProcess>	Pflichtfelder: bookingMethods, MobileAppAccess/Url, latestBookingTime
Kombinierter Tarif für alle Teilstrecken	FareStructure mit components-Element	<FareStructure><components><Component><ServiceJourneyRef .../></Component></components></FareStructure>	Jede Teilstrecke muss referenziert werden.

Fachliche Anforderung	Technische Datenstruktur	Beispiel-XML	Konventionen
Echtzeit-Fahrzeugpositionen	SIRI-VM (VehicleMonitoring) + NeTeX-RT	<VehicleActivity><MonitoredVehicleJourney>...<VehicleLocation>...</VehicleLocation></MonitoredVehicleJourney></VehicleActivity>	Koordinaten nach WGS84, Aktualisierungsintervall ≤ 30 Sekunden.
Mindestumsteigezeit zwischen Taxi und Bus	ServiceJourneyInterchange mit MinimumTime	<ServiceJourneyInterchange><MinimumTime>PT15M</MinimumTime></ServiceJourneyInterchange>	Zeitangabe im ISO-8601-Format (z. B. PT15M = 15 Minuten).
Dynamische Bedienegebiete für Taxis	FlexibleStopPlace mit Polygon-Geometrie	<FlexibleStopPlace><Polygon><posList>51.3875 7.4714...</posList></Polygon></FlexibleStopPlace>	Polygone mit mind. 3 Koordinatenpaaren, max. 1.000 Punkte.
Mehrsprachige Haltestellenamen	Text-Elemente mit language-Attribut	<Name><Text language="de">Arnsberg Marktplatz</Text></Name>	Sprachcode nach ISO 639-1 (z. B. de, en, tr).
Festpreis für Taxifahrten	keyList mit KeyValue-Paar FESTPREIS	<keyList><KeyValue><Key>FESTPREIS</Key><Value>40.00</Value></KeyValue></keyList>	Preis in EUR, zwei Dezimalstellen, ohne Währungssymbol.
Integration des WestfalenTarifs	TypeOfTariffRef auf regionales Profil	<TypeOfTariffRef ref="VSPV:Tariff:WestfalenTariff"/>	Referenzierte Tarifprofile müssen im ResourceFrame definiert sein.
Zahlungsbedingungen für MaaS	conditions-Element in FareStructure	<conditions><Condition><Text>Gilt nur mit Registrierung</Text></Condition></conditions>	Klartext in Deutsch, max. 255 Zeichen.
Rollstuhlgerechtes Taxi	VehicleType mit keyList-Attribut	<keyList><KeyValue><Key>WHEELCHAIR_ACCESS</Key><Value>true</Value></KeyValue></keyList>	Schlüssel WHEELCHAIR_ACCESS ist verbindlich.
Kapazitätsgrenzen für Ridepooling	passengerCapacity in FlexibleServiceJourney	<FlexibleServiceJourney><passengerCapacity>6</passengerCapacity></FlexibleServiceJourney>	Wert ≥ 1, Standard: 4.
Betriebszeiten des Buchungskiosks	OpeningTimes in KioskAccess	<OpeningTime><FromTime>06:00:00</FromTime><ToTime>22:00:00</ToTime></OpeningTime>	Zeitangaben im 24h-Format, Zeitzone: Europe/Berlin.

### 3.3 Feldgrößen, Zeichensätze und Namenskonventionen

Einheitliche Feldgrößen und Zeichensätze sind essenziell, um die technische Interoperabilität, Datenqualität und Validierbarkeit im VSPV-Standard 101 sicherzustellen. Die nachfolgende Übersicht legt für alle relevanten Felder die maximal zulässigen Längen, die erlaubten Zeichensätze sowie ggf. Besonderheiten und Ausnahmen fest. Diese Vorgaben sind verbindlich für alle Datenlieferungen und werden systematisch geprüft.

#### Tabellarische Übersicht der wichtigsten Felder

Feldtyp	Maximale Länge	Zeichensatz / erlaubte Zeichen	Hinweise / Besonderheiten
ID	50	[A-Za-z0-9:_-] (ASCII), keine Leer- oder Steuerzeichen	Aufbau: Namensraum:Objekttyp:Schlüssel, eindeutig

Name	70	UTF-8, Buchstaben, Ziffern, Leerzeichen, Umlaute, Sonderzeichen wie - / ( ) .	Keine Steuerzeichen, keine Emojis
Beschreibung	255	UTF-8, wie Name, zusätzlich Satzzeichen	Zeilenumbrüche als \n erlaubt, sonst keine Steuerzeichen
Freitextfeld	255	UTF-8, wie Beschreibung	Für Hinweise, Bedingungen, Einschränkungen
Code (z. B. Liniencode)	20	[A-Za-z0-9] (ASCII), keine Sonderzeichen	Keine Leerzeichen
Telefonnummer	20	[0-9+ ]	Telefonnummern sollen – soweit verfügbar – im internationalen Format gemäß E.164 bereitgestellt werden (z. B. +49233566666), wobei Leerzeichen aus Lesbarkeitsgründen erlaubt sind.
E-Mail-Adresse	100	[A-Za-z0-9@._-]	Muss RFC-konform sein
URL	255	ASCII, keine Leerzeichen, nur gültige URL-Zeichen	Muss RFC-konform sein

## Erlaubte und verbotene Zeichen

- **Erlaubt:**

- Buchstaben (A–Z, a–z), Ziffern (0–9), Umlaute (ä, ö, ü, Ä, Ö, Ü, ß)
- Satzzeichen: Punkt (.), Komma (,), Bindestrich (-), Schrägstrich (/), Klammern (), Doppelpunkt (:)
- Leerzeichen (außer in IDs und Codes)
- Für Beschreibungen und Freitext: auch Semikolon (;), Ausrufezeichen (!), Fragezeichen (?), Apostroph (')

- **Nicht erlaubt:**

- Steuerzeichen (z. B. Tabulator, Backspace, Steuersequenzen)
- Emojis und nicht druckbare Unicode-Zeichen
- HTML- oder XML-Steuerzeichen (z. B. <, >, &, sofern nicht maskiert)
  - XML-relevante Sonderzeichen wie <, > und & dürfen ausschließlich als Maskierungen (&lt;, &gt;, &amp;) verwendet werden. Rohes Einfügen dieser Zeichen führt zu Parsing-Fehlern und wird bei der Validierung abgewiesen.

- HTML-Tags dürfen nicht verwendet werden. Falls in Quellsystemen vorhanden, sind sie vor der Konvertierung vollständig zu entfernen oder als Klartext (escaped) zu speichern.
- Zeilenumbrüche nur in Beschreibungen/Freitext als explizites \n

## Beispiele für korrekte und fehlerhafte Einträge

### Korrekt:

```
<Name>
  <Text language="de">Arnsberg Marktplatz</Text>
  <Text language="en">Arnsberg Market Square</Text>
</Name>
<Description>Historischer Marktplatz im Zentrum von
Arnsberg.</Description>
<StopPlace id="VSPV:StopPlace:NRW-1001" version="1.0"/>
```

### Fehlerhaft:

```
<Name>
  <Text language="de">Arnsberg 🏰 Marktplatz</Text> <!-- Emoji nicht
erlaubt -->
</Name>
<StopPlace id="VSPV:Stop Place:NRW-1001" version="1.0"/> <!--
Leerzeichen in ID -->
<Description>Marktplatz<tab>Zentrum</Description> <!-- Tabulator nicht
erlaubt -->
```

## Validierungsregeln

- Bei Überschreitung der maximalen Feldlänge wird die Datenlieferung abgelehnt.
- Nicht erlaubte Zeichen führen zu einer Fehlermeldung mit Angabe des betroffenen Feldes.
- IDs müssen eindeutig und stabil sein; Dubletten werden abgewiesen.
- Für mehrsprachige Felder ist der Sprachcode nach ISO 639-1 verpflichtend.

## Namenskonventionen für IDs und Bezeichnungen

Die konsequente Anwendung einheitlicher Namenskonventionen für Identifikatoren (IDs) und Bezeichnungen ist eine zentrale Voraussetzung für die technische Interoperabilität, Nachvollziehbarkeit und Wartbarkeit aller Daten im VSPV-Standard 101. Die nachfolgenden Regeln sind verbindlich für alle Datenlieferungen und gelten sowohl für interne als auch für externe Schnittstellen.

### Aufbau und Struktur von IDs

- Jede ID besteht aus drei Hauptbestandteilen, die durch Doppelpunkte getrennt werden:
  - Namensraum (Projekt/Organisation, z. B. VSPV, NRW, DIVA)

- Objekttyp (z. B. StopPlace, ServiceJourney, VehicleType)
- Schlüssel/Laufnummer (z. B. NRW-1001, Bus-432-2025-01)
- Beispiel:
  - VSPV:StopPlace:NRW-1001
  - VSPV:ServiceJourney:Bus-432-2025-01
- Regeln für IDs:
  - IDs dürfen ausschließlich aus Groß- und Kleinbuchstaben (A–Z, a–z), Ziffern (0–9), Bindestrichen (-), Unterstrichen (\_), und Doppelpunkten (:) bestehen.
  - Keine Leerzeichen, keine Sonderzeichen, keine Umlaute, keine Steuerzeichen.
  - IDs sind systemweit eindeutig und persistent zu vergeben – sie dürfen nachträglich nicht geändert oder wiederverwendet werden.
  - Jede Änderung am Objekt, die zu einer neuen Version führt, wird über das Attribut version abgebildet, nicht über eine neue ID.
- Namensraum-Konventionen:
  - Der Namensraum identifiziert eindeutig die Organisation oder das Projekt, das die ID vergibt.
  - Bei föderierten Systemen oder Datenmigrationen ist eine zentrale Registry für Namensräume zu führen.

### **Konventionen für Bezeichnungen (Namen)**

- Bezeichnungen (Namen) sind sprechend, eindeutig und möglichst kurz zu halten (max. 70 Zeichen).
- Namen sind UTF-8-codiert und dürfen Umlaute, Leerzeichen und gängige Satzzeichen enthalten.
- Keine Emojis, keine Steuerzeichen, keine HTML-Tags.
- Namen sind in der jeweils relevanten Sprache zu pflegen. Für mehrsprachige Systeme sind alle relevanten Sprachversionen als <Text language="...">-Elemente zu hinterlegen.

Beispiel für mehrsprachigen Namen:

```
<Name>
  <Text language="de">Arnsberg Marktplatz</Text>
  <Text language="en">Arnsberg Market Square</Text>
</Name>
```

- Sonderregeln für bestimmte Objekttypen:

- **Liniennamen:**  
Format: Bus 432, RE57, Taxi Wetter. Keine Sonderzeichen außer Bindestrich, keine Leerzeichen am Anfang/Ende.
- **Haltestellenamen:**  
Eindeutig im regionalen Kontext, ggf. Zusatz (z. B. „Bahnhof“, „Marktplatz“).
- **Fahrzeugtypen:**  
Sprechende Bezeichnung, ggf. mit Zusatzmerkmalen (z. B. „Rollstuhltaxi E“).

### Beispiele für korrekte und fehlerhafte Benennungen

Korrekt:

```
VSPV:StopPlace:NRW-1001
VSPV:ServiceJourney:Bus-432-2025-01
<Text language="de">Arnsberg Marktplatz</Text>
```

Fehlerhaft:

```
VSPV:Stop Place:NRW 1001 (Leerzeichen nicht erlaubt)
VSPV:ServiceJourney:Bus#432@2025 (Sonderzeichen nicht erlaubt)
<Text language="de">Arnsberg 🇩🇪 Marktplatz</Text> (Emoji nicht erlaubt)
```

### Validierungsregeln

- IDs werden bei jeder Datenlieferung auf Einhaltung der Namenskonventionen und Eindeutigkeit geprüft.
- Namen, die die erlaubten Zeichen oder Längen überschreiten, werden abgelehnt und müssen korrigiert werden.
- Mehrsprachige Namen müssen mit gültigem ISO-639-1-Sprachcode versehen sein.

### Mehrsprachigkeit und Textfelder

Die Unterstützung von Mehrsprachigkeit ist im VSPV-Standard 101 verbindlich vorgesehen, um die Nutzbarkeit der Daten für unterschiedliche Zielgruppen, internationale Fahrgäste und digitale Plattformen zu gewährleisten. Textfelder wie Namen, Beschreibungen und Hinweise müssen so gepflegt werden, dass sie in mehreren Sprachen bereitgestellt und eindeutig zugeordnet werden können.

### Grundprinzipien der Mehrsprachigkeit

- Für alle relevanten Textfelder (zum Beispiel Name, Beschreibung, Hinweise) ist die Pflege in mehreren Sprachversionen möglich und empfohlen.
- Jede Sprachversion wird als separates Text-Element mit dem Attribut language gemäß ISO 639-1 hinterlegt, zum Beispiel de für Deutsch, en für Englisch, tr für Türkisch.

- Die Standardsprache ist Deutsch (de). Für internationale Plattformen und Schnittstellen ist mindestens eine englische (en) Übersetzung bereitzustellen.
- Fehlt eine Übersetzung, wird die Standardsprache ausgeliefert.
- Für jedes Objekt ist mindestens eine Sprachversion des Namens in der Standardsprache Deutsch (language="de") verpflichtend. Ohne diese Angabe wird die Datenlieferung abgelehnt.

### Beispiel für mehrsprachige Textfelder

```
<Name>
  <Text language="de">Arnsberg Marktplatz</Text>
  <Text language="en">Arnsberg Market Square</Text>
  <Text language="tr">Arnsberg Pazar Meydanı</Text>
</Name>
<Description>
  <Text language="de">Historischer Marktplatz im Zentrum von
Arnsberg.</Text>
  <Text language="en">Historic market square in the centre of
Arnsberg.</Text>
</Description>
```

### Konventionen für Textfelder

- Die maximale Länge pro Textfeld beträgt 70 Zeichen für Namen und 255 Zeichen für Beschreibungen und Hinweise.
- Textfelder sind UTF-8-codiert und dürfen Umlaute, Satzzeichen und Leerzeichen enthalten.
- Für alle UTF-8-Felder ist die Unicode-Normalform NFC (Normalization Form C) verbindlich anzuwenden, um konsistente Darstellung, Speicherung und Vergleichbarkeit sicherzustellen.
- Emojis, Steuerzeichen und HTML-Tags sind nicht zulässig.
- Für Hinweise und Bedingungen (zum Beispiel Tarifbedingungen) gilt ebenfalls die Mehrsprachigkeitsregel. Hier ist auf eine klare, verständliche Formulierung zu achten.

### Validierung und Fehlerbehandlung

- Jede Sprachversion eines Textfeldes muss mit einem gültigen Sprachcode nach ISO 639-1 versehen sein.
- Fehlen Pflichtfelder in der Standardsprache, wird die Datenlieferung abgelehnt.
- Übersetzungen, die die maximale Feldlänge überschreiten oder unerlaubte Zeichen enthalten, werden abgewiesen.
- Bei der Integration in Plattformen ohne Mehrsprachigkeit werden nur die Felder der Standardsprache ausgegeben.

## **Best Practices**

- Die Pflege der Übersetzungen sollte zentral dokumentiert und regelmäßig aktualisiert werden, insbesondere bei Änderungen an Namen oder Beschreibungen.
- Bei neuen Objekten ist die englische Übersetzung zeitnah mitzuliefern, um internationale Nutzbarkeit sicherzustellen.
- Für häufig verwendete Begriffe (wie „Bahnhof“, „Marktplatz“, „Hauptstraße“) empfiehlt sich eine einheitliche Übersetzung in allen Datensätzen.

## **Fehlerbehandlung und Validierung**

Die Sicherstellung der Datenqualität im VSPV-Standard 101 erfordert verbindliche Regeln für die Fehlerbehandlung und Validierung aller Felder. Ziel ist es, fehlerhafte, unvollständige oder nicht konforme Daten frühzeitig zu erkennen, abzuweisen oder gezielt zu korrigieren, bevor sie in produktive Systeme oder Plattformen übernommen werden.

### **Grundprinzipien der Fehlerbehandlung**

- Jede Datenlieferung wird automatisiert auf Einhaltung der Feldgrößen, Zeichensätze, Namenskonventionen und Mehrsprachigkeitsregeln geprüft.
- Fehlerhafte Einträge werden mit einer klaren Fehlermeldung versehen, die das betroffene Feld, die Art des Fehlers und eine Korrekturhilfe angibt.
- Korrekturen sind vor der erneuten Datenübermittlung durchzuführen. Die Verantwortung liegt beim Datenlieferanten.

### **Typische Fehlerquellen und ihre Behandlung**

- Überschreitung der maximalen Feldlänge (z. B. Name mit mehr als 70 Zeichen): Die Datenlieferung wird abgelehnt, bis das Feld gekürzt wurde.
- Verwendung unerlaubter Zeichen (z. B. Emojis, Steuerzeichen, HTML-Tags): Das betroffene Feld wird abgelehnt und muss bereinigt werden.
- Fehlender oder ungültiger Sprachcode bei mehrsprachigen Feldern: Die Datenlieferung wird abgelehnt, bis ein gültiger ISO 639-1-Code angegeben ist.
- Nicht eindeutige oder nicht konforme IDs: Die betroffenen Objekte werden abgelehnt, bis die IDs korrigiert und in der Registry abgestimmt sind.
- Pflichtfelder (z. B. Name in der Standardsprache) fehlen: Die Datenlieferung wird abgelehnt, bis das Pflichtfeld ergänzt wurde.

### **Validierungsmechanismen**

- Die technische Validierung erfolgt automatisiert gegen das definierte XSD-Schema (z. B. NeTeX/VDV 462). XML-Dateien sind ausschließlich mit UTF-8-

Kodierung auszuliefern. Die XML-Deklaration am Dateianfang muss explizit das Encoding angeben: `<?xml version="1.0" encoding="UTF-8"?>`.

- Zusätzlich erfolgen semantische Prüfungen, etwa auf Eindeutigkeit von IDs, Konsistenz von Referenzen, Plausibilität von Zeitangaben und Einhaltung von Namenskonventionen.
- Bei Fehlern wird ein Validierungsprotokoll erstellt, das alle Beanstandungen auflistet und die Nachbesserung unterstützt.

### Beispiele für Fehlermeldungen

- „Feld `<Name>`: Überschreitet maximale Länge von 70 Zeichen.“
- „ID `VSPV:Stop Place:NRW 1001` enthält unerlaubte Leerzeichen.“
- „Text `language='de'` fehlt im Name-Element.“
- „Beschreibung enthält nicht erlaubtes Zeichen: 🚧.“

### Best Practices

- Vor jeder Datenlieferung empfiehlt sich eine lokale Vorvalidierung mit den bereitgestellten Prüftools.
- Fehlerprotokolle sollten dokumentiert und für die kontinuierliche Qualitätsverbesserung ausgewertet werden.
- Bei wiederkehrenden Fehlerbildern empfiehlt sich eine Schulung der Datenpflegenden und die Anpassung der Erfassungswerkzeuge.

### Mappingtabelle Kapitel 3.3

Fachliche Anforderung	Technische Datenstruktur	Beispiel-XML	Konventionen
Eindeutige, stabile ID	Attribut „id“ in jedem Objekt	<code>&lt;StopPlace id="VSPV:StopPlace:NRW-1001" version="1.0"/&gt;</code>	Max. 50 Zeichen, keine Leerzeichen/Sonderzeichen, Aufbau nach Schema
Sprechender Name in mehreren Sprachen	Name/Text mit language-Attribut	<code>&lt;Name&gt;&lt;Text language="de"&gt;Arnsberg Marktplatz&lt;/Text&gt;&lt;Text language="en"&gt;Arnsberg Market Square&lt;/Text&gt;&lt;/Name&gt;</code>	Max. 70 Zeichen, UTF-8, ISO 639-1 Sprachcode
Beschreibung mit Zusatzinformationen	Description/Text	<code>&lt;Description&gt;&lt;Text language="de"&gt;Historischer Marktplatz im Zentrum von Arnsberg.&lt;/Text&gt;&lt;/Description&gt;</code>	Max. 255 Zeichen, UTF-8, keine Emojis oder Steuerzeichen
Liniencode	Code-Attribut oder Name	<code>&lt;Line id="VSPV:Line:Bus-432"&gt;&lt;Name&gt;&lt;Text language="de"&gt;Bus 432&lt;/Text&gt;&lt;/Name&gt;&lt;/Line&gt;</code>	Max. 20 Zeichen, nur Buchstaben und Ziffern, keine Leerzeichen
Telefonnummer	TelephoneAccess/PhoneNumber	<code>&lt;PhoneNumber&gt;0233566666&lt;/PhoneNumber&gt;</code>	Max. 20 Zeichen, nur Ziffern, Pluszeichen und Leerzeichen
Eindeutigkeit und Versionierung	id + version-Attribut	<code>&lt;StopPlace id="VSPV:StopPlace:NRW-1001" version="1.0"/&gt;</code>	Jede Änderung als neue Version, ID bleibt stabil; Das Attribut version ist in jedem Objekt

Fachliche Anforderung	Technische Datenstruktur	Beispiel-XML	Konventionen
			verpflichtend zu setzen und verwendet das Schema X.Y (z. B. version="1.0"). Ein Fehlen oder eine abweichende Formatierung führt zur Ablehnung.
Pflichtfeld Name in Standardsprache	Name/Text language="de"	<Name><Text language="de">Arnsberg Marktplatz</Text></Name>	Muss immer vorhanden sein
Mehrsprachigkeit bei Bedarf	Name/Text mit weiteren Sprachen	<Text language="en">Arnsberg Market Square</Text>	Empfohlen für internationale Nutzung
Feldlängen und Zeichensatz	Attributlänge und UTF-8	<Description><Text language="de">...</Text></Description>	Siehe Vorgaben zu max. Länge und erlaubten Zeichen
Fehlerbehandlung und Validierung	Validierungsprotokoll	—	Lieferung wird bei Verstoß abgelehnt, Fehlerprotokoll wird erstellt

### 3.4 Buchungs- und Zahlungslogik im Datenmodell

In diesem Abschnitt werden die Buchungsvorgänge und Zahlungsarten im Datenmodell des VSPV-Standards 101 formal festgelegt. Ziel ist es, klar und verbindlich zu definieren, wie Buchungsprozesse (von der Anfrage bis zur Bestätigung) sowie Bezahlverfahren (von Barzahlung bis Drittzahler) in NeTeX/VDV-462-konformer Weise abzubilden sind. Damit wird sichergestellt, dass die Integration von Gelegenheitsverkehren in digitale Auskunftssysteme, Buchungs- und Vertriebssysteme den Anforderungen der Mobilitätsdatenverordnung (insb. Angabe von Buchungs- und Bezahlmöglichkeiten) entspricht und alle Systeme einheitlich auf diese Informationen zugreifen können. Da Buchungslogiken und Zahlungsabwicklungen in klassischen ÖV-Datenstandards bisher nur unzureichend berücksichtigt wurden, werden an geeigneten Stellen Erweiterungen des Datenmodells vorgenommen, die mit den VDV-462-Profilen kompatibel sind und die neuen Anforderungen adressieren.

#### Buchungskanäle, Buchungswege und Benutzerrollen

Gelegenheitsverkehre (On-Demand-Verkehre, flexibler ÖPNV und Taxi-Integration) können über unterschiedliche Kanäle gebucht werden. Der VSPV-Standard 101 sieht vor, dass alle Buchungswege eines Angebots explizit im Datenmodell hinterlegt werden. Typische Buchungskanäle sind unter anderem:

- Mobile App / Web – Selbstbedienungs-Buchung durch den Fahrgast über eine mobile Applikation oder Web-Oberfläche (im NeTeX-Kontext meist als *online*-Buchung klassifiziert).
- Telefon (Callcenter) – Buchung durch Anruf bei einer Dispositionszentrale oder direkt beim Betreiber (*callOffice* oder *callDriver* in bestehenden Profilen). Hier ist die Benutzerrolle typischerweise ein Vermittler (Mitarbeiter im Callcenter), der die Buchung im System für den Kunden durchführt.

- Kiosk/Terminal vor Ort – Buchung über ein physisches Terminal oder einen Automaten am Haltepunkt (vergleichbar mit *phoneAtStop*). Diese Variante erfordert die Modellierung der Terminal-Verfügbarkeit an bestimmten Standorten (z. B. als Ausstattung eines StopPlace) sowie ggf. von Öffnungszeiten und Zugriffsrechten. Im Datenmodell werden die Buchungswege *Kiosk* und *Telefon* beispielsweise mit zugehörigen Details wie Kiosk-Betriebszeiten oder Telefonnummer des lokalen Betreibers hinterlegt.

Jeder dieser Buchungskanäle wird im NeTEx-Datenmodell über das Konzept der Buchungsmodalitäten abgebildet. Hierzu wird pro flexibler Linie bzw. pro Angebot ein Booking Arrangement definiert, das folgende Informationen umfasst:

- Zulässige Buchungswege (BookingMethods): Liste der unterstützten Kanäle (z. B. phone, online, kiosk etc.). Mindestens ein Buchungsweg muss angegeben werden, sofern das Angebot reservierungspflichtig ist. Die im NeTEx-Standard vorgesehenen Werte (z. B. *callDriver*, *callOffice*, *online*, *phoneAtStop*) werden verwendet und bei Bedarf um *kiosk* oder ähnliche Einträge erweitert. Fehlt die Angabe eines Buchungswegs für einen reservierungspflichtigen Dienst, ist dies ein Validierungsfehler (die Datenlieferung wird abgelehnt, bis das Feld ergänzt wurde).
- Buchungskontakte: Für jeden Buchungskanal sind die erforderlichen Kontaktinformationen zu modellieren. Beispielsweise eine Telefonnummer für telefonische Buchung (<PhoneNumber> gemäß NeTEx, max. 20 Zeichen), eine URL für App/Web-Buchungen oder die Angabe der Standortkennung eines Kiosks. Diese Informationen werden typischerweise als Unter-elemente des Booking Arrangement erfasst (z. B. <ContactPerson>, <Phone> oder <Url>). Fehlende Kontaktdaten trotz angegebenem Buchungsweg führen zur Zurückweisung der Datenlieferung.
- Benutzerrolle: Das Datenmodell unterscheidet implizit, wer den Buchungsvorgang initiiert. Eine Online-Buchung wird in der Regel vom Endnutzer selbst ausgelöst, während eine telefonische Buchung oder Kiosk-Buchung häufig durch Vermittler (Callcenter-Mitarbeiter, Kiosk-Personal) erfolgt. Diese Unterscheidung kann z. B. in Form unterschiedlicher TypeOfBookingService oder über Metadaten im Booking Arrangement ausgedrückt werden. Zwar wird die Benutzerrolle nicht als eigenes Datenfeld in NeTEx geführt, doch kann über die Art des BookingMethod und ggf. zusätzliche Attribute abgeleitet werden, ob es sich um Selbstbedienung oder Vermittlung handelt.

Zusätzlich können spezielle Parameter der Buchungslogik als Erweiterungen erfasst werden. So werden etwa Vorlaufzeiten (zeitliche Fristen für Vorabbuchungen), Buchungsfenster (z. B. Buchung nur zu bestimmten Tageszeiten möglich) oder Kapazitätsgrenzen als optionale Attribute modelliert. Der VSPV-Standard empfiehlt hierfür die Nutzung von KeyValue-Paaren innerhalb einer keyList oder von strukturierten Feldern im <Extensions>-Bereich, sofern keine Standardfelder existieren. Zum Beispiel kann eine minimale Vorlaufzeit von 30 Minuten vor Fahrtbeginn als KeyValue {"LATEST\_BOOKING\_TIME": "PT30M"} abgebildet werden. Im Ergebnis ist für jeden

flexiblen Dienst eindeutig in den Daten hinterlegt, wie und bis wann Fahrgäste buchen können und über welche Kanäle die Kommunikation erfolgt.

**Praxisbeispiel: Kioskbuchung einer Krankenfahrt in der Arztpraxis**

Ein typisches zukünftiges Anwendungsbeispiel ist die Buchung einer Krankenfahrt durch eine Arztpraxis in Hamm. Die Praxis verfügt über ein autorisiertes Kiosksystem, das an das zentrale Buchungs- und Vertriebssystem angebunden ist – dieselbe Plattform, über die auch reguläre Fahrkarten für Bus und Bahn erworben werden. Die Mitarbeiterin der Praxis gibt dort Start- und Zielort, Abfahrtszeit, Fahrgasteigenschaften (z. B. eingeschränkte Mobilität) sowie abrechnungsrelevante Informationen wie die Versichertennummer, das Genehmigungskennzeichen der Krankenkasse und ggf. besondere Anforderungen an das Fahrzeug (z. B. Rollstuhlrampe) ein. Alternativ können die Daten auch aus der Praxensoftware übernommen werden.

Der VSPV-Standard bildet diesen Vorgang durch ein BookingProcess-Objekt mit der Methode kiosk, ergänzt durch ein KioskAccess-Element mit Standortbezug und Öffnungszeiten. Die zugehörige FlexibleServiceJourney referenziert Start- und Zielpunkte, Buchungsprozess und – falls nötig – eine Fahrzeugkategorie (VehicleTypeRef). Abrechnungsrelevante Daten werden in einem BillingRecord-Objekt oder äquivalenten Attributen modelliert und können automatisiert an die Kostenträger übermittelt werden.

Die Buchung erfolgt über ein BookingProcess-Objekt, das die zulässigen Buchungsmethoden (z. B. App, Kiosk, Telefon), etwaige Vorlaufzeiten und die erforderlichen Authentifizierungsverfahren festlegt. Der VSPV-Standard erlaubt die Differenzierung nach Zugangskanälen, etwa mobileApp, kiosk, telephone, web oder thirdPartySystem.

Beispielhafte Modellierung:

```
<StopPlace id="VSPV:StopPlace:Klinikum-Hamm" version="1.0">
  <Name>
    <Text language="de">Klinikum Hamm</Text>
  </Name>
  <Centroid>
    <Location>
      <Longitude>7.8185</Longitude>
      <Latitude>51.6803</Latitude>
    </Location>
  </Centroid>
</StopPlace>

<StopPlace id="VSPV:StopPlace:Wohnort-Fahrgast" version="1.0">
  <Name>
    <Text language="de">Wohnort des Fahrgasts</Text>
  </Name>
</StopPlace>

<BookingProcess id="VSPV:BookingProcess:Klinik-Hamm" version="1.0">
```

```

<Name>
  <Text language="de">Kioskbuchung Klinik Hamm</Text>
</Name>
<bookingMethods>kiosk</bookingMethods>
<bookingAccess>
  <KioskAccess>
    <StopPlaceRef ref="VSPV:StopPlace:Klinikum-Hamm"/>
    <OpeningTimes>
      <OpeningTime>
        <FromTime>07:00:00</FromTime>
        <ToTime>18:00:00</ToTime>
      </OpeningTime>
    </OpeningTimes>
  </KioskAccess>
</bookingAccess>
<latestBookingTime>PT30M</latestBookingTime>
</BookingProcess>

<FlexibleServiceJourney id="VSPV:ServiceJourney:Krankentransport-
Hamm" version="1.0">
  <Name>
    <Text language="de">Krankentransport Klinik Hamm → Wohnort</Text>
  </Name>
  <TransportMode>taxi</TransportMode>
  <JourneyPattern>
    <StopPointInJourneyPattern>
      <Order>1</Order>
      <StopPointRef ref="VSPV:StopPlace:Klinikum-Hamm"/>
    </StopPointInJourneyPattern>
    <StopPointInJourneyPattern>
      <Order>2</Order>
      <StopPointRef ref="VSPV:StopPlace:Wohnort-Fahrgast"/>
    </StopPointInJourneyPattern>
  </JourneyPattern>
  <BookingProcessRef ref="VSPV:BookingProcess:Klinik-Hamm"/>
  <keyList>
    <KeyValue>
      <Key>FESTPREIS</Key>
      <Value>22.50</Value>
    </KeyValue>
    <KeyValue>
      <Key>KOSTENTRAEGER</Key>
      <Value>GKV:IKK-123456</Value>
    </KeyValue>
    <KeyValue>

```

```

<Key>VERSICHERTENNUMMER</Key>
<Value>X123456789</Value>
</KeyValue>
<KeyValue>
<Key>GENEHMIGUNGSNUMMER</Key>
<Value>GZ-88-2025-04</Value>
</KeyValue>
</keyList>
</FlexibleServiceJourney>

```

## Modellierung der Zahlungsarten und Abrechnung

Der VSPV-Standard 101 schreibt vor, dass alle unterstützten Zahlungsarten eines Angebots im Datenmodell ausgewiesen werden. Folgende Zahlungsarten werden dabei unterschieden und müssen jeweils gemäß Datenformat modelliert werden:

- **Barzahlung:** Zahlung des Fahrpreises in bar an den Fahrer oder am Kiosk. Diese Option wird im Datenmodell als cash (Bargeld) hinterlegt, typischerweise in einem PaymentMethod-Feld. Ist Barzahlung möglich, muss kein elektronischer Zahlungsnachweis in der Buchung übertragen werden; allerdings muss der Fahrgast im Buchungssystem darauf hingewiesen werden, dass während der Fahrt zu zahlen ist. In NeTEx kann dies z. B. durch eine Verkaufskonfiguration (*SalesOfferPackage*) ausgedrückt werden, die den Verkaufskanal „on board“ mit Zahlungsart Bar verknüpft. Barzahlung sollte nur für Angebote vorgesehen werden, bei denen das Personal vor Ort die Zahlung entgegennehmen kann (z. B. Taxi, Bürgerbus).
- **E-Payment (Kreditkarte/Lastschrift/Wallet):** Elektronische Vorauszahlung im Zuge der Buchung (z. B. per Kreditkarte, SEPA-Lastschrift, PayPal o. ä.). Im Datenmodell wird dies als electronicPayment oder spezifischer (z. B. card, directDebit) codiert. Die Angabe erfolgt analog in einem PaymentMethod-Feld oder über die Verkaufsart. Ist E-Payment vorgesehen, muss das Buchungssystem die entsprechenden Zahlungsinformationen sicher erfassen; für das Datenaustauschmodell genügt die Kennzeichnung, dass elektronisch gezahlt wird. Die Abwicklung selbst (Payment-Gateway, Autorisierung) liegt außerhalb von NeTEx, wird aber durch die Buchungsbestätigung (vgl. TRIAS-BookingConfirmation) mit einem erfolgreichen Status abgebildet.
- **Rechnung:** Bezahlung auf Rechnung nach Leistungserbringung. Diese Zahlungsart (invoice) erfordert im Datenmodell besondere Kennzeichnungen, da hier abrechnungsrelevante Zusatzinformationen anfallen. Insbesondere ist eine Kostenträgerkennung zu übermitteln, die den Rechnungsempfänger oder die Kostenstelle eindeutig identifiziert. Beispielsweise kann für interne Abrechnungen ein Feld costCarrierID im Extensions-Bereich des Buchungsdatensatzes genutzt werden. Wird Zahlungsart *Rechnung* unterstützt, ist bei jeder Buchung anzugeben, welcher Kostenträger die Fahrt übernimmt (z. B. Kundennummer, Vertrags-ID oder Kennung der zahlenden Organisation). Fehlt diese Angabe, darf die Buchung nicht bestätigt werden. Die Datenstruktur muss also vorsehen, dass

paymentMethod="invoice" nur in Kombination mit einem gültigen Kostenträger-Schlüssel verwendet wird (Validierungsregel).

- **Gutscheine:** Bezahlung mittels Gutscheincodes oder Wertgutscheinen. Im Datenmodell wird dies als voucher oder coupon Zahlungsart angegeben. Ist diese Option vorgesehen, muss das Buchungssystem einen Gutscheincode beim Fahrgast abfragen und den Wert verifizieren. Im NeTEx-FareFrame kann ein Gutschein als spezielles Tarifprodukt modelliert werden – etwa als Fahrausweis mit 100% Rabatt – oder die Unterstützung von Gutscheinen wird durch ein einfaches boolesches Merkmal in den Buchungsregeln signalisiert. In jedem Fall sollte in der Dokumentation der Schnittstelle erläutert sein, wie Gutscheine technisch eingebunden sind. Oft erfolgt die Einlösung im Buchungssystem selbst, sodass im Datenaustausch lediglich vermerkt wird, dass ein Gutschein eingesetzt wurde (z. B. als PaymentMethod-Code). Der konkrete Gutscheincode wird aus Sicherheitsgründen nicht im öffentlichen Datenaustausch verbreitet, kann aber in der BookingConfirmation für Abrechnungszwecke in einem verschlüsselten Feld erscheinen.
- **Drittzahler:** Kostenübernahme durch Dritte, z. B. Krankenkassen, Arbeitgeber oder andere Institutionen. Diese Zahlungsart (thirdParty) ähnelt der Rechnungszahlung, erfordert jedoch zusätzlich die Identifikation des Drittzahlers im Datenmodell. Hierzu wird – analog zur Rechnung – die Kostenträgerkennung verwendet, die im Fall von Krankentransporten z. B. die Krankenkasse und die Ordnungsnummer umfassen kann. Im Buchungsdatensatz muss für *Drittzahler*-Buchungen zwingend angegeben werden, welcher externe Träger zahlt (z. B. mittels eines Codes oder Verweises auf einen hinterlegten Kostenträger-Stammdatensatz). Das Datenmodell kann hierfür einen Verweis auf eine Organisation (NeTEx-Objekt Organisation im ResourceFrame) vorsehen, welche als *Payer* fungiert. Ist kein gültiger Drittzahler angegeben, wird die Buchung zurückgewiesen.

Zur Abbildung der Zahlungsarten in NeTEx bietet Teil 3 (Tarifdaten) des Standards entsprechende Mechanismen. Insbesondere können über das Element PaymentMethod innerhalb eines FareStructure bzw. SalesPackage die erlaubten Zahlungsformen definiert werden. Sofern das Standard-XSD nicht alle o. g. Arten direkt als Enumeration anbietet (z. B. fehlen dort u. U. spezielle Werte für Gutschein oder Drittzahler), werden im VSPV-Standard Erweiterungen genutzt. Dies geschieht bevorzugt über kodierte Werte in einer keyList des FareFrame oder über profilkonforme Erweiterungen. Beispielsweise könnte eine keyList in den FareStrukturen folgendes enthalten:

```
<keyList>
  <KeyValue>
    <Key>ACCEPTED_PAYMENTS</Key>
    <Value>cash, electronic, invoice, voucher, thirdParty</Value>
  </KeyValue>
</keyList>
```

Diese vereinfachte Darstellung listet alle zugelassenen Zahlungsarten für einen bestimmten Tarif oder ein Angebot auf. Alternativ könnte pro Zahlungsart ein eigenes Key-Value-Paar mit dem Wert true/false geführt werden. Wichtig ist, dass mindestens eine Zahlungsart pro kostenpflichtigem Angebot ausgewiesen wird. Enthält ein Tarif oder Angebot keine Angabe zur Zahlungsart, wird angenommen, dass keine Buchung erforderlich bzw. die Fahrt unentgeltlich ist – dies ist im Zweifel zu vermeiden, indem explizit klargestellt wird, wie der Fahrschein erworben wird.

**Abrechnungsrelevante Datenfelder:** Wie oben beschrieben, müssen für Rechnung und Drittzahler besondere Felder vorgesehen werden. Der VSPV-Standard definiert dafür zwei zentrale Datenfelder, die im Buchungsprozess und bei der Abrechnung verwendet werden:

- **Kostenträgerkennung:** Ein alphanumerischer Code, der den zahlenden Dritten oder die Kostenstelle eindeutig identifiziert (z. B. Krankenkassennummer + Versichertennummer, Arbeitgeber-ID, Projektnummer etc.). Dieses Feld wird im Buchungsdatensatz benötigt, wenn paymentMethod = invoice oder thirdParty gewählt wurde. Im Datenmodell kann die Kostenträgerkennung als optionales Attribut an der Buchung (im BookingFrame/BookingRecord) geführt werden, das per Validierungsregel verpflichtend wird, sobald die Zahlungsart eine solche Angabe erfordert. Bei der Übermittlung an externe Abrechnungssysteme (z. B. Abrechnung über die Krankenkasse) ist dieses Feld essenziell, um die Leistung zuzuordnen.
- **Leistungsnachweis / Beförderungsbeleg:** Hierbei handelt es sich um einen Nachweis, dass die Fahrt erbracht wurde, der insbesondere bei nicht-qualifizierten Krankentransporten gemäß PBefG erforderlich ist. Im Datenmodell kann hierfür ein Feld wie ServiceDeliveryRef oder TripProofID vorgesehen werden, das z. B. auf einen vom Fahrer unterzeichneten Beförderungsbeleg referenziert. Da dieser Nachweis oft in Papierform oder als digitales Dokument vorliegt, wird er nicht als komplexes Objekt im NeTEx abgebildet, aber die Möglichkeit zur Referenzierung (etwa durch eine Belegnummer) sollte gegeben sein. In einer BookingConfirmation könnte ein solches Feld die Nummer des generierten Leistungsnachweis-Dokuments enthalten. Für die Planung und Buchung im Voraus genügt es, festzuhalten, dass ein Leistungsnachweis erforderlich sein wird (z. B. indem im Angebotstext vermerkt ist: „Fahrt erfordert Vorlage eines Berechtigungsscheins“). Technisch ließe sich eine Binärdatei als Attachment in einer Erweiterung übermitteln, jedoch wird dies im VSPV-Standard aus Datenschutz- und Praktikabilitätsgründen nicht vorgesehen – stattdessen erfolgt die Abrechnung solcher Sonderfälle außerhalb des eigentlichen NeTEx-Datenaustauschs, auf Basis des referenzierten Nachweises.

### Typische und atypische Buchungsszenarien

Das Datenmodell muss eine Reihe von Buchungsszenarien abdecken – von der einfachen Einzelbuchung eines Fahrgasts bis zu komplexen Sammelbestellungen durch Dritte. In diesem Abschnitt werden einige wichtige Szenarien und deren Abbildung erläutert:

- **Standard-Einzelbuchung eines Fahrgasts:** Dies ist der Grundfall, in dem ein Fahrgast für sich selbst eine Fahrt bucht (per App, Telefon oder Kiosk). Das Modell erfasst dabei die erforderlichen Daten wie Start/Ziel, Uhrzeit, Personenzahl (standardmäßig 1) und gewählte Zahlungsart. Besonderheiten ergeben sich hier kaum, außer der generellen Pflicht zur Angabe aller notwendigen Felder (Buchungsweg, Kontakt, Zahlungsart). Die Bestätigung (BookingConfirmation) referenziert dann die Buchung mit einer eindeutigen ID und enthält ggf. Tarifinformationen (Preis). Dieses Szenario wird vom Datenmodell ohne Erweiterungen durch die Standardstrukturen (FlexibleLine, ServiceJourney, FareProduct) abgebildet.
- **Vorabbuchung (Langfristige Reservierung):** Bei einigen Angeboten können Fahrgäste Fahrten weit im Voraus buchen (z. B. Wochen vorher für eine Anruf-Sammeltaxi-Fahrt zu einem Arzttermin). Hierfür müssen Buchungsfristen definiert werden. Im Datenmodell wird eine maximale Vorlaufzeit entweder als Attribut (z. B. <EarliestBookingTime> oder <BookingHorizon>) oder als Key-Value (MAX\_ADVANCE\_BOOKING) modelliert. Ist keine explizite Begrenzung angegeben, wird davon ausgegangen, dass eine Tiefenvorabbuchung nicht möglich ist bzw. durch operative Maßnahmen begrenzt wird. Im Beispiel könnte eine flexible Linie erlauben: "Buchung bis zu 30 Tage im Voraus". Diese Regel würde als BookingArrangement.EarliestBookingTime = PT720H (30\*24h) hinterlegt oder in der Dokumentation des Dienstes beschrieben. Wichtig ist, dass auch das Gegenteil modelliert werden kann – also Kurzfristbuchungen: Einige On-Demand-Dienste erlauben Buchungen in Echtzeit (sofortige Fahrtenanforderung ohne Vorlauf). In dem Fall wird LatestBookingTime sehr gering (z. B. PT0M oder PT5M) gesetzt und im Angebotstext klargestellt, dass spontane Buchungen möglich sind. Das Datenmodell nach VDV 462 ermöglicht solche Angaben bereits im Standardumfang (FlexibleServiceProperties für Vorlaufzeiten), sodass hier keine proprietären Erweiterungen nötig sind, lediglich die korrekte Parametrierung.
- **Sammelbestellungen durch Dritte (z. B. Arztpraxen):** Ein häufiges Szenario im Krankenbeförderungssektor ist, dass z. B. eine Arztpraxis oder ein Krankenhaus für mehrere Patienten Fahrten bucht (Sammelbestellung). Das bedeutet, ein Akteur (Drittbesteller) reserviert ggf. regelmäßig eine Reihe von Fahrten für unterschiedliche Fahrgäste. Aus Sicht des Datenmodells ist hier zu beachten: Die buchende Partei ist nicht identisch mit dem Reisenden. Daher sollte im Buchungsvorgang kenntlich sein, wer der *Auftraggeber* ist und wer die *Leistungsempfänger* sind. In TRIAS-Nachrichten lässt sich das durch separate Felder oder im Klartext angeben; im NeTeX-Datenmodell könnte eine Erweiterung BookingInitiator eingeführt werden, die im Falle von Drittbestellungen die Organisation oder Person des Auftraggebers referenziert. Alternativ wird für solche Fälle die Buchung über einen *Buchungskanal* "Telefon/Vermittler" vorgenommen, sodass implizit klar ist, dass es ein Vermittler (Praxispersonal) ist. Die Anzahl der benötigten Fahrten bzw. Fahrgäste kann entweder durch mehrere einzelne Buchungen (eine pro Patient) oder eine Gruppenbuchung mit Angabe der Personenzahl erfolgen. Der VSPV-Standard unterstützt beides: Eine

Gruppenbuchung kann über das Feld *partySize* in einer *BookingRequest* (TRIAS) dargestellt werden – im NeTeX statischen Modell sollte in solchen Fällen dennoch jeder Fahrtwunsch als separater *ServiceJourney* instanziiert werden, da Fahrten für verschiedene Personen im ÖPNV-Auskunftssystem in der Regel getrennt behandelt werden. Zusammengefasst: Das Datenmodell erlaubt Drittbestellern, mehrere Buchungen einzureichen, und hält über die Kostenträgerkennung fest, wer die Kosten trägt (z. B. die Klinik, die dann mit der Krankenkasse abrechnet). Eine spezielle Kennzeichnung „Sammelbestellung“ als solche ist im Modell nicht notwendig; es ergibt sich aus dem Auftreten mehrerer Buchungen durch dieselbe Auftraggeber-Identität innerhalb kurzer Zeit.

- **Krankenfahrten (nicht-qualifizierte Krankenförderung):** Hierunter fallen Fahrten, die aufgrund einer medizinischen Notwendigkeit durchgeführt werden (z. B. Taxi als Krankentransport zum Arzt oder Klinikfahrten mit Tragestuhl). Solche Fahrten sind oft genehmigungspflichtig (der Patient benötigt eine Verordnung) und werden von der Krankenkasse bezahlt. Im Datenmodell wird dies insofern berücksichtigt, als dass die Art der Fahrt gekennzeichnet werden kann. Für eine als *Krankenförderung* deklarierte Fahrt könnte z. B. ein Feld *MedicalTransportIndicator* = true eingeführt werden (etwa in einer Extension von *ServiceJourney* oder im *BookingRecord*). Dadurch können angeschlossene Systeme (Abrechnungsstellen, spezialisierte Anbieter) diese Fahrten herausfiltern und entsprechend behandeln. Außerdem wird – wie oben beschrieben – ein Leistungsnachweis erforderlich: Im Buchungsprozess muss hinterlegt sein, dass der Fahrgast eine gültige Verordnung hat (dies könnte durch Auswahl eines Tarifprodukts „Krankenfahrt“ geschehen, welches nur buchbar ist, wenn entsprechende Berechtigung vorliegt). Tariflich werden Krankenfahrten oft zum regulären Taxitarif abgerechnet, allerdings mit Kostenträger Krankenkasse. Im *FareFrame* kann hierzu entweder ein spezielles *FareProduct* „Krankenfahrt“ definiert werden oder man nutzt das normale Tarifschema und unterscheidet nur über die Zahlungsart (Drittzahler = Krankenkasse). Wichtig ist, dass im Buchungsdatensatz ein Verweis auf die Verordnung oder Genehmigungsnummer möglich ist – dies ließe sich als freies Textfeld im *BookingFrame* oder als Extension (z. B. <MedicalPermitNo>) lösen. Der VSPV-Standard schreibt vor, dass alle für die Kostenträgerrelevanten Sonderfahrten benötigten Informationen abbildbar sein müssen. Dazu zählt neben der Kennzeichnung als Krankenförderung auch die Angabe besonderer Bedürfnisse (etwa "Patient muss liegen" oder "Begleitperson erforderlich"), was im freien Text der Buchung oder in zukünftigen Erweiterungen spezifiziert werden kann. Solche zusätzlichen Hinweise können in NeTeX im Feld <Description> einer *ServiceJourney* oder als <BookingNote> im *BookingArrangement* untergebracht werden.
- **Rollstuhltaxi und inklusiver Transport:** Fahrten mit Rollstuhlfahrern erfordern geeignete Fahrzeuge und geschultes Personal. Im Datenmodell werden rollstuhlgerechte Fahrzeuge bereits im *ResourceFrame* über Attribute gekennzeichnet (z. B. *VehicleType* mit *wheelchairAccessible* = true). Für den Buchungsvorgang ist entscheidend, dass der Bedarf eines Rollstuhls angegeben

werden kann. TRIAS bietet in der BookingRequest typischerweise die Möglichkeit, Optionen wie „Rollstuhlplatz benötigt“ zu übermitteln. In NeTeX kann auf Ebene der Angebotspezifikation ein Flag wheelchairBooking = true vorgesehen werden, das bedeutet, dass das Angebot rollstuhlgerecht buchbar ist. Fehlt diese Kennzeichnung, sollte das System Buchungen mit Rollstuhloption ablehnen. Außerdem kann in den Buchungsregeln definiert sein, dass Rollstuhlfahrer nur telefonisch buchen sollen (wenn etwa eine genaue Abstimmung nötig ist) – in diesem Fall wären im BookingArrangement die Methoden eingeschränkt (phone statt online). Der VSPV-Standard verlangt, dass jede Einschränkung der Buchbarkeit für mobilitätseingeschränkte Personen transparent im Datenmodell dokumentiert wird. Beispielsweise könnten in einer keyList folgende Einträge stehen: `<Key>WHEELCHAIR_ACCESS</Key><Value>true</Value>` und `<Key>WHEELCHAIR_BOOKING_METHOD</Key><Value>phoneOnly</Value>`. Damit wird klar kommuniziert, dass das Angebot rollstuhlgerecht ist, die Buchung aber nur per Telefon erfolgen kann. Sollten Rollstuhltaxis andere Tarife haben (z. B. keine Zuschläge für Hilfsmittel), lässt sich das im FareFrame als besonderes Produkt abbilden; ansonsten gelten die Standardtarife. Für die Integration in Buchungs-Apps bedeutet diese Modellierung, dass entsprechende Optionen (Checkbox „Rollstuhl mitführen“ etc.) in der Nutzerschnittstelle auf das Vorhandensein solcher Flags im Datenmodell zurückgreifen.

Zusätzlich zu den genannten Szenarien muss das Datenmodell auch Fehler- und Ausnahmefälle handhaben, etwa wenn Buchungen storniert oder geändert werden. Stornierungen und Änderungen werden in NeTeX selbst nicht als separate Objekte geführt, können aber über TRIAS-Nachrichten (CancelBooking, ChangeBooking) abgebildet werden. Der ursprüngliche Buchungsvorgang bleibt davon unberührt; es wird lediglich sein Status aktualisiert. In der Datenhaltung kann vorgesehen werden, den Status (gebucht, storniert, geändert) als Attribut im BookingFrame mitzuführen, dies ist aber primär für betriebliche Systeme relevant, weniger für den Datenaustausch nach außen.

### **NeTeX-Datenstrukturen für Buchungs- und Zahlungsinformationen**

Die Integration der beschriebenen Buchungs- und Zahlungslogik erfolgt im Rahmen der bestehenden NeTeX-Strukturen gemäß VDV 462. Wo immer möglich, werden vorhandene Elemente genutzt (z. B. FlexibleLine, FlexibleServiceJourney, FareProduct). In Fällen, die über den Standard hinausgehen, kommen die in Abschnitt 3.3 beschriebenen Erweiterungsmechanismen zum Einsatz. Im Folgenden wird skizziert, wie die Daten in den relevanten Frames organisiert sind und welche Erweiterungen ggf. ergänzt werden:

- **ServiceFrame / TimetableFrame (Booking-Daten):** Buchungsrelevante Informationen zu Linien und Fahrten werden hauptsächlich im ServiceFrame und TimetableFrame gepflegt. Eine flexible Linie (FlexibleLine) enthält die Buchungsregeln im *BookingArrangement* (wie oben beschrieben) sowie Hinweise zur Betriebsweise. Konkrete Fahrten, die nur bei Bedarf stattfinden, werden als FlexibleServiceJourney im TimetableFrame modelliert. Jede FlexibleServiceJourney repräsentiert eine potentiell buchbare Fahrt (ggf. mit

Zeitfenster statt fester Zeit). Um die Integrationstiefe zu erhöhen, können an diesen Objekten **Extensions** verwendet werden: Beispielsweise erlaubt es der VSPV-Standard, an einer *FlexibleServiceJourney* einen *BookingLink* zu hinterlegen – einen direkten URL-Deeplink zur Buchungsseite – sowie Referenzen zu Echtzeitdatendiensten. Dies wurde bereits für Integrationsstufe 2 und 3 in Kapitel 3.3 demonstriert. Ebenfalls können hier einfache buchungsbezogene Attribute per KeyValue ergänzt werden (etwa für Mindestfahrweiten, Ausschlusskriterien etc.). Die Verwendung solcher Erweiterungen ist mit den VDV-462-Prinzipien abgestimmt und sollte restriktiv erfolgen – d. h. nur dort, wo Standard-NeTEx keine direkte Möglichkeit bietet. Alle *Extensions* sind im Namensraum vspv: oder als Key eindeutig zu benennen und zu dokumentieren, um die Interoperabilität zu gewährleisten.

- **FareFrame (Tarif- und Zahlungsdaten):** Alle tariflichen Informationen, inklusive Preise, Produkte und Zahlungsweisen, werden im FareFrame gebündelt. Für Gelegenheitsverkehre, insbesondere solche mit variablen Tarifen (z. B. Taxi-Meter), muss der FareFrame unter Umständen **erweitert** werden. Einfache Tarife (z. B. zonal oder pauschal) können mit Standard-NeTEx-Mitteln (FareZone, FareProduct, Tariff) modelliert werden. Komplexere Tarife wie Distanz- oder Zeitabhängige Taxi-Tarife, die in Taxentarifordnungen festgelegt sind, können als strukturierte Erweiterung eingebunden werden. Der Standard erlaubt z. B. die Abbildung eines Taxitarifs über ein <Extensions>-Element innerhalb des FareFrame, in dem die Parameter BaseFare, PerKm, Zuschläge etc. angegeben sind. Neben Preisen beinhaltet der FareFrame auch **Vertriebs- und Zahlungsinformationen**. Hier kommen *SalesOfferPackages* zum Einsatz, um zu definieren, auf welchen Vertriebswegen welche Tickets erhältlich sind und wie bezahlt werden kann. So könnte man ein *SalesOfferPackage* „Barverkauf im Fahrzeug“ definieren, mit *PaymentMethod* = cash, oder ein *Package* „App-Verkauf“ mit *PaymentMethod* = electronicPayment. In der Praxis kann ein Angebot mehrere *SalesOfferPackages* haben – eines pro erlaubter Zahlungsart bzw. Vertriebskanal. Sollte das Profil VDV-462 diesbezüglich Einschränkungen haben, werden diese im VSPV-Standard aufgehoben, um alle oben genannten Zahlarten abbilden zu können. Für die Angabe von *Berechtigungen* (z. B. Gutschein gültig nur für bestimmte Nutzergruppen) können Conditions im FareFrame genutzt werden; dies fällt jedoch eher in den Bereich Tarifbedingungen als in die Buchungslogik. Wichtig ist: **Zahlungsarten und Tarife sind im Datenmodell miteinander verknüpft**, sodass aus der Tarifbeschreibung hervorgeht, wie ein Fahrschein erworben und bezahlt werden kann.
- **Geplante Erweiterung:** BookingFrame/BillingFrame: Sollte es die Architektur erforderlich machen, können die Buchungs- und Abrechnungsdaten in eigene Rahmen ausgelagert werden. Der BookingFrame wäre ein logischer Container für alle buchungsbezogenen Entitäten, z. B. ein Objekt „Buchungsservice“ mit Kontaktinformationen, oder Vorlagen für Buchungsbestätigungen. Im aktuellen NeTEx-Standard existiert ein solcher Rahmen nicht ausdrücklich; der VSPV-Standard 101 empfiehlt daher, nach Möglichkeit innerhalb von ServiceFrame und

TimetableFrame zu bleiben, um Standardkompatibilität zu gewährleisten. Ein dedizierter BookingFrame könnte als *CompositeFrame* realisiert werden, der BookingArrangements oder Referenzdaten (z. B. eine Liste aller BookingMethods oder der Vertriebskanäle) enthält. Ähnlich könnte ein BillingFrame konzipiert werden, um Abrechnungselemente (Kostenstellen-Stammdaten, Rechnungsempfänger-Profile etc.) aufzunehmen. Da diese Ansätze über die gegenwärtigen Standards hinausgehen, sind sie *optionale* Erweiterungskonzepte. Sie kämen zum Einsatz, wenn ein Betreiber z. B. eine separate Austauschdatei nur für Buchungs-/Abrechnungsinformationen bereitstellen möchte. Grundsätzlich bleibt aber auch bei Nutzung solcher Frames die *Verknüpfung* zu den übrigen Daten gewahrt (z. B. Verweise von einer Buchung auf einen FareProduct im FareFrame).

Durch die klare Trennung der Daten in Frames und die Möglichkeit, mittels Extensions und KeyValue zusätzliche Informationen einzufügen, bleibt das Modell zugleich strukturiert und anpassungsfähig. Dies ermöglicht es, auch zukünftige neue Buchungs- und Zahlungsplattformen anzubinden, ohne das gesamte Modell ändern zu müssen. Neue Daten können in Extensions vorerst transportiert und bei breiterer Nutzung in künftige Standardversionen überführt werden. Wichtig ist dabei stets, die Kompatibilität mit bestehenden NeTEx-Parsern zu erhalten – d.h. Erweiterungen so zu gestalten, dass sie von Standardsoftware ignoriert werden können, falls diese die spezifischen Felder nicht kennt.

### **Interaktion mit Echtzeitsystemen (TRIAS) und Austauschprozessen**

Die in den vorherigen Abschnitten definierten Datenstrukturen bilden die statischen Grundlagen. Für die eigentliche Buchungsinteraktion in Echtzeit kommt üblicherweise die Schnittstelle TRIAS (bzw. eine vergleichbare API) zum Einsatz. Der VSPV-Standard 101 stellt sicher, dass seine Datenfelder mit den entsprechenden TRIAS-Elementen konsistent sind:

- **TRIAS BookingRequest:** Eine Buchungsanfrage im TRIAS-Format enthält alle wesentlichen Informationen, um eine Fahrt zu reservieren – z. B. die gewünschte Verbindung/Leg (meist referenziert durch eine Journey-/Trip-ID), die Anzahl der Personen, besondere Bedürfnisse (Rollstuhl etc.) und die präferierte Zahlungsart. Das VSPV-Datenmodell liefert die dafür nötigen Referenzen: Jede im NeTEx definierte *FlexibleServiceJourney* hat eine eindeutige ID, die in TRIAS als Referenz für die Buchung verwendet werden kann. Ebenso sind alle Haltepunkte, Tarifzonen usw. mit IDs versehen, die in der BookingRequest wiederverwendet werden. **Zahlungsinformationen:** TRIAS erlaubt es, in der BookingRequest einen Zahlungstyp oder Tarifwunsch anzugeben (dies kann je nach TRIAS-Version variieren, oft geschieht die Tarifwahl aber implizit oder in einem separaten FareRequest). Sollte TRIAS eine Auswahl der PaymentMethod zulassen, so müssen die Werte deckungsgleich mit den im Datenmodell definierten PaymentMethod-Codes sein (z. B. "bar", "karte", "invoice" etc.). Hier empfiehlt der VSPV-Standard, die englischen Codes gemäß NeTEx/VDV 462 auch in TRIAS zu verwenden, um Verwechslungen zu vermeiden. **Beispiel:** Ein Fahrgast wählt in der

App „Barzahlung“. Die App sendet in der TRIAS-BookingRequest `PaymentMethod = "cash"`; das Backend erkennt, dass "cash" im Datenmodell als Barzahlung definiert ist und weiß, dass keine Online-Zahlung erfolgen muss.

- **TRIAS FareRequest und FareResponse:** Vor der Buchung kann ein System über eine `FareRequest` den Fahrpreis für eine geplante Fahrt ermitteln. Die `FareResponse` wird auf Basis der `FareFrame`-Daten (Tariflogik) erstellt. Hier zählt sich die Modellierung der Tarife und Zuschläge im NeTEX aus: Das Auskunftssystem kennt durch den `FareFrame` alle Tarife (inkl. z. B. Taxi-Grundgebühr und km-Preis in `Extensions`) und kann so einen Preis berechnen. In der `FareResponse` kann dem Nutzer z. B. mitgeteilt werden: "Preis: 12,30 € bei Barzahlung; 12,00 € bei Online-Zahlung" – falls unterschiedliche `PaymentMethods` unterschiedliche Tarife haben (etwa Rabatt bei Vorauszahlung). Diese Logik muss im Datenmodell hinterlegt sein, z. B. als separates `FareProduct` für Online-Tarif vs. Bar-Tarif oder als `Rabatt-Condition`. Der `FareRequest` verwendet als Eingabe die `ServiceJourney-ID` oder die Wegangaben (`Origin/Destination`), die ebenfalls aus den NeTEX-Daten stammen. Der VSPV-Standard stellt also sicher, dass jede tarifrelevante Information (Entfernung, Zeit, Zuschläge) im Datenmodell vorhanden oder berechenbar ist, sodass eine `FareResponse` präzise erstellt werden kann.
- **TRIAS BookingConfirmation:** Nach Absenden einer `BookingRequest` liefert das System eine Bestätigung mit allen relevanten Details. Dazu gehören üblicherweise eine `Booking-ID`, die Fahrtdetails (Zeit, Fahrzeug etc.), ggf. ein Hinweis zur Zahlung sowie ein aggregierter Preis bzw. Abrechnungsbetrag (`chargeableAmount`). Letzterer wird aus dem `FareFrame` entnommen oder vom Tarifsysteem berechnet und muss in der `BookingConfirmation` strukturiert übergeben werden. Hierfür verwendet TRIAS ein `Fare-Element` innerhalb der Bestätigung, in dem z. B. `FareProduct`, `Amount`, `Currency` und `PaymentMethod` aufgeführt sind. Der VSPV-Standard verlangt, dass die `FareProduct-IDs` konsistent sind: Sollte also ein bestimmtes Tarifprodukt (z. B. "Rufbus-Ticket Erwachsene") verkauft werden, muss dessen ID im NeTEX-`FareFrame` definiert und in der `BookingConfirmation` referenziert sein. Ebenso wird die `PaymentMethod` erneut bestätigt (z. B. `PaymentMethod = cash` als Hinweis „Bitte zahlen Sie 12,30 € in bar beim Fahrer“). Durch diese Referenzierung kann ein nachgelagertes System (etwa ein Drittabrechner oder eine andere Behörde) eindeutig nachvollziehen, was gebucht und bezahlt werden soll.
- **Weitere TRIAS-Elemente:** Der Standard verweist ebenfalls auf TRIAS-Funktionen wie `CancelBooking`, `BookingStatus` usw., die hier zwar nicht im Detail beschrieben werden, aber eng mit dem Datenmodell verzahnt sind. Beispielsweise würde eine Stornierung (`CancelBooking`) die vorherige `Booking-ID` referenzieren, welche wiederum aus dem Datenmodell generiert wurde (häufig Kombination aus `ServiceJourney-ID` und `Laufnummer`). Ein `BookingStatusRequest` könnte verwendet werden, um den aktuellen Status einer Buchung abzufragen; die Antwort würde z. B. "assigned to vehicle XYZ" oder "completed" lauten –

Informationen, die im betrieblichen System vorhanden sind, aber im statischen Datenmodell höchstens indirekt (z. B. über Fahrzeugzuweisung in Echtzeitdaten) abgebildet werden. Wichtig ist: Das statische Datenmodell liefert die Grundlage, TRIAS füllt diese mit Live-Daten und Transaktionsdetails. Beide zusammen ermöglichen eine Tiefenintegration der Buchung in Auskunftssysteme, so dass Fahrgäste nahtlos von der Reiseauskunft zur Buchung und Bezahlung gelangen können, ohne Medienbrüche.

## Validierung und verpflichtende Datenfelder

Für die Buchungs- und Zahlungslogik gelten strenge Validierungsregeln, um eine hohe Datenqualität und Interoperabilität sicherzustellen. Viele Regeln aus den vorherigen Kapiteln (z. B. eindeutige IDs, gültige Referenzen, Sprachangaben) finden auch hier Anwendung. Ergänzend dazu sind folgende Felder und Zusammenhänge besonders zu prüfen:

- **Buchungsinformationen bei bedarfspflichtigen Verkehren:** Jede flexible Linie, die nur nach Buchung verkehrt (erkennbar an `FlexibleLineType` und fehlenden festen Fahrten), *muss* ein `BookingArrangement` mit mindestens einem `BookingMethod` und einem `BookingContact` besitzen. Fehlt dies, ist das Datenpaket ungültig. Beispiel Fehlermeldung: *„Buchungsinformation fehlt für FlexibleLine 'VSPV:Line:...': Keine BookingMethods angegeben.“* Entsprechende Prüfschritte sind im Schema bzw. durch Zusatzscripts sicherzustellen. Auch Kombinationen sind valide: Ist ein Angebot sowohl per App als auch Telefon buchbar, müssen beide Wege aufgeführt sein; umgekehrt darf bei einem rein telefonisch buchbaren Angebot nicht irrtümlich ein Online-Link im Datensatz stehen.
- **Kontaktangaben und Erreichbarkeit:** Zu jedem in `BookingMethods` genannten Kanal müssen die Kontaktdaten vollständig und formal korrekt angegeben sein. Für Telefonnummern gilt z. B. das Format gemäß internationaler Norm (nur Ziffern, Leerzeichen und führendes + zulässig). Wird ein Web-Link angegeben, muss dieser dem URL-Format genügen (Schema `http(s)://`). Validierungswerkzeuge prüfen syntaktisch die Felder und geben Fehler analog folgender Beispiele aus: *„PhoneNumber enthält ungültige Zeichen“* oder *„BookingLink ist keine gültige URL“*. Ferner soll überprüft werden, dass z. B. bei `BookingMethod phone` auch wirklich ein Phone-Kontakt hinterlegt ist (und nicht nur ein Url, was unlogisch wäre).
- **Pflicht zur Angabe der Zahlungsart:** Für jede tarifpflichtige Leistung muss mindestens eine zulässige `PaymentMethod` deklariert sein. Ist im `FareFrame` kein `PaymentMethod/Verkaufsangebot` definiert, wird angenommen, dass der Datensatz unvollständig ist. Fehlermeldung in diesem Fall: *„Zahlungsart für Tarif XYZ nicht definiert.“* Im Rahmen der Validierung wird auch die Konsistenz geprüft: z. B. darf ein `SalesOfferPackage` mit `PaymentMethod "cash"` nicht als Vertriebsweg "online" haben – solche Widersprüche würden zu einer Warnung oder Fehler führen. Der VSPV-Standard stellt hierzu ein Regelwerk bereit, das gängige

Kombinationen erlaubt (Barzahlung nur bei onBord-Verkauf, E-Payment nur bei Online-Verkauf etc.).

- **Abhängigkeiten bei Rechnung/Drittzahler:** Wie oben ausgeführt, dürfen bestimmte Felder nicht leer bleiben, wenn entsprechende Zahlungsarten gewählt wurden. Konkret: Bei *PaymentMethod* = *invoice* oder *thirdParty* muss ein gültiger Kostenträger angegeben sein. Die Validierung kann hier z. B. prüfen, ob ein Feld *costCarrierID* nicht leer ist oder ein Verweis auf Organisation existiert. Falls nicht, wird die Buchung entweder abgelehnt (im Online-Prozess) oder der Datensatz als fehlerhaft markiert. Ebenso muss – falls vom Angebot gefordert – die Angabe eines Berechtigungscode/Verordnung für Krankenfahrten erfolgen. Der Standard selbst kann die Gültigkeit dieser Codes nicht prüfen (das obliegt dem abrechnenden System), aber er stellt sicher, dass ein Feld dafür vorhanden ist und nicht null sein darf, sobald *MedicalTransportIndicator* = *true*. Solche Regeln werden in den Mappingtabellen dokumentiert und lassen sich in die XSD-Validierung nur bedingt integrieren, weshalb ergänzende Prüfskripte (z. B. Schematron-Regeln) zum Einsatz kommen.
- **Datenkonsistenz und Referenzen:** Alle Verknüpfungen zwischen Buchungs-/Zahlungsdaten und anderen Teilen des Modells sind konsistent zu halten. Beispielsweise muss jeder in einer *BookingConfirmation* genannte *FareProductRef* auf ein existierendes *FareProduct* im *FareFrame* zeigen. Jeder *BookingLink* (Deeplink) sollte idealerweise eine gültige Journey- oder Line-Referenz enthalten, die im Zielsystem aufgelöst werden kann – hier kann zwar nicht die XSD, aber ein integrativer Test prüfen, ob z. B. die Journey-ID im Link tatsächlich im gelieferten Datensatz vorkommt. Unstimmigkeiten werden als Fehler protokolliert.

Die technische Validierung erfolgt wie in Kapitel 3.1 beschrieben automatisiert gegen das NeTeX/VDV-462-XSD-Schema sowie via semantischer Prüfungen. Bei Verstößen gegen die oben genannten Regeln wird der Datensatz abgelehnt oder mit Warnungen versehen, die zu beheben sind, bevor ein Go-Live erfolgt. Ein Auszug möglicher Fehlermeldungen im Kontext Buchung/Zahlung:

- „*Buchungsweg 'online' angegeben, aber kein URL-Kontakt hinterlegt.*“
- „*Ungültiger PaymentMethod-Wert 'credit' – erlaubt sind: cash, electronic, invoice, voucher, thirdParty.*“
- „*Kostenstellen-ID fehlt bei Drittzahler-Buchung für Journey 12345.*“
- „*FlexibleServiceJourney 12345 als Krankenfahrt markiert, aber kein Verordnungsbezug angegeben.*“

Entwickler, Datenpfleger und Integratoren werden angehalten, vor jeder Datenlieferung eine lokale Validierung durchzuführen. Insbesondere bei den komplexen Buchungs- und Zahlungsdaten empfiehlt es sich, die bereitgestellten Beispiel-XML-Dateien und Mappingtabellen als Referenz heranzuziehen. Der VSPV-Standard stellt für typische Anwendungsfälle (z. B. Buchungslink-Einbindung, Kostenstellennutzung, Taxi-Tarif in

Extensions) validierte Beispiel-Dateien zur Verfügung. Diese dienen als Blaupause und erleichtern die Entwicklung konformer Schnittstellen in den operativen Systemen.

Durch die in Kapitel 3.4 definierten Strukturen wird die Buchungs- und Zahlungslogik im VSPV-Datenmodell nahtlos in die bestehende Kapitel-3-Struktur integriert. Alle erforderlichen Aspekte – von Buchungskanälen über Zahlungsarten bis hin zu Krankenfahrten – sind berücksichtigt und technisch modellierbar. Das Vorgehen orientiert sich streng an VDV 462 und NeTeX, ergänzt diese wo nötig aber gezielt um branchenspezifische Erweiterungen, um eine umfassende, interoperable und zukunftssichere Abbildung der Buchungs- und Zahlungsprozesse zu gewährleisten. Dies schafft die Voraussetzung dafür, dass Gelegenheitsverkehre gleichberechtigter Teil einer digital vernetzten Mobilitätslandschaft werden – von der Fahrgastinformation über die Buchung bis zur Bezahlung – wie im Zielbild des SDGV-Projekts vorgesehen.

### 3.5 Flexible Bedienformen und technische Objekte

Gelegenheitsverkehre gemäß § 46 PBefG zeichnen sich durch ein zentrales Merkmal aus: sie bedienen Adressen, keine Haltestellen. Anders als Linienverkehre – ob fahrplanbasiert oder in Form von Linienbedarfsverkehren nach § 44 PBefG – sind Gelegenheitsverkehre nicht an feste Betriebszeiten, Linienführungen oder Haltestellen gebunden. Vielmehr ermöglichen sie eine situativ nutzbare, ortsflexible Beförderung, insbesondere in ländlichen, suburbanen oder medizinisch unterversorgten Regionen. Die digitale Modellierung solcher flexiblen Bedienformen erfordert eine Erweiterung bestehender Datenstrukturen, die im VSPV-Standard 101 systematisch abgebildet wird.

Ziel dieses Abschnitts ist es, die für Gelegenheitsverkehre typischen technischen Objekte, ihre Beziehungen und die relevanten Konventionen darzustellen – einschließlich solcher Konfigurationen, bei denen adressbasierte Fahrten mit festen Anknüpfungspunkten (etwa Bahnhöfen oder Krankenhäusern) kombiniert werden.

#### **Merkmale flexibler Bedienformen im Gelegenheitsverkehr**

Typische Eigenschaften, die flexible Bedienformen im Gelegenheitsverkehr aufweisen, sind:

- Adressbedienung statt Haltestellenbedienung: Start- und Zielpunkt der Fahrt sind reale Adressen (z. B. Wohnadresse, Klinik, Pflegeheim), die im Modell als „FlexibleStopPlace“ mit exakter Geokoordinate abgebildet werden.
- Individuelle Buchung durch den Fahrgast oder autorisierte Dritte: Die Fahrten werden bedarfsorientiert gebucht – per App, Telefon oder Kiosksystem –, häufig auch im Auftrag (z. B. durch Arztpraxis, Krankenhaus).
- Flexibilität in der Fahrtplanung: Es gibt keine festgelegten Linien oder Taktfrequenzen. Fahrten werden je nach Nachfrage, Verkehrsaufkommen und Dispositionslogik individuell geplant und durchgeführt.
- Unterschiedliche Fahrzeugtypen: Vom regulären Taxi über Liegemietwagen bis zu rollstuhlgerechten Spezialfahrzeugen (z. B. BTWs) kommen verschiedene Beförderungsmittel zum Einsatz, die im Datenmodell als „VehicleType“ mit Merkmalen klassifiziert werden.

- Spezifische Tarif- und Abrechnungsmodelle: Es kommen sowohl dynamische (z. B. Taxameter-basierte) als auch statische Tarife (z. B. Festpreise) zum Einsatz. In bestimmten Fällen erfolgt die Abrechnung über Dritte, z. B. gesetzliche Krankenkassen bei Krankenfahrten.

Im Folgenden werden die zentralen technischen Objekte zur Abbildung flexibler Bedienformen dargestellt und durch ein praxistaugliches Beispiel veranschaulicht – im nächsten Teil folgt die konkrete Modellierung.

### 3.5.1 Technische Objekte zur Abbildung von Gelegenheitsverkehren

Die digitale Modellierung flexibler Bedienformen gemäß § 46 PBefG erfordert spezifische technische Objekte, die im VSPV-Standard 101 wie folgt definiert werden:

#### **FlexibleStopPlace (Adressbedienung und Sammelpunkte)**

Im Gelegenheitsverkehr werden Start- und Zielpunkte als reale Adressen (Hausnummer, Straße, Ort) oder alternativ als virtuelle Sammelpunkte definiert. Der `FlexibleStopPlace` dient dazu, diese Orte eindeutig und eindeutig geokodiert abzubilden.

**Adressbedienung:** Jeder tatsächlich buchbare Punkt (Adresse) erhält zur Laufzeit eine dynamische Geokoordinate. Die Adresse selbst muss nicht im Voraus als eigenes Objekt definiert sein, sondern wird durch ein allgemeines Bediengebiet mit Polygon-Geometrie definiert:

```
<FlexibleStopPlace id="VSPV:Bediengebiet:Hamm-Innenstadt"
version="1.0">
  <Name>
    <Text language="de">Bediengebiet Innenstadt Hamm</Text>
  </Name>
  <Polygon>
    <posList>
      51.6750 7.8150 51.6750 7.8250 51.6850 7.8250 51.6850 7.8150
51.6750 7.8150
    </posList>
  </Polygon>
</FlexibleStopPlace>
```

**Virtuelle Sammelpunkte:** Für bestimmte Fahrten (z. B. gebündelte Fahrten ab einem Krankenhaus) werden explizite virtuelle Sammelpunkte definiert, die fest verortet und mit eindeutiger Kennzeichnung versehen sind:

```
<StopPlace id="VSPV:StopPlace:Klinikum-Hamm" version="1.0">
  <Name>
    <Text language="de">Sammelpunkt Klinikum Hamm,
Haupteingang</Text>
  </Name>
  <Centroid>
```

```

<Location>
  <Longitude>7.8185</Longitude>
  <Latitude>51.6803</Latitude>
</Location>
</Centroid>
<PrivateCode>SAMMELPUNKT</PrivateCode>
</StopPlace>

```

Diese Sammelpunkte sind für Fahrgäste eindeutig sichtbar und ermöglichen eine einfache Buchung und Orientierung vor Ort.

### **FlexibleServiceJourney (Buchbare Fahrt)**

Jede buchbare Fahrt im Gelegenheitsverkehr wird als `FlexibleServiceJourney` abgebildet. Diese enthält alle relevanten Merkmale einer Fahrt, einschließlich Buchungsprozess, Fahrzeugtyp, Tarif und buchbarer Zeiten:

```

<FlexibleServiceJourney id="VSPV:ServiceJourney:Hamm-Zentral"
version="1.0">
  <Name>
    <Text language="de">Krankenfahrt Klinikum Hamm →
Wohnadresse</Text>
  </Name>
  <TransportMode>taxi</TransportMode>
  <JourneyPattern>
    <StopPointInJourneyPattern>
      <Order>1</Order>
      <StopPointRef ref="VSPV:StopPlace:Klinikum-Hamm"/>
    </StopPointInJourneyPattern>
    <StopPointInJourneyPattern>
      <Order>2</Order>
      <StopPointRef ref="VSPV:Bediengebiet:Hamm-Innenstadt"/>
    </StopPointInJourneyPattern>
  </JourneyPattern>
  <FlexibleServiceProperties id="VSPV:FlexProps:Klinikumfahrten"
version="1.0">
    <TypeOfFlexibleServiceRef
ref="VSPV:FlexibleServiceType:Adressbedienung"/>
    <BookingMethods>kiosk, telephone</BookingMethods>
    <MinimumBookingPeriod>PT30M</MinimumBookingPeriod>
  </FlexibleServiceProperties>
  <keyList>
    <KeyValue>
      <Key>KOSTENTRAEGER</Key>
      <Value>GKV:IKK-123456</Value>
    </KeyValue>
  </keyList>

```

```
</FlexibleServiceJourney>
```

Hier ist der Sammelpunkt „Klinikum Hamm“ als fester Startpunkt definiert, während der Zielpunkt flexibel im Bediengebiet „Innenstadt Hamm“ liegt, also eine beliebige Wohnadresse sein kann.

### **BookingProcess (Buchungsprozess)**

Im VSPV-Standard wird jede buchbare Fahrt mit einem `BookingProcess` verbunden, der festlegt, wie und über welche Kanäle (Telefon, Kiosk, App) gebucht wird, sowie Vorlaufzeiten und Öffnungszeiten definiert:

```
<BookingProcess id="VSPV:BookingProcess:Klinik-Hamm" version="1.0">
  <Name>
    <Text language="de">Krankenfahrt Kiosk Klinikum Hamm</Text>
  </Name>
  <bookingMethods>kiosk, telephone</bookingMethods>
  <latestBookingTime>PT30M</latestBookingTime>
  <bookingAccess>
    <KioskAccess>
      <StopPlaceRef ref="VSPV:StopPlace:Klinikum-Hamm"/>
    <OpeningTimes>
      <OpeningTime>
        <FromTime>07:00:00</FromTime>
        <ToTime>18:00:00</ToTime>
      </OpeningTime>
    </OpeningTimes>
  </KioskAccess>
  <TelephoneAccess>
    <PhoneNumber>02381 123456</PhoneNumber>
  </TelephoneAccess>
</bookingAccess>
</BookingProcess>
```

Dieses Objekt stellt sicher, dass alle notwendigen Buchungsinformationen und Zugangsmöglichkeiten zentral verfügbar sind und Auskunftssysteme oder Buchungskioske korrekt mit Daten versorgt werden.

### **VehicleType (Fahrzeugtyp)**

Um spezielle Anforderungen, insbesondere bei Krankenfahrten, zu berücksichtigen, werden Fahrzeugtypen detailliert definiert. Typische Merkmale sind:

- Fahrzeug mit Rollstuhlzugang (WHEELCHAIR\_ACCESSIBLE)
- Fahrzeug für liegende Beförderung ohne medizinische Betreuung (LIEGEMIETWAGEN)

```
<VehicleType id="VSPV:VehicleType:Liegemietwagen" version="1.0">
  <Name>
```

```

<Text language="de">Liegemietwagen (nicht-qualifizierter
Krankentransport)</Text>
</Name>
<keyList>
  <KeyValue>
    <Key>LIEGEMIETWAGEN</Key>
    <Value>>true</Value>
  </KeyValue>
</keyList>
</VehicleType>

```

Diese Fahrzeuge werden explizit referenziert, um eine korrekte Zuordnung und Disposition sicherzustellen.

### 3.5.2 Praxisbeispiel: Krankenfahrt per Kioskbuchung aus der Klinik

Ein Klinikum in Hamm verfügt über einen Buchungskiosk, an dem Mitarbeiter für Patienten direkt Krankenfahrten buchen können. Im konkreten Fall bucht die Klinikmitarbeiterin eine Fahrt für eine Patientin, die liegend ohne medizinische Betreuung (nicht-qualifizierter Krankentransport) zu ihrer Wohnadresse innerhalb der Innenstadt Hamm befördert werden soll.

#### **Ablauf der Buchung:**

1. Die Mitarbeiterin wählt am Kiosk den Fahrtbeginn („sofort“, mit 30 Minuten Vorlaufzeit), die Fahrzeugart (Liegemietwagen) und bestätigt die Patientendaten inklusive Kostenträger (gesetzliche Krankenkasse).
2. Das System validiert die Eingaben anhand der Buchungsregeln ( `BookingProcess` ) und übermittelt die Buchung an das zentrale Buchungssystem.
3. Das Buchungssystem prüft automatisch die Verfügbarkeit passender Fahrzeuge ( `VehicleType` ) für die angefragte Zeitspanne und bestätigt die Buchung. Die Fahrt wird disponiert und zugewiesen.
4. Die Fahrt startet pünktlich am definierten Sammelpunkt („Haupteingang Klinikum Hamm“) und endet direkt an der Wohnadresse der Patientin.

#### **Technische Abbildung im Modell:**

Die konkrete Buchung erzeugt im Datenmodell zur Laufzeit eine spezifische Instanz der zuvor definierten FlexibleServiceJourney, wobei der Zielpunkt (Adresse) dynamisch gesetzt wird:

```

<ServiceJourneyInterchange id="VSPV:Buchung:Klinikum-Hamm-001"
version="1.0">
  <FlexibleServiceJourneyRef ref="VSPV:ServiceJourney:Hamm-Zentral"/>
  <ActualDestinationLocation>
    <Longitude>7.8201</Longitude>
    <Latitude>51.6789</Latitude>
  </ActualDestinationLocation>

```

```

<Address>
  <Street>Marktstraße 12</Street>
  <Postcode>59065</Postcode>
  <Town>Hamm</Town>
</Address>
</ActualDestinationLocation>
<BookingStatus>confirmed</BookingStatus>
</ServiceJourneyInterchange>

```

Dieser Ablauf zeigt, wie die im VSPV-Standard definierten Objekte nahtlos ineinandergreifen, um flexible, adressbasierte Gelegenheitsverkehre technisch präzise und praktisch handhabbar zu machen.

## 3.6 Erweiterungsmechanismen (KeyValue & XML-Erweiterungen)

### 3.6.1 Zweck und Geltungsbereich

Dieses Kapitel definiert verbindlich, **wie** im VSPV-Standard 101 Informationen modelliert werden, die **nicht** im NeTEx-Kern oder den VDV-462-Profilen abbildbar sind. Dafür stehen zwei Mechanismen zur Verfügung:

1. **KeyValue in keyList** für einfache Zusatzattribute/IDs und
2. **<Extensions>** für strukturierte, komplexe Erweiterungen. VDV 462 beschreibt genau diese beiden Wege und deren Platzierung im NeTEx-Modell (u. a. Abschnitt „Proprietäre Erweiterungen“, inkl. Beispiel und Abbildung).

### 3.6.2 Entscheidungslogik: KeyValue vs. Extensions

Verwende **KeyValue (keyList)**, wenn

- es um **einfach typisierte Zusatzinfos** geht (Bool/Int/Decimal/String/ISO-Dauer),
- oder um **zusätzliche IDs/Referenzen**, die nicht im Standard vorgesehen sind. VDV 462 nennt KeyValue ausdrücklich für „zusätzliche IDs“ und empfiehlt eindeutige Schlüsselbezeichnungen.

Verwende **<Extensions>**, wenn

- eine **strukturierte Datenmenge** übertragen werden muss (verschachtelte Elemente),
- oder Du **mehrere zusammengehörige Felder** brauchst (z. B. Tarifkomponenten, Deeplink-Payload, komplexe Buchungsparameter). VDV 462 weist dafür auf <Extensions> hin und zeigt die zulässige Platzierung an Datenobjekten.

**Praxisregel im VSPV-Profil:**

- **Erst KeyValue prüfen**, nur wenn damit **keine** vollständige/valide Abbildung möglich ist → **<Extensions>** nutzen. Diese Priorisierung ist bereits in Eurem

Kapiteltext angelegt (KeyValue für Betriebsparameter wie MINDESTFAHRWEITE, INNERORTSVERBOT; Extensions für Deeplinks/Echtzeit-Referenzen).

### 3.6.3 Namensräume, Governance und Versionierung

- **KeyValue-Profil (VSPV):**
  - Alle zulässigen Schlüssel werden in einer **zentralen, versionierten Key-Registry** geführt (z. B. v1.0, v1.1).
  - Schlüssel sind **eindeutig, dokumentiert** (Datentyp, erlaubter Wertebereich, Semantik) und **rückwärtskompatibel** zu pflegen.
  - Bereits in Euren Projekttexten verwendete Schlüssel (z. B. MINDESTFAHRWEITE, INNERORTSVERBOT, FESTPREIS, POOLING\_ALLOWED, MAX\_DETOUR\_TIME, MAX\_KAPAZITAET) werden **unverändert übernommen** und in der Registry spezifiziert.
- **XML-Extensions:**
  - Eigene Elemente benutzen den **Namensraumpräfix vspv:** und werden in der Schnittstellendokumentation mit **XSD-Fragment** oder **Schematron-Regeln** versioniert.
  - Beispiele wie vspv:BookingLink, vspv:RealTimeDataRef und vspv:TaxiTariff sind bereits im Projekttext eingeführt und werden hiermit **verbindlich** gemacht.
- **Änderungsmanagement:**
  - Neue oder geänderte Keys/Extensions erhalten **Changelog, Beispielfiles** und **Mappingtabellen**; das entspricht den VDV-462-Empfehlungen zur Dokumentation.

### 3.6.4 KeyValue-Profil (verbindliche Schlüssel, Datentypen, Validierung)

Die folgende Auswahl konkretisiert bereits **im Projekt genutzte** Schlüssel. (Die vollständige Liste liegt in der zentralen Registry und wird dort gepflegt.)

Schlüssel	Typ	Zulässige Werte / Format	Semantik / Einsatz	Referenz
MINDESTFAHRWEITE	String (Distanz)	z. B. 3km (Regex: <code>^[0-9]+(\.[0-9]+)?(m)</code> )	km)\$` )	Mindestfahrweite für Annahme/Abrechnung
INNERORTSVERBOT	Bool	true/false	Fahrten innerhalb Ortsgrenzen ausgeschlossen	
FESTPREIS	Decimal	2 Nachkommastellen, EUR ohne	Festpreis für die Fahrt/Relation	

Schlüssel	Typ	Zulässige Werte / Format	Semantik / Einsatz	Referenz
		Währungssymbol, Punkt als Dezimaltrenner		
POOLING_ALLOWED	Bool	true/false	Pooling zulässig	
MAX_DETOUR_TIME	ISO-8601 Dauer	z. B. PT15M	maximaler Umweg je Buchung	
MAX_KAPAZITAET	Integer	≥ 1	maximale Sitzplätze im Service	
LEGACY_ID	String	max. 50, [A-Za-z0-9:_-]	Alt-ID zur Migration/Traceability	

### Validierung (KeyValue):

- Jeder Key ist **registry-pflichtig**; unbekannte Keys → **Fehler**.
- Datentyp/Format wird anhand Registry-Spezifikation geprüft (z. B. Regex für Distanzen/Dauern).
- KeyValue darf **keine Standardfelder duplizieren** (Designregel VDV 462: erst Standard prüfen, dann erweitern).

### Beispiel (KeyValue an FlexibleServiceJourney):

```
<FlexibleServiceJourney id="VSPV:ServiceJourney:POOL-2025-001"
version="1.0">
  <Name><Text language="de">Köln Ridepooling</Text></Name>
  <keyList>
    <KeyValue><Key>POOLING_ALLOWED</Key><Value>>true</Value></KeyV
alue>
    <KeyValue><Key>MAX_DETOUR_TIME</Key><Value>PT15M</Value></Key
Value>
    <KeyValue><Key>FESTPREIS</Key><Value>12.00</Value></KeyValue>
  </keyList>
</FlexibleServiceJourney>
```

### 3.6.5 XML-Erweiterungen (<Extensions>) – Struktur und Leitplanken

- **Namensraum:** xmlns:vspv="urn:vspv:extensions:1" (Versionierung am Namensraum).
- **Einsatzfälle:** mehrteilige Tarife, Deeplink-Payloads, Realtime-Verweise. In Euren Unterlagen finden sich dafür konkrete Beispiele (vspv:BookingLink, vspv:RealTimeDataRef, vspv:TaxiTariff).

- **Gestaltungsregeln:**
  - Elemente **selbsterklärend benennen**;
  - **Datentypen** fixieren (XSD oder Schematron);
  - **keine Pflichtfelder duplizieren**;
  - **kompatibel** zu Standard-Parsern (Unbekanntes wird ignoriert, Struktur bleibt valide). VDV 462 empfiehlt genau dieses Vorgehen.

#### Beispiel A (Deeplink + Realtime-Verweis an FlexibleServiceJourney):

```
<FlexibleServiceJourney id="VSPV:Journey:2025-Q3-501" version="1.0">
  <Extensions>
    <vspv:BookingLink>https://booking.vspv.de?journey=501</vspv:BookingLink>
    <vspv:RealTimeDataRef>RTD_VSPV_501</vspv:RealTimeDataRef>
  </Extensions>
</FlexibleServiceJourney>
```

#### Beispiel B (Taxi-Tarifstruktur an FareStructure):

```
<FareStructure id="VSPV:Fare:CGN-City-Fixed" version="1.0">
  <Extensions>
    <vspv:TaxiTariff>
      <vspv:BaseFare>4.50</vspv:BaseFare>
      <vspv:PerKm>2.20</vspv:PerKm>
      <vspv:NightSurcharge>1.00</vspv:NightSurcharge>
    </vspv:TaxiTariff>
  </Extensions>
</FareStructure>
```

### 3.6.6 Validierung und PrüfregeIn

- **XSD-Validierung:** NeTEx-Schema **plus** optionales **VSPV-Extension-XSD** (Namensraum urn:vspv:extensions:\*).
- **Schematron (empfohlen):**
  - **Key-Existenz:** (Key muss im Registry-Vokabular vorhanden sein)
  - **Key-Datentyp:** (Regex/Typprüfung je Key)
  - **<Extensions>-Inhalte:** Pflichtknoten/Datentypen pro VSPV-Struktur. VDV 462 fordert explizit saubere Dokumentation/Prüfbarkeit proprietärer Erweiterungen; Eure Kapitel 2/3 nennen Mappingtabellen als QS-Instrument.

### 3.6.7 Kompatibilität, Migration und Interoperabilität

- **Rückwärtskompatibilität:** neue Keys **additiv** einführen; Entfernen/Umbenennen nur mit Major-Version der Registry.

- **Migration Altmerkmale:** wenn Legacy-IDs/Attribute nötig sind → **KeyValue LEGACY\_ID** verwenden; das passt zur VDV-462-Empfehlung „KeyValue für zusätzliche IDs“.
- **Konverter (GTFS/GTFS-flex):** Keys/Extensions werden in Konvertern **gemappt** oder (falls nicht darstellbar) **neutralisiert**; Euer Kapitel 2 skizziert diesen Ansatz bereits.

### 3.6.8 Beispiel-Set

#### 1) FlexibleStopPlace mit Betriebsparametern (KeyValue):

```
<FlexibleStopPlace id="VSPV:StopPlace:NRW-1001" version="1.0">
  <Name><Text language="de">Taxi-Bediengebiet
Innenstadt</Text></Name>
  <keyList>
<KeyValue><Key>MINDESTFAHRWEITE</Key><Value>3km</Value></KeyV
alue>
<KeyValue><Key>INNERORTSVERBOT</Key><Value>>false</Value></KeyV
alue>
  </keyList>
</FlexibleStopPlace>
```

#### 2) FlexibleServiceJourney mit Festpreis + Pooling (KeyValue) und Deeplink (Extensions):

```
<FlexibleServiceJourney id="VSPV:ServiceJourney:POOL-2025-010"
version="1.0">
  <Name><Text language="de">Taxi Innenstadt → Bahnhof</Text></Name>
  <keyList>
  <KeyValue><Key>FESTPREIS</Key><Value>12.00</Value></KeyValue>
<KeyValue><Key>POOLING_ALLOWED</Key><Value>>true</Value></KeyV
alue>
<KeyValue><Key>MAX_DETOUR_TIME</Key><Value>PT10M</Value></Key
Value>
  </keyList>
  <Extensions>
  <vspv:BookingLink>https://booking.vspv.de?journey=POOL-2025-
010</vspv:BookingLink>
  <vspv:RealTimeDataRef>RTD_VSPV_POOL_010</vspv:RealTimeDataRef>
  </Extensions>
</FlexibleServiceJourney>
```

#### 3) FareStructure mit strukturierter Taxi-Tarifkomponente (Extensions):

(siehe Beispiel B oben)

### 3.6.9 Mappingtabelle (Auszug)

Fachliche Anforderung	NeTeX/VDV-462-Struktur	VSPV-Erweiterung	Konvention/Prüfung
Deeplink zur Buchung	FlexibleServiceJourney	<Extensions><vspv:BookingLink>	Gültige URL, max. 255, RFC-konform; Pflicht, wenn Integrationsstufe 2
Echtzeit-Referenz	FlexibleServiceJourney	<Extensions><vspv:RealTimeDataRef>	Referenz muss im Realtime-Backend existieren
Festpreis pro Relation	FlexibleServiceJourney	keyList/FESTPREIS	Decimal, 2 Nachkommastellen, EUR ohne Symbol
Betriebsparameter (z. B. Mindestfahrweite)	FlexibleStopPlace	keyList/MINDESTFAHRWEITE	Regex-Prüfung ( `m
Strukturierter Taxi-Tarif	FareStructure	<Extensions><vspv:TaxiTariff>	XSD/Schematron-geprüft, Pflichtfelder BaseFare/PerKm

### 3.6.10 Konformitätskriterien (prüfbar)

Eine Datenlieferung ist **kapitelkonform**, wenn:

1. **KeyValue nur registrierte Schlüssel** mit korrektem Datentyp enthält; **Duplikate Standardfelder → Fehler.** (VDV 462-Leitlinie)
2. **<Extensions> ausschließlich vspv-Namensräume** nutzt und gegen die VSPV-Extension-Regeln (XSD/Schematron) validiert.
3. **Beispiel-XMLs und Mappingtabellen** die Erweiterung dokumentieren (Version, Zweck, Felder), wie im Projekt verankert.

### 3.6.11 Anhang 3.6-A: VSPV KeyValue-Registry v0.91 (Auszug)

Zweck: Verbindliche Definition zulässiger Schlüssel, Datentypen, Formate, Geltungsbereich und Beispiele.

Geltung: Kapitel 3.6 (Erweiterungsmechanismen). Die vollständige Liste wird in der zentralen Registry gepflegt (Versionierung: v1.0, v1.1, ...).

Schlüssel (Key)	Datentyp	Format / Regex	Geltungsbereich (Objektyp)	Kard.	Semantik	Beispiel
<b>MINDESTFAHRWEITE</b>	Distanz (String)	<code>^[0-9]+(\.[0-9]+)?(m</code>	<code>km)\$`</code>	FlexibleStopPlace (inkl. Bediengebiet), ggf. Service	0..1	Mindestfahrweite für Annahme/Abrechnung
<b>INNERORTSVERBOT</b>	Bool	<code>^(true</code>	<code>false)\$`</code>	FlexibleStopPlace / Service	0..1	Fahrten innerhalb Ortsgrenzen ausgeschlossen
<b>FESTPREIS</b>	Decimal (EUR)	<code>^[0-9]+(\.[0-9]{2})\$</code>	FlexibleServiceJourney	0..1	Festpreis je Relation/Fahrt (ohne Währungssymbol)	12.00
<b>POOLING_ALLOWED</b>	Bool	<code>^(true</code>	<code>false)\$`</code>	FlexibleServiceJourney	0..1	Pooling zulässig
<b>MAX_DETOUR_TIME</b>	ISO-8601 Dauer	<code>^P(T(\d+H)?(\d+M)?(\d+S)?</code>	<code>\d+D)\$`</code>	FlexibleServiceJourney	0..1	max. Umweg pro Buchung
<b>MAX_KAPAZITAET</b>	Integer	<code>^[1-9][0-9]*\$</code>	FlexibleServiceJourney, VehicleType	0..1	maximale Sitzplätze	4
<b>KOSTENTRAEGER</b>	String (Code)	<code>^(GKV</code>	PKV	KOMMUNE):[A-Za-z0-9-_/]{1,32}\$`	FlexibleServiceJourney, Fare, Billing	0..1

Schlüssel (Key)	Datentyp	Format / Regex	Geltungsbereich (Objektyp)	Kard.	Semantik	Beispiel
WHEELCHAIR_ACCESSIBLE	Bool	^(true	false)\$`	VehicleType	0..1	rollstuhlgerechtes Fahrzeug
VEHICLE_PROPULSION	Enum	^(electric	diesel	hybrid	hydrogen)\$`	VehicleType
LIEGEMIETWAGEN	Bool	^(true	false)\$`	VehicleType	0..1	Fahrzeug für liegende Beförderung (ohne med. Betreuung)
LEGACY_ID	String	^[A-Za-z0-9:_-]{1,50}\$	alle Objekttypen	0..n	Alt-ID zur Migration/Traceability	DIVA:Stop:4711

#### Konventionen (Registry):

- Jeder Key ist **eindeutig, versioniert** (Registry-Version), mit **Semantik, Datentyp, Regex, Geltungsbereich** und Beispiel dokumentiert.
- Keys **duplizieren keine Standardfelder**; zuerst Standard prüfen, dann KeyValue nutzen.
- Pro Objekt darf ein Key **höchstens einmal** gesetzt werden (außer LEGACY\_ID, 0..n).
- Erweiterungen an der Liste erfolgen via **Minor-Version** (additiv); Breaking-Changes nur mit **Major-Version**.

### 3.6.12 Anhang 3.6-B: Schematron-Snippets (Validierung)

Zweck: Maschinelle Prüfung der Registry-Regeln (zulässige Keys, Datentyp/Format, Geltungsbereich, Eindeutigkeit).

Hinweis: Wir verwenden local-name() für Namespacetoleranz, damit die Regeln sowohl mit als auch ohne explizites netex:-Präfix greifen.

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
queryBinding="xslt2">
  <ns prefix="vspv" uri="urn:vspv:extensions:1"/>

  <!-- Hilfsfunktionen via let-Variablen sind in Schematron nicht nativ;
  wir arbeiten mit wiederholten Ausdrücken und local-name(). -->

  <!-- ===== 1) Allgemeine Regeln für keyList/KeyValue
  ===== -->
  <pattern id="key-general">
    <!-- Key darf je Objekt nur einmal vorkommen (LEGACY_ID
    ausgenommen) -->
    <rule context="*[local-name()='keyList']/*[local-name()='KeyValue']">
      <assert test="count(preceding-sibling::*[local-
      name()='KeyValue']*[local-name()='Key']=current())/*[local-name()='Key']]
      = 0
          or normalize-space(./*[local-name()='Key']) = 'LEGACY_ID'">
        [KV-001] Key darf pro Objekt nur einmal vorkommen (Ausnahme:
        LEGACY_ID).
      </assert>
      <!-- Key muss in Registry stehen (Whitelisting der Start-Registry) -->
      <assert test="normalize-space(./*[local-name()='Key']) =
      ('MINDESTFAHRWEITE','INNERORTSVERBOT','FESTPREIS','POOLING_ALLO
      WED','MAX_DETOUR_TIME','MAX_KAPAZITAET','KOSTENTRAEGER','WHEELC
      HAIR_ACCESSIBLE','VEHICLE_PROPULSION','LIEGEMIETWAGEN','LEGACY
      _ID')">
        [KV-002] Unbekannter Key: <value-of select=".*[local-
        name()='Key']"/>.
      </assert>
    </rule>
  </pattern>

  <!-- ===== 2) Geltungsbereiche (welche Keys an welchen
  Objekten?) ===== -->
  <pattern id="key-scope">
    <!-- FlexibleStopPlace: MINDESTFAHRWEITE, INNERORTSVERBOT -->
```

```

<rule context="*[local-name()='FlexibleStopPlace']/*[local-
name()='keyList']">
  <assert test="not(./*[local-name()='KeyValue']/*[local-
name()='Key']='FESTPREIS')">
    [KV-101] FESTPREIS ist an FlexibleStopPlace unzulässig.
  </assert>
</rule>

<!-- FlexibleServiceJourney: FESTPREIS, POOLING_ALLOWED,
MAX_DETOUR_TIME, MAX_KAPAZITAET, KOSTENTRAEGER -->
<rule context="*[local-name()='FlexibleServiceJourney']/*[local-
name()='keyList']">
  <assert test="not(./*[local-name()='KeyValue']/*[local-
name()='Key']='MINDESTFAHRWEITE')
    and not(./*[local-name()='KeyValue']/*[local-
name()='Key']='INNERORTSVERBOT')">
    [KV-102] MINDESTFAHRWEITE/INNERORTSVERBOT gehören nicht an
FlexibleServiceJourney.
  </assert>
</rule>

<!-- VehicleType: WHEELCHAIR_ACCESSIBLE, VEHICLE_PROPULSION,
LIEGEMIETWAGEN -->
<rule context="*[local-name()='VehicleType']/*[local-name()='keyList']">
  <assert test="not(./*[local-name()='KeyValue']/*[local-
name()='Key']='FESTPREIS')">
    [KV-103] FESTPREIS ist an VehicleType unzulässig.
  </assert>
</rule>
</pattern>

<!-- ===== 3) Wert-/Formatprüfungen je Key ===== -->
<pattern id="key-values">
  <!-- Bool -->
  <rule context="*[local-name()='KeyValue']/*[local-
name()='Key']='INNERORTSVERBOT' or *[local-
name()='Key']='POOLING_ALLOWED' or *[local-
name()='Key']='WHEELCHAIR_ACCESSIBLE' or *[local-
name()='Key']='LIEGEMIETWAGEN'">
    <assert test="matches(normalize-space(./*[local-name()='Value']),
'^(true|false)$')">
      [KV-201] Boolescher Wert erwartet (true|false).
    </assert>
  </rule>

```

```

<!-- FESTPREIS -->
<rule context="*[local-name()='KeyValue']*[local-
name()='Key']='FESTPREIS'">
  <assert test="matches(normalize-space(./*[local-name()='Value']),
'^[0-9]+(\.[0-9]{2})$')">
    [KV-202] FESTPREIS muss als Dezimalzahl mit genau zwei
Nachkommastellen vorliegen (z. B. 12.00).
  </assert>
</rule>

<!-- MINDESTFAHRWEITE -->
<rule context="*[local-name()='KeyValue']*[local-
name()='Key']='MINDESTFAHRWEITE'">
  <assert test="matches(normalize-space(./*[local-name()='Value']),
'^[0-9]+(\.[0-9]+)?(m|km)$')">
    [KV-203] MINDESTFAHRWEITE muss in m oder km angegeben werden
(z. B. 3km, 250m).
  </assert>
</rule>

<!-- MAX_DETOUR_TIME -->
<rule context="*[local-name()='KeyValue']*[local-
name()='Key']='MAX_DETOUR_TIME'">
  <assert test="matches(normalize-space(./*[local-name()='Value']),
'^P(T(\d+H)?(\d+M)?(\d+S)?|\d+D)$')">
    [KV-204] MAX_DETOUR_TIME muss als ISO-8601-Dauer angegeben
werden (z. B. PT15M).
  </assert>
</rule>

<!-- MAX_KAPAZITAET -->
<rule context="*[local-name()='KeyValue']*[local-
name()='Key']='MAX_KAPAZITAET'">
  <assert test="matches(normalize-space(./*[local-name()='Value']),
'^[1-9][0-9]*$')">
    [KV-205] MAX_KAPAZITAET muss eine positive ganze Zahl sein (≥1).
  </assert>
</rule>

<!-- VEHICLE_PROPULSION -->
<rule context="*[local-name()='KeyValue']*[local-
name()='Key']='VEHICLE_PROPULSION'">
  <assert test="matches(normalize-space(./*[local-name()='Value']),
'^(electric|diesel|hybrid|hydrogen)$')">

```

```

    [KV-206] VEHICLE_PROPULSION muss einer der Werte
    {electric|diesel|hybrid|hydrogen} sein.
    </assert>
  </rule>

  <!-- KOSTENTRAEGER -->
  <rule context="*[local-name()='KeyValue']*[local-
  name()='Key']='KOSTENTRAEGER'">
    <assert test="matches(normalize-space(./*[local-name()='Value']),
    '^((GKV|PKV|KOMMUNE):[A-Za-z0-9\-\_\ \.]){1,32}$)">
      [KV-207] KOSTENTRAEGER muss als Namensraum-prefixed Code
      angegeben werden (z. B. GKV:IKK-123456).
    </assert>
  </rule>

  <!-- LEGACY_ID -->
  <rule context="*[local-name()='KeyValue']*[local-
  name()='Key']='LEGACY_ID'">
    <assert test="matches(normalize-space(./*[local-name()='Value']),
    '^([A-Za-z0-9:_-]){1,50}$)">
      [KV-208] LEGACY_ID muss 1..50 Zeichen aus [A-Za-z0-9:_-] enthalten.
    </assert>
  </rule>
</pattern>

<!-- ===== 4) Konsistenz zwischen KeyValue und Extensions
===== -->
<pattern id="kv-vs-extensions">
  <!-- Entweder FESTPREIS ODER strukturierter vspv:TaxiTariff, aber nicht
  beides -->
  <rule context="*[local-name()='FlexibleServiceJourney'">
    <assert test="not( ./*[local-name()='keyList']/*[local-
    name()='KeyValue']*[local-name()='Key']='FESTPREIS' )
    or not( ./*[local-name()='Extensions']/vspv:TaxiTariff)">
      [KV-301] FESTPREIS und strukturierter TaxiTariff dürfen nicht
      gleichzeitig gesetzt sein.
    </assert>
  </rule>
</pattern>
</schema>

```

#### Einsatzhinweise:

- **Namensräume:** Wenn eine XML die NeTeX-Elemente mit Präfix (z. B. netex:) führt, kann die Schematron wie oben (mit local-name()) **unverändert** genutzt werden.

- **Erweiterung:** Weitere Keys ergänzt man, indem man **[KV-002]** um die zulässigen Keynamen erweitert **und** neue Wertprüfungen (Pattern *key-values*) hinzufügt.
- **Build-Pipeline:** Schematron in die bestehende Validierungskette einhängen (XSD → Schematron). Fehlertexte sind so formuliert, dass sie direkt in das Validierungsprotokoll übernommen werden können.

### 3.6.13 Anhang 3.6-C: Mini-Registry als YAML (für Tooling)

Für die **maschinenlesbare** Pflege der Registry ein startfähiges YAML-Fragment:

```

version: 0.91
keys:
  MINDESTFAHRWEITE:
    type: distance
    regex: '^[0-9]+(\.[0-9]+)?(m|km)$'
    scope: [FlexibleStopPlace, Service]
    cardinality: '0..1'
    example: '3km'
  INNERORTSVERBOT:
    type: boolean
    regex: '^(true|false)$'
    scope: [FlexibleStopPlace, Service]
    cardinality: '0..1'
    example: 'false'
  FESTPREIS:
    type: money
    regex: '^[0-9]+(\.[0-9]{2})$'
    scope: [FlexibleServiceJourney]
    cardinality: '0..1'
    example: '12.00'
  POOLING_ALLOWED:
    type: boolean
    regex: '^(true|false)$'
    scope: [FlexibleServiceJourney]
    cardinality: '0..1'
    example: 'true'
  MAX_DETOUR_TIME:
    type: duration
    regex: '^P(T(\d+H)?(\d+M)?(\d+S)?|\d+D)$'
    scope: [FlexibleServiceJourney]
    cardinality: '0..1'
    example: 'PT15M'
  MAX_KAPAZITAET:
    type: integer
    regex: '^[1-9][0-9]*$'
    scope: [FlexibleServiceJourney, VehicleType]

```

```

cardinality: '0..1'
example: '4'
KOSTENTRAEGER:
type: code
regex: '^(GKV|PKV|KOMMUNE):[A-Za-z0-9\-\_\.\-]{1,32}$'
scope: [FlexibleServiceJourney, Fare, Billing]
cardinality: '0..1'
example: 'GKV:IKK-123456'
WHEELCHAIR_ACCESSIBLE:
type: boolean
regex: '^(true|false)$'
scope: [VehicleType]
cardinality: '0..1'
example: 'true'
VEHICLE_PROPULSION:
type: enum
enum: ['electric','diesel','hybrid','hydrogen']
scope: [VehicleType]
cardinality: '0..1'
example: 'electric'
LIEGEMIETWAGEN:
type: boolean
regex: '^(true|false)$'
scope: [VehicleType]
cardinality: '0..1'
example: 'true'
LEGACY_ID:
type: string
regex: '^[A-Za-z0-9:\-]{1,50}$'
scope: ['*']
cardinality: '0..n'
example: 'DIVA:Stop:4711'

```

## 3.7 REST-API-Schnittstellen für Buchung und Verfügbarkeit

### 3.7.1 Ziel und Abgrenzung

Dieses Kapitel spezifiziert eine **REST-Schnittstelle** zur **Verfügbarkeitsprüfung**, **Preisermittlung (Quote)**, **Buchung**, **Stornierung** und **Statusverfolgung** für **Gelegenheitsverkehre** mit **Adressbedienung**. Die API ist **ressourcenorientiert**, **idempotent** bei Anlagevorgängen (über Idempotency-Key) und **token-gesichert** (OAuth2/OIDC). **Echtzeit-Events** (Positions-/Statusupdates) und **Auskunftsschnittstellen** werden in **3.8** behandelt.

### 3.7.2 Grundlegende Designprinzipien

- **Versionierung:** Basis-Pfad /v1/. Deprecation via Header Sunset + Migrationshinweis.
- **Zeiten/Zeitzone:** ISO-8601 (2025-09-01T14:05:00+02:00). Keine implizite Server-TZ.
- **IDs & Feldgrößen:** Gemäß 3.3 (z. B. id ≤ 50, Pattern [A-Za-z0-9:\_-]).
- **Namensräume/Referenzen:** Verweise auf NeTeX/VDV-462-Objekte als ref (z. B. StopPlaceRef, FlexibleServiceJourneyRef).
- **Fehlerformat:** Einheitlich (type, code, message, field, correlationId).
- **Idempotenz:** Header Idempotency-Key (UUIDv4). Wiederholte POST mit gleichem Key → 200/201 mit alter Antwort.
- **Ratenbegrenzung:** Antwortheader X-RateLimit-Limit/Remaining/Reset.
- **Sicherheit/Scopes:** OAuth2/OIDC; Scopes z. B. availability:read, booking:write, booking:read, payment:write.
- **Datenschutz/PII:** Datenminimierung; Vertreterbuchung (Arztpraxis/Klinik) kennzeichnen; sensible Daten nur, wenn zwingend erforderlich (siehe 3.4/3.5).

### 3.7.3 Ressourcenübersicht

Ressource	Methode	Zweck	Synchronität	Antwort
/v1/availability	POST	Verfügbarkeit & Optionen (Zeitfenster, Fahrzeugmerkmale, Kapazität)	sync	200 mit Optionsliste
/v1/quotes	POST	Preisermittlung („Quote“) für konkrete Anfrage	sync	201 (Quote), enthält Tarifiedetails
/v1/bookings	POST	Buchung anlegen (App/Kiosk/Telefon)	<b>sync (201)</b> oder <b>async (202)</b>	Buchung/Accept-Token
/v1/bookings/{id}	GET	Buchungsstatus & Details	sync	200
/v1/bookings/{id}/cancel	POST	Stornierung	sync	200
/v1/payments	POST	Zahlungsinitialisierung (App/Kiosk)	sync	201 (Payment-Intent)
/v1/payments/{id}/capture	POST	Zahlung einziehen	sync	200

Ressource	Methode	Zweck	Synchronität	Antwort
/v1/webhooks	POST	Empfangs-Endpunkt beim Partner (für Events)	async	2xx vom Partner
/v1/events	GET	(Optional) Pull-Mechanismus für Events	sync (Long-Polling)	200

Hinweis: Für Endkunden-Apps wird i. d. R. **Quote** → **Booking** → **Payment** sequentiell genutzt. Kiosk/Telefon kann **Booking (mit Kostenträger)** → **Payment (Rechnung/Bar)** nutzen.

### 3.7.4 Datenmodelle (REST) und Mapping zu NeTEx/VDV-462

#### 3.7.4.1 Verfügbarkeit prüfen

**Request** /v1/availability (POST)

```
{
  "origin": {
    "type": "StopPlaceRef|Address",
    "ref": "VSPV:StopPlace:Klinikum-Hamm",
    "address": null
  },
  "destination": {
    "type": "Address|StopPlaceRef",
    "address": {
      "street": "Marktstraße 12",
      "postcode": "59065",
      "town": "Hamm",
      "location": { "lat": 51.6789, "lon": 7.8201 }
    },
    "ref": null
  },
  "earliestDepartureTime": "2025-09-01T10:00:00+02:00",
  "latestArrivalTime": "2025-09-01T12:00:00+02:00",
  "passengers": 1,
  "requirements": {
    "wheelchairAccessible": true,
    "lyingTransport": true
  }
}
```

**Response** 200

```
{
  "options": [
```

```

{
  "flexibleServiceJourneyRef": "VSPV:ServiceJourney:Hamm-Zentral",
  "vehicleTypeRef": "VSPV:VehicleType:Liegemietwagen",
  "pickupWindow": { "from": "2025-09-01T10:20:00+02:00", "to": "2025-09-01T10:40:00+02:00" },
  "estArrivalTime": "2025-09-01T11:15:00+02:00",
  "poolingPossible": false,
  "pricingModel": "fixed|meter",
  "currency": "EUR"
}
],
"correlationId": "8e2a6d1e-..."
}

```

### Mapping (NeTex):

origin/destination → StopPlaceRef oder Adresspunkt innerhalb

FlexibleStopPlace/Bediengebiet;

flexibleServiceJourneyRef → FlexibleServiceJourney.id;

vehicleTypeRef → VehicleType.id.

#### 3.7.4.2 Quote (Preisermittlung)

##### Request /v1/quotes (POST)

```

{
  "journey": {
    "flexibleServiceJourneyRef": "VSPV:ServiceJourney:Hamm-Zentral",
    "originRef": "VSPV:StopPlace:Klinikum-Hamm",
    "destinationAddress": { "street": "Marktstraße 12", "postcode": "59065",
    "town": "Hamm" }
  },
  "passengers": 1,
  "requirements": { "wheelchairAccessible": true, "lyingTransport": true },
  "payer": { "type": "KOSTENTRAEGER|CUSTOMER", "code": "GKV:IKK-123456" }
}

```

##### Response 201

```

{
  "quoteId": "q_01H8D85Y...",
  "fare": {
    "type": "fixed|meter|composed",
    "amount": 27.00,
    "currency": "EUR",
    "conditions": [ "Gilt für Start innerhalb 60 Minuten" ]
  },
  "validUntil": "2025-09-01T10:10:00+02:00"
}

```

**Mapping (NeTex):** Quote basiert auf `FareStructure/FareProduct` bzw. `keyList/FESTPREIS` oder `Extensions/vspv:TaxiTariff`.

### 3.7.4.3 Buchung anlegen

**Request** `/v1/bookings` (POST) (mit Idempotency-Key)

```
{
  "quoteId": "q_01H8D85Y...",
  "customer": {
    "type": "PERSON|ORGANISATION",
    "name": "Anna Beispiel",
    "contact": { "phone": "+49 2381 123456", "email": "anna@example.org" },
    "actingOnBehalf": { "type": "HOSPITAL", "name": "Klinikum Hamm",
    "authorisationRef": "KH-Case-7788" }
  },
  "requirements": { "wheelchairAccessible": true, "lyingTransport": true },
  "payment": {
    "mode": "INVOICE|CARD|CASH|DIRECT_DEBIT",
    "payer": { "type": "KOSTENTRAEGER|CUSTOMER", "code": "GKV:IKK-123456" }
  },
  "consents": { "termsAccepted": true, "dataProcessingAccepted": true }
}
```

**Response**

- **201 Created** (sofortige Bestätigung möglich)

```
{
  "bookingId": "b_01H8D8...",
  "status": "CONFIRMED",
  "pickupWindow": { "from": "2025-09-01T10:25:00+02:00", "to": "2025-09-01T10:35:00+02:00" },
  "vehicleTypeRef": "VSPV:VehicleType:Liegemietwagen",
  "payment": { "status": "AUTHORIZED|INVOICE_REQUESTED" }
}
```

- **202 Accepted** (asynchron, z. B. bei komplexer Disposition)

```
{
  "bookingId": "b_01H8D8...",
  "status": "PENDING",
  "checkAfter": "2025-09-01T10:05:00+02:00"
}
```

### 3.7.4.4 Buchung lesen & stornieren

`GET /v1/bookings/{id}`

**Antwort 200**

```
{
```

```

"bookingId": "b_01H8D8...",
"status":
"ASSIGNED|ARRIVED|IN_PROGRESS|COMPLETED|CANCELLED",
"timeline": [
  { "ts": "2025-09-01T10:05:00+02:00", "event": "CONFIRMED" },
  { "ts": "2025-09-01T10:20:00+02:00", "event": "ASSIGNED", "vehicle": {
"plate": "DO-AB 1234" } }
]
}

```

POST /v1/bookings/{id}/cancel

```
{ "reason": "CUSTOMER_REQUEST", "note": "Termin verschoben" }
```

**Antwort 200** { "status": "CANCELLED" } (Stornoregeln gemäß Quote/Produkt).

### 3.7.4.5 Zahlung

POST /v1/payments (z. B. App-Kauf/Co-Payment)

```

{
  "bookingId": "b_01H8D8...",
  "method": "CARD|DIRECT_DEBIT",
  "amount": 27.00,
  "currency": "EUR",
  "returnUrl": "app://callback/payment"
}

```

**Antwort 201**

```

{
  "paymentId": "pay_01H8D9..",
  "status": "REQUIRES_AUTH",
  "redirectUrl": "https://psp.example/...",
  "expiresAt": "2025-09-01T10:15:00+02:00"
}

```

POST /v1/payments/{id}/capture → 200 { "status": "CAPTURED" }

## 3.7.5 Statusmodell (Buchung)

```

REQUESTED → (PENDING) → CONFIRMED → ASSIGNED → ARRIVED → IN_PROGRESS → COMPLETED
      ↳ FAILED
REQUESTED/CONFIRMED → CANCELLED

```

- **Übergangsregeln:**
  - **PENDING** max. N Minuten → automatisch **FAILED** bei Zeitüberschreitung.
  - **CANCELLED** nur gemäß Produktbedingungen (Gebührenfenster, Cut-Off).
- **Events (Webhooks / Pull):** **BOOKING\_CONFIRMED**, **BOOKING\_ASSIGNED**, **VEHICLE\_ARRIVED**, **TRIP\_STARTED**, **TRIP\_COMPLETED**, **BOOKING\_CANCELLED**, **BOOKING\_FAILED**.

### 3.7.6 Sicherheit & Autorisierung

- **OAuth2/OIDC:**
  - **Kiosk/Server-zu-Server:** Client-Credentials-Flow (gerätespezifischer Client).
  - **Endkunden-App:** Authorization-Code-Flow mit **PKCE**.
- **Scopes (Beispiele):**
  - availability:read, quote:write, booking:write, booking:read, payment:write.
- **Rollen:** ENDUSER, KIOSK\_OPERATOR, MEDICAL\_FACILITY, OPERATOR\_ADMIN.
- **DSGVO/PII:** Minimierung; Vertreterrolle (z. B. Klinik) **pflichtig kennzeichnen** (actingOnBehalf).
- **Transport:** TLS 1.2+; HSTS; JWT Access-Tokens mit kurzer TTL; Refresh-Token nur für App-User.

### 3.7.7 Idempotenz, Nebenläufigkeit, Fehlermanagement

- **Idempotenz:** POST /bookings und POST /payments **müssen** Idempotency-Key akzeptieren.
- **Optimistische Sperre:** Ressourcenzugriffe mit ETag + If-Match (z. B. bei Teil-Updates).
- **Fehlercodes (Auszug):**
  - 400 VALIDATION\_ERROR (z. B. ungültige Adresse),
  - 401/403 AUTHORIZATION\_ERROR (fehlende/ungültige Scopes),
  - 404 NOT\_FOUND (Booking/Quote nicht vorhanden),
  - 409 CONFLICT (Idempotency-Konflikt),
  - 410 GONE (abgelaufene Quote),
  - 422 BUSINESS\_RULE\_VIOLATION (z. B. Fahrzeugtyp nicht verfügbar, Kostenträger ungültig),
  - 429 RATE\_LIMITED,
  - 500 SERVER\_ERROR.
- **Beispiel-Fehlerpayload:**

```
{  
  "type": "https://vspv.example/errors/BUSINESS_RULE_VIOLATION",  
  "code": "KOSTENTRAEGER_INVALID",  
}
```

```

"message": "Angegebener Kostenträger-Code ist unbekannt oder
gesperrt.",
"field": "payer.code",
"correlationId": "8e2a6d1e-..."
}

```

### 3.7.8 Sonderfälle & Pflichtattribute

- **Adressgenauigkeit:** `destinationAddress.location` (Lat/Lon) empfohlen für Routing/ETA.
- **Barrierefreiheit:** `requirements.wheelchairAccessible` zwingend, wenn erforderlich; bei Liegendbeförderung `requirements.lyingTransport=true`.
- **Kostenträger:** `payer.type="KOSTENTRAEGER"` + `code` (vgl. 3.6 KOSTENTRAEGER).
- **Pooling:** Angebot nur, wenn `poolingPossible=true` (aus Availability); zusätzlich `passengers ≤ MAX_KAPAZITAET`.
- **Preismodelle:**
  - **fixed** → Festpreis (Quote verbindlich bis `validUntil`),
  - **meter** → Taxameter-basiert, Anzeige `baseFare/estTotal` möglich, finaler Betrag nach Abschluss,
  - **composed** → Kombi-Tarife (z. B. Anteil ÖPNV + Taxi).

### 3.7.9 Webhooks

**Zweck:** Asynchrone Zustellwege für Statuswechsel (Kiosk/Plattform).

**Registrierung:** pro Mandant ein `targetUrl` + `events[]`.

**Beispiel-Event (BOOKING\_ASSIGNED):**

```

{
  "eventId": "evt_01H8DA...",
  "eventType": "BOOKING_ASSIGNED",
  "occurredAt": "2025-09-01T10:20:00+02:00",
  "data": {
    "bookingId": "b_01H8D8...",
    "vehicle": { "typeRef": "VSPV:VehicleType:Liegemietwagen" },
    "eta": "2025-09-01T10:28:00+02:00"
  },
  "signature": "base64(hmac-sha256(payload, secret))"
}

```

**Sicherheit:** Signaturprüfung (HMAC) + Replay-Schutz (Timestamp/Nonce).

### 3.7.10 Beispielabläufe (konkret)

#### A) Kioskbuchung Klinik → Wohnadresse (Liegend, Kostenträger)

1. /availability (Klinik-Kiosk, Client Credentials) → Optionen.
2. /quotes mit payer.type="KOSTENTRAEGER".
3. /bookings mit actingOnBehalf (Klinikfall-Nr.).
4. Optional /payments (INVOICE/CARD je Produkt).
5. Webhooks BOOKING\_CONFIRMED → ASSIGNED → COMPLETED.

#### B) App-Buchung Adresse → Bahnhof (Festpreis, Karten-Zahlung)

1. /availability (Endkunde, PKCE).
2. /quotes (fixed 12.00 EUR).
3. /bookings (Endkunde).
4. /payments → returnUrl 3-DS → /payments/{id}/capture.
5. GET /bookings/{id} oder Webhooks für Status.

### 3.7.11 Mappingtabelle Kapitel 3.7 (Auszug)

REST-Feld	NeTeX/VDV-462-Referenz	Pflicht	Hinweise
origin.ref	StopPlaceRef	O	Sammelpunkt, z. B. Klinik-Eingang
destinationAddress	Adresse innerhalb FlexibleStopPlace/Bediengebiet	O	Lat/Lon empfohlen
flexibleServiceJourneyRef	FlexibleServiceJourney.id	O	Fahrtenobjekt
vehicleTypeRef	VehicleType.id	O	z. B. Liegemitwagen
pricingModel/fare	FareStructure / keyList(FESTPREIS) / Extensions(vspv:TaxiTariff)	O	Siehe 3.6
payer.code	Key KOSTENTRAEGER / Billing-Profil	Δ	Bei Drittzahler nötig
requirements.*	Fahrzeug-/Service-Attribute	O	Rollstuhl, liegend, Kapazität
status	Buchungs-State (Kap. 3.7.5)	O	REST-seitig normiert

Δ = abhängig vom Produkt/Fall.

### 3.7.12 Konformitätskriterien (prüfbar)

1. **IDs/Zeichen** gemäß 3.3; **Zeit- und Währungsformate** gemäß Vorgaben dieses Kapitels.
2. **Auth/Scopes** passend zur Operation; Auftragsdaten nur im zulässigen Umfang.
3. **Idempotency-Key** bei allen schreibenden **POST** eingesetzt.
4. **Quote-Ablauf** (**validUntil**) durchgesetzt; abgelaufene Quotes → **410 GONE**.
5. **Kostenträgerfälle** nur mit gültigem Code (vgl. 3.6 Registry) → sonst **422**.
6. **Webhook-Signaturen** verifiziert; Events genau einmal verarbeitet (Idempotenz serverseitig).

## 3.8 Echtzeit- und Auskunftsschnittstellen (TRIAS, SDG-Plattformen)

### Zweck und Geltungsbereich

Dieser Abschnitt beschreibt, wie die im VSPV-Standard 101 statisch in NeTeX/VDV 462 modellierten Daten (Netz, Fahrten, Tarife, Buchungsregeln) mit Echtzeit- und Auskunftsschnittstellen zusammenspielen. Im Fokus stehen TRIAS (für Auskunft und Buchungs-/Absprungprozesse) sowie die Bereitstellung gegenüber landesweiten Datendrehscheiben/NAP-Plattformen (z. B. mobidrom.nrw). NeTeX bleibt dabei die Quelle der fachlich-statischen Wahrheit; Echtzeit und Buchung laufen über TRIAS/SIRI-Dienste, die die NeTeX-IDs referenzieren.

### Architekturüberblick

- **Stammdaten & Profile:** NeTeX/VDV 462 (PublicationDelivery mit ServiceFrame/TimetableFrame etc.) als verbindliches Austauschformat; KeyValue/Extensions nur wo zwingend nötig.
- **Echtzeit/Auskunft & Buchung:** TRIAS (insb. v1.4) für Reiseauskunft, Tarifierung und den standardisierten „Absprung“/Buchungsfluss; ergänzend SIRI-Profile für Fahrzeug-/Prognosedaten, wo vorhanden.
- **SDG/NAP-Bereitstellung:** Regelmäßige und inkrementelle Lieferungen an den NAP (z. B. mobidrom.nrw) mit Metadaten und Aktualitätsanforderungen gemäß EU-Vorgaben.

### TRIAS-Integration (Auskunft, Absprung, Buchung)

**Absprung/Buchung (TRIAS 1.4):** Für Gelegenheitsfahrten empfiehlt die SDGV-Projektdokumentation den standardisierten **BookingRequest** als Absprung vom Auskunftssystem in das Buchungssystem. Minimal erforderlich sind u. a. Pick-up/Set-down als GeoPosition + Name, geplante Abfahrts/Ankunftszeit (ServiceDeparture/ServiceArrival) sowie ein Zeitband (EarliestDeparture/LatestArrival). Das Buchungssystem darf hierbei direkt mit einer HTML-Seite (UI der Buchung) antworten; eine TRIAS-BookingResponse ist nicht zwingend.

**FareRequest/FareResponse:** Vor der Buchung kann ein Fahrpreis über TRIAS angefragt werden; Grundlage sind die NeTeX-Tarifobjekte (FareFrame). Dadurch lassen sich z. B. Festpreise/Zuschläge korrekt aus dem VSPV-Modell heraus bepreisen.

**Weitere TRIAS-Operationen:** Storno/Status (CancelBooking/BookingStatus) referenzieren die zuvor vergebene Booking-ID; auch hier verweisen Inhalte auf NeTeX-IDs (Fahrten, Tarifprodukte), sodass nachgelagerte Systeme eindeutig zuordnen können.

**Verbindliche ID-Bezüge:** Alle in TRIAS verwendeten Objekte (Stop/CallLocations, Journey/Trip, FareProducts) müssen auf NeTeX-IDs aus VDV 462 zeigen; so bleibt die Kette „Auskunft → Buchung → Abrechnung“ durchgängig referenzierbar.

### Minimal-Mapping VSPV ↔ TRIAS (Auszug)

VSPV/NeTeX	TRIAS-Feld (Beispielpfad)	Hinweis
FlexibleServiceJourney.id	BookingRequestContent/PublicTransport/.../ServiceRef	Buchungsobjekt (Fahrt) referenzieren.
StopPlace (Origin/Destination)	.../PickUpLocation/CallLocation (GeoPosition/LocationName) / .../SetDownLocation/CallLocation	Geodaten + klarer Name.
Zeitfenster/Vorlauf	MinMaxTimeBandGroup/EarliestDepartureTime / LatestArrivalTime	ISO-Zeitangaben.
FareProduct/FareStructure	FareRequest/FareResponse (FareProduct, Amount, Currency)	Tariflogik aus FareFrame.

### Echtzeitdaten: TRIAS und SIRI

TRIAS deckt Echtzeit-Auskunft (Prognosen, Anschlüsse, Störungen) integrativ ab; SIRI ist eine alternative/ergänzende Echtzeitschnittstelle (z. B. EstimatedTimetable/VehicleMonitoring). Beide ersetzen **nicht** die statische Modellierung in NeTeX, sondern referenzieren sie.

### SDG/NAP-Plattformen (Bereitstellung & Aktualität)

Landesweite Plattformen/NAP (z. B. mobidrom.nrw) erwarten harmonisierte, aktuelle Datenlieferungen: NeTeX für Stammdaten, TRIAS/SIRI für dynamische Inhalte; zudem Metadaten/Verfügbarkeitsangaben und Aktualität gemäß EU-Vorgaben. Diese Anforderungen (inkl. Echtzeit-Aktualisierung) sind in den SDGV-Unterlagen adressiert.

### Konformitätsregeln und Fehlerbehandlung

1. **Referenzkonsistenz:** Jede TRIAS-Nachricht muss auf existierende NeTeX-IDs zeigen (Fahrten, Haltepunkte, FareProducts). Ansonsten Import ablehnen und Fehler mit betroffener Referenz protokollieren.

2. **Zeitangaben:** ServiceDeparture/ServiceArrival und EarliestDeparture/LatestArrival müssen logisch konsistent sein (Zeitband einschließlich). Validierung im Booking-Gateway.
3. **Tarif-Kohärenz:** FareResponse/BookingConfirmation dürfen nur FareProducts referenzieren, die im NeTEx-FareFrame vorhanden sind; PaymentMethod-Codes müssen deckungsgleich sein.
4. **Liefermodell:** Statische Daten ausschließlich als PublicationDelivery mit Frames; inkrementelle Updates zulässig, Version/Gültigkeit pflegen.
5. **NAP-Bereitstellung:** Publikationsrhythmus und Metadaten so wählen, dass NAP-Aktualitätsanforderungen erfüllt werden.

### Praxisbeispiel „Absprung“ aus der Auskunft in die Buchung

Ein Fahrgast wählt in der ÖPNV-Auskunft eine Gelegenheitsfahrt. Das Auskunftssystem ruft die in den Solldaten hinterlegte Buchungs-URL auf und übergibt einen **TRIAS BookingRequest** mit mindestens: PickUp GeoPosition/Name, SetDown GeoPosition/Name, ServiceDeparture/ServiceArrival sowie EarliestDeparture/LatestArrival (Zeitband). Das Buchungssystem kann direkt mit der HTML-Buchungsseite antworten (statt BookingResponse). Damit ist der Übergang medienbruchfrei, und alle Referenzen (z. B. Journey/StopPlace) bleiben konsistent.

## 3.9 Versionierung, Validierung und Qualitätssicherung

### Ziel und Geltungsbereich

Dieser Abschnitt definiert verbindliche Regeln und Verfahren zur Versionierung, Gültigkeitssteuerung, Validierung und laufenden Qualitätssicherung aller VSPV-Datenlieferungen. Die Vorgaben orientieren sich am NeTEx-/VDV-462-Profil (u. a. *PublicationDelivery*, Frame-Modell, Objektversionen) und präzisieren deren Anwendung für Gelegenheitsverkehre im VSPV-Standard 101.

#### 3.9.1 Versionierung von Objekten

- **Schlüsselattribute:** Jedes versionierbare Objekt besitzt @id und @version als internen Schlüssel. Referenzen müssen stets **ref und version** des Zielobjekts führen; fehlt die Version, kann die Schema-Validierung Bezüge nicht prüfen.
- **Mehrere Versionen je ID:** Es kann beliebig viele Instanzen gleicher id mit unterschiedlicher version geben (Versionierung auch für Stammdaten wie StopPlaces/Tarifzonen zulässig).
- **Nicht-versionierbare Klassen:** Für nicht versionierbare Objekte ist im Attribut version der konstante Wert any zu setzen.
- **Stabile IDs:** IDs sind persistent; Änderungen am Objekt erzeugen **neue Versionen**, nicht neue IDs (vgl. 3.2/3.3-Konventionen). Eine zentrale ID-Registry wird empfohlen, um Kollisionen zu vermeiden.

## Beispiel

```
<StopPlace id="VSPV:StopPlace:NRW-1001"
version="1.0">...</StopPlace>
<StopPlace id="VSPV:StopPlace:NRW-1001"
version="2.0">...</StopPlace>
```

### 3.9.2 Gültigkeitszeiträume

- **Ort der Gültigkeit:** Im VDV-NeTeX-Profil werden Gültigkeiten als **ValidBetween/FromDate/ToDate** geführt; pro Element ist **maximal ein** Zeitraum zulässig.
- **Frames und Disjunktheit:** Gültigkeit auf *CompositeFrame*, *ServiceFrame*, *TimeTableFrame*, *VehicleScheduleFrame*; Frames gleicher Art innerhalb eines *CompositeFrame* müssen **disjunkt** und jeweils **in sich konsistent und vollständig** sein.

## Beispiel

```
<ServiceFrame id="VSPV:ServiceFrame:2025-01" version="1.0">
  <ValidBetween><FromDate>2025-01-01</FromDate><ToDate>2025-03-
31</ToDate></ValidBetween>
</ServiceFrame>
```

### 3.9.3 Liefermodelle (Initial- und Delta-Updates)

- **Verpackung:** Jede Lieferung ist in **PublicationDelivery** zu kapseln.
- **Initiale Komplettlieferung:** Vollständige Frames für einen Stichtag/Zeitraum.
- **Inkrementelle Updates:** Übermittlung ausschließlich **neuer/geänderter** Objekte (gleiche **id**, höhere **version**) und ggf. **Enddatierung** auslaufender Gültigkeiten – geeignet zur schlanken, nachvollziehbaren Pflege ohne Vollabzug.
- **Abnahmekriterien:** Delta-Update darf keine referenzlosen Objekte erzeugen; alle **ref/version** müssen in der Zielbasis auflösbar sein. (Siehe Validierung unten.)

### 3.9.4 Technische Validierung (Schema)

- **XSD-Konformität:** Pflichtprüfung gegen das referenzierte NeTeX/VDV-462-Schema (Struktur, Datentypen, Kardinalitäten). Nichtkonforme Lieferungen werden abgewiesen.
- **Grundprüfungen** (Auszug):
  - Vorhandensein von **@id/@version** an relevanten Klassen; **ref/version** an Referenzen.
  - Frame-Struktur, *PublicationDelivery*-Hülle.

- Feldlängen/Zeichensatz nach 3.3 (z. B. Name ≤ 70, UTF-8; IDs ASCII-Subset).

### 3.9.5 Semantische Validierung

- **Schematron/SHACL-Regeln** für projektspezifische Semantik, die XSD nicht abbildet (z. B. Listen zulässiger Werte, Querbezüge, Geschäftsregeln).
- **Kernregeln** (Auszug):
  - **Referenzintegrität:** Jede ...Ref zeigt auf existierendes Objekt gleicher id/version.
  - **Eindeutigkeit:** Keine doppelten id+version; IDs global eindeutig (vgl. 3.2/3.3).
  - **Mehrsprachigkeit:** ISO-639-1 Sprachcodes; Pflichtsprache **de** vorhanden; Längen/Zeichen gültig.
  - **Profilkonformität Erweiterungen:** keyList-Schlüssel aus dokumentierter Liste; **Extensions** nur, wenn im Standardfeld/Key nicht darstellbar (Implementationshinweis VDV 462).
  - **Interchange/Umsteigezeiten** (falls verwendet): Dauerfelder im ISO-8601-Format, Wertebereiche plausibel.

### 3.9.6 Qualitätssicherungsprozess (von der Erstellung bis Go-Live)

1. **Erzeugung & lokale Vorvalidierung** beim Datenlieferanten (XSD → Schematron/SHACL → interne Geschäftsregeln). Bereitgestellte Beispiel-XMLs/Mappingtabellen sind Referenz.
2. **Eingangsprüfung** im Zielsystem (Schema-Check, Profil-Check, ID-Kollisionen, Referenzen). Bei Fehlern: Abweisung mit Protokoll.
3. **Staging/Abnahme:** Testimport, differenzierte Diff-Analyse (alt vs. neu), stichtagsbezogene Stichproben (z. B. Frame-Gültigkeiten lückenlos/disjunkt).
4. **Produktivnahme:** Nach Freigabe; Monitoring der Schnittstellen (Fehlerraten, Aktualität, Objektzuwachs), regelmäßige Re-Validierung geplanter Lieferzyklen.

#### Fehlerprotokoll – Beispiele (semantische Ebene)

- „ID **VSPV:StopPlace:NRW-1001** doppelt geliefert (Version 1.0 bereits vorhanden).“
- „Referenz **FareProductRef=VSPV:Fare:XYZ** nicht auflösbar.“
- „Name/de fehlt (Pflichtsprache).“
- „ServiceJourneyRef im Komponenten-Tarif zeigt nicht in gelieferten Frames.“

### 3.9.7 Artefakte für Validierung & QS

- **Mappingtabellen** (maßgeblich für technische Umsetzung; Pfade, Kardinalitäten, Pflichtfelder).
- **Beispiel-XMLs** (validierte Musterfälle für typische Use-Cases).
- **Profilierte PrüfregeIn** (Schematron/SHACL) als Ergänzung zum XSD.

### 3.9.8 Konformität und Änderungsdisziplin

- **Konformität zum VDV-Profil** ist herzustellen (u. a. Frame-Modell, Gültigkeitsmodell, *PublicationDelivery*-Pflicht). Abweichungen/Proprietäres sind explizit zu kennzeichnen und zu dokumentieren.
- **Änderungsmanagement:** Jede inhaltliche Änderung → neue version; Abkündigungen über ToDate. Delta-Lieferungen müssen diese Änderungen explizit transportieren.

### 3.9.9 Beispiel: Delta-Lieferung (Versionserhöhung & Enddatierung)

```
<PublicationDelivery>
  <ServiceFrame id="VSPV:ServiceFrame:2025-01" version="1.1">
    <StopPlace id="VSPV:StopPlace:NRW-1001" version="2.0">
      <ValidBetween><FromDate>2026-01-01</FromDate></ValidBetween>
    </StopPlace>
    <StopPlace id="VSPV:StopPlace:NRW-2002" version="1.0">
      <ValidBetween><FromDate>2025-01-01</FromDate><ToDate>2025-
12-31</ToDate></ValidBetween>
    </StopPlace>
  </ServiceFrame>
</PublicationDelivery>
```

(Neue Geometrie/Bezeichnung für NRW-1001 mit version="2.0", gleichzeitige Enddatierung NRW-2002.)

## 4 Schnittstellen zu ÖPNV-Auskunfts- und Buchungssystemen (Integrationsstufen SDGV)

### 4.1 Statischer Datenaustausch (Integrationsstufe 1)

#### Zweck und Geltungsbereich

Integrationsstufe 1 dient der **reinen Fahrgastinformation**: Bereitstellung von Grunddaten zu Netz/Topologie (Haltestellen, Bediengebiete), Angebot (Linien/Fahrten inkl. flexibler Bedienformen) und – wo nötig – tariflichen Hinweisen, **ohne** Buchungsfunktionen. Technische Basis ist NeTeX gemäß VDV-Schrift 462; Lieferung stets als **PublicationDelivery** mit inhaltlichen Frames, organisiert in **CompositeFrame** mit Gültigkeitszeiträumen. Frames derselben Art müssen in einem **CompositeFrame** **disjunkt** sein; jede Lieferung ist in sich konsistent und vollständig.

Die SDGV-Projektdokumentation ordnet diese Stufe als „**reiner Datenaustausch (Information)**“ im dreistufigen Modell ein.

### **Datenpakete (NeTEx/VDV-462)**

Die Lieferung umfasst mindestens folgende Frames; die Zuordnung folgt der im VSPV-Standard 101 verwendeten Frame-Systematik.

- **ResourceFrame** – Betreiber/Organisationen (Name, Kurzname, Kontakt). Für FGI-Profile sind ID, Version, Name obligatorisch; weitere Felder je Profilstufe (L1–L3).
- **NetworkFrame** – Haltestellen/StopPlaces und **FlexibleStopPlace** (Bediengebiete als Polygon, Sammelpunkte mit Centroid).
- **ServiceFrame** – Linien/Services inkl. **Flexible...**-Objekte für Adressbedienung (z. B. **FlexibleServiceJourney** / **FlexibleLine**) und zugehörige Regeln auf Datenebene.
- **TimetableFrame** – Betriebskalender/OperatingDays, ggf. Umsteige-Regeln (z. B. **StandardTransferTime**, **MaximumWaitTime**) zur Auskunft.
- **FareFrame (optional)** – nur, wenn für die Auskunft **statische** Tarifhinweise nötig sind (z. B. Produkthinweis, Kombitarif ohne Preisausrechnung).

Hinweis: VDV-462 verlangt die Verpackung aller Inhalte unter **PublicationDelivery**; **CompositeFrame** bündelt die inhaltlichen Frames und trägt den Gültigkeitszeitraum (**ValidBetween**).

### **Muss-/Soll-Umfang (Integrationsstufe 1)**

#### **MUSS**

- **ResourceFrame/Operator** (Betreiberstammdaten).
- **NetworkFrame/StopPlace** **und/oder** **FlexibleStopPlace** (Sammelpunkte, Bediengebiete als Polygon).
- **ServiceFrame** mit Angebotsobjekten (klassische **ServiceJourney** **oder** flexible **FlexibleServiceJourney/FlexibleLine**, je Betriebsform).
- **TimetableFrame** mit Kalenderelementen (mindestens Tagesarten/Betriebszeiträume).
- Verpackung als **PublicationDelivery** mit **CompositeFrame** (disjunkte inhaltliche Frames).

#### **SOLL**

- Umsteige-Regeln (**InterchangeRule...**) für verlässliche Anschlussdarstellung.
- **FareFrame** für lesbare Tarifhinweise (ohne Berechnung).

#### **KANN**

- Zusatzattribute per **keyList** (z. B. Bedienart „Adresse/Sammelpunkt“) – bevorzugt **nur**, wenn Standardstrukturen es nicht abbilden.

### Modellierungsleitplanken (Konsistenz mit Kap. 3)

- **IDs/Versionen:** stabile @id, Änderungen als neue @version; Gültigkeiten über ValidBetween.
- **Flexible Bedienung:** Bedienegebiete als FlexibleStopPlace-Polygon; Sammelpunkte als StopPlace mit Centroid; Adressbedienung in Flexible...-Objekten.
- **Sprachen/Feldgrenzen:** Mehrsprachige Namen per MultilingualString (ISO 639-1); Felder und Zeichensätze wie in Kap. 3.3 festgelegt.

### Minimalbeispiel (schematisiert, gekürzt)

```
<PublicationDelivery>
  <dataObjects>
    <CompositeFrame id="VSPV:CF:2025-01" version="1.0">
      <ValidBetween><FromDate>2025-01-01</FromDate></ValidBetween>
      <frames>
        <ResourceFrame id="VSPV:RF:org" version="1.0">
          <organisations>
            <Operator id="VSPV:Op:123" version="1.0">
              <Name><Text language="de">Taxi & ÖPNV
Westfalen</Text></Name>
            </Operator>
          </organisations>
        </ResourceFrame>
        <NetworkFrame id="VSPV:NF:netz" version="1.0">
          <stopPlaces>
            <StopPlace id="VSPV:StopPlace:Klinikum-Hamm" version="1.0">
              <Name><Text language="de">Sammelpunkt Klinikum
Hamm</Text></Name>
              <Centroid>...</Centroid>
            </StopPlace>
            <FlexibleStopPlace id="VSPV:Bedienegebiet:Hamm-Innenstadt"
version="1.0">
              <Polygon>...</Polygon>
            </FlexibleStopPlace>
          </stopPlaces>
        </NetworkFrame>
        <ServiceFrame id="VSPV:SF:angebot" version="1.0">
          <!-- klassisch oder flexibel je nach Betriebsform -->
        </ServiceFrame>
        <TimetableFrame id="VSPV:TT:kalender" version="1.0">
          <!-- OperatingDays/ServiceCalendar, optional InterchangeRules -->
        </TimetableFrame>
      </frames>
    </CompositeFrame>
  </dataObjects>
</PublicationDelivery>
```

```

</CompositeFrame>
</dataObjects>
</PublicationDelivery>

```

Bezug der Objektarten und Beispiele siehe Kap. 3/5 des VSPV-101-Entwurfs (FlexibleStopPlace/Sammelpunkte, FlexibleServiceJourney).

### Aktualisierung & Gültigkeiten

- Versionierte Nachlieferungen als neue **CompositeFrame**-Versionen mit **lückenlos** abgedeckten Zeiträumen; parallele Fahrplanversionen in einer Lieferung sind zulässig.
- Änderungen an Betreiber-/Haltestellstammdaten über neue Objektversionen; historische Stände bleiben auskunftsfähig.

### Qualitätssicherung (Stufe 1)

- **Profilkonformität VDV-462** (FGI-Profile L1–L3 je Projektvereinbarung), inkl. Pflichtfelder/ Kardinalitäten.
- **Sparsame Erweiterungen:** zuerst Standardobjekte nutzen; proprietäre **Extensions** nur, wenn fachlich nötig und abgestimmt.
- **Stufenmodell-Konformität:** Keine Buchungsoperationen; die SDGV-Stufe 1 liefert ausschließlich Informationsdaten.

### Mapping (Auszug) – fachlich → technisch (Stufe 1)

Fachliche Info	NeTEx/Frame	Element/Beispiel	Konvention
Betreiber	ResourceFrame	Operator/Name, ShortName	Pflichtfelder gem. FGI-Profil.
Sammelpunkt	NetworkFrame	StopPlace mit Centroid	Mehrsprachige Namen möglich.
Bediengebiet	NetworkFrame	FlexibleStopPlace mit Polygon	WGS84-Koordinaten; ID stabil.
Flexibles Angebot	ServiceFrame	FlexibleServiceJourney/FlexibleLine	Nur Informationszweck in Stufe 1.
Kalenderegeln	TimetableFrame	OperatingDays/Calendar	Für Auskunft ausreichend granular.

Damit ist Integrationsstufe 1 klar umrissen: **NeTEx/VDV-462-konforme** Lieferung von Netz-, Angebots- und (optional) Tarifhinweisen als PublicationDelivery, mit disjunkten, versionierten Frames und minimalen Pflichtinhalten für eine robuste Fahrgastinformation.

## 4.2 Buchungslinks (Integrationsstufe 2)

**Ziel der Stufe 2** ist der medienbruchfreie „Absprung“ aus der Auskunft in ein externes Buchungssystem, ohne dass der Fahrgast Daten erneut eingeben muss. In den SDGV-Unterlagen ist dazu ausdrücklich gefordert, Basis-URL, Fahrt-Identifikation und zu übergebende Parameter zu standardisieren. Aktuelle Bestandsanalysen (u. a. IVU.pool/HAFAS, DIVA/EFA) zeigen funktionierende, aber proprietäre Lösungen – z. B. eine hinterlegte Buchungs-URL inkl. **Rekonstruktionskontext** bei HAFAS bzw. fest konfigurierte URL-Parameter bei EFA. Austausch über Solldaten ist damit (noch) nicht möglich und soll künftig durch einheitliche Festlegungen ersetzt werden.

### 4.2.1 Modellierungsprinzip in VDV 462/NeTEx

Buchungslinks werden **am buchbaren Angebot** hinterlegt, vorzugsweise an **FlexibleServiceJourney** (konkrete bedarfsbasierte Fahrt) oder – falls nur auf Angebots-Ebene – an **FlexibleLine** bzw. am zugeordneten **BookingProcess**. Technisch erfolgt dies konform VDV 462 über **Extensions** (strukturierte Felder) oder eine **keyList** (einfache Key/Value-Paare). Beispiele und Empfehlungen zur klaren Schlüsselbenennung sind im VSPV-Standard 101 bereits verankert.

#### **Beispiel (NeTEx/VDV 462, Extensions):**

```
<FlexibleServiceJourney id="VSPV:ServiceJourney:Taxi-4711"
version="1.0">
  <Extensions>

  <vspv:BookingLinkTemplate>https://booking.partner.tld/deeplink</vspv:Bo
okingLinkTemplate>
  <vspv:BookingLinkMode>GET</vspv:BookingLinkMode>
  <vspv:BookingLinkParamSpec>
    <vspv:param name="jRef" required="true"
source="ServiceJourney.id"/>
    <vspv:param name="oRef" required="true"
source="Origin.StopPlaceRef"/>
    <vspv:param name="dRef" required="true"
source="Destination.StopPlaceRef"/>
    <vspv:param name="dep" required="true"
source="PlannedDepartureTime"/>
    <vspv:param name="pax" required="true"
source="RequestedPaxCount"/>
    <vspv:param name="ctx" required="false"
source="ReconstructionContext"/>
  </vspv:BookingLinkParamSpec>
</Extensions>
</FlexibleServiceJourney>
```

## 4.2.2 Einheitliches URL-Schema (normativ)

**Basis-URL (registriert):** <https://booking.partner.tld/deeplink>

**HTTP-Methode:** **GET** (sichtbarer Link) oder **POST** (Form-Post aus App/Kiosk)

**Zeichensätze/Längen:** gemäß Kap. 3.3 (IDs ASCII ohne Leerzeichen; Texte UTF-8; URLs RFC-konform). (Siehe VSPV-101 Feldregeln.)

**Pflichtparameter (Vorgaben aus SDGV „Buchungslink“ abgeleitet):**

Parameter	Bedeutung	Quelle im Datenmodell
jRef	zu buchende Fahrt (ID)	FlexibleServiceJourney/@id
oRef	Start-Objekt	StopPointInJourneyPattern[1]/StopPointRef
dRef	Ziel-Objekt	letztes StopPointRef
dep	Abfahrtszeit	ISO-8601, aus Fahrtauskunft (oder Buchungszeitfenster)
pax	Personenanzahl	Buchungskontext/Benutzereingabe

**Optionale Parameter (empfohlen):**

- **ctx** – **Rekonstruktions-Kontext** (opaque Token); ermöglicht, dass das Buchungssystem die Detaildaten server-seitig nachlädt (HAFAS-Prinzip).
- **needs** – Bedarfscodes (z. B. wheelchair, stretcher), abgeleitet aus Fahrzeug-/Dienstmerkmalen. (Schlüssel/Flags sind in den VSPV-Beispielen etabliert.)
- **kb** – **Kanal/Kiosk-Bezug** (z. B. kb=VSPV:StopPlace:Klinikum-Hamm), wenn Vermittler bucht. (KioskAccess/BookingProcess im Modell vorhanden.)
- **ret** – optionale Rücksprung-URL.
- **sig, ts, nonce** – Signatur, Zeitstempel, Nonce (siehe Sicherheit).

**Beispiel-Link (GET):**

```
https://booking.partner.tld/deeplink
?jRef=VSPV:ServiceJourney:Taxi-4711
&oRef=VSPV:StopPlace:Klinikum-Hamm
&dRef=VSPV:StopPlace:Hamm-Innenstadt
&dep=2025-03-01T10:30:00+01:00
&pax=1
&ctx=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9... (opaque)
&sig=5f1d...&ts=1740821400&nonce=2b7c...
```

**ctx** repräsentiert den in den Bestandsystemen verwendeten Rekonstruktions-Kontext; die Standardisierung hebt ihn aus der proprietären Ecke ins Profil.

## 4.2.3 Sicherheit & Datenschutz (Stufe 2)

- **Signatur:** sig = HMAC-SHA256(secret, canonical\_querystring); zusätzlich ts (Unix-Zeit, max. Skew ±5 min) und nonce (Einmalverwendung).

- **Gültigkeit:** Links sollten kurzlebig sein (z. B. 10–15 Min.) und server-seitig gegenverifiziert werden.
- **Kontext-Abruf:** Wenn `ctx` verwendet wird, darf das Buchungssystem die Details nur über einen authentifizierten Backend-Call (z. B. per registriertem API-Key) nachladen.
- **SSO-Kompatibilität:** Stufe 2 **erfordert** kein SSO; Vorgaben hierzu folgen in 4.3. (Die SDGV-Doku diskutiert SSO primär für die Tiefenintegration.)

#### 4.2.4 Platzierung im Datenmodell

- **Pro Fahrt:** `FlexibleServiceJourney/Extensions/vspv:BookingLink...` – wenn der Link exakt eine konkrete Verbindung abbildet.
- **Pro Angebot:** `FlexibleLine/Extensions/vspv:BookingLinkTemplate` – wenn der Anbieter einen generischen Deeplink bereitstellt, der mit Parametern befüllt wird.
- **Über `BookingProcess`:** Hinterlegung der Basis-URL und unterstützter Methoden/Kanäle zur Wiederverwendung in mehreren Fahrten. (*BookingProcess ist im VSPV-Modell etabliert.*)
- **Einfache Zusatzangaben:** als `keyList` (z. B. `BOOKING_LINK_TEMPLATE`, `BOOKING_PROVIDER_REF`).

#### 4.2.5 Kompatibilität zu Bestandswelten

- **HAFAS/IVU.pool:** Der bekannte **Rekonstruktions-Kontext** wird als `ctx` standardisiert; die Basis-URL wandert aus der proprietären BL-Zeile in das NeTEx-Profil (Solldaten).
- **EFA/DIVA:** Statt hartcodierter Parameter im Backend werden die Link-Parameter **aus Solldaten** befüllt; so wird Austauschbarkeit erreicht.
- **TRIAS als Zwischenstufe?** SDGV diskutiert TRIAS auch für Stufe 2 (z. B. Übergabe identifizierender Codes statt freier URLs); das Profil bleibt anschlussfähig, ohne TRIAS zu erzwingen.

#### 4.2.6 Validierung (ergänzend zu Kap. 3.3)

- **Pflichtfelder vorhanden:** `jRef`, `oRef`, `dRef`, `dep`, `pax`.
- **Formate korrekt:** ISO-8601-Zeit, IDs regelkonform, URL RFC-konform. (*siehe Feldregeln/Mappings Kap. 3.2/3.3*)
- **Link prüfbar:** Signatur verifizierbar; `ts/nonce` gültig; `ctx` (falls vorhanden) auflösbar.
- **Semantik:** `jRef` muss auf existierende `FlexibleServiceJourney` zeigen; `oRef/dRef` zum dazugehörigen `JourneyPattern` passen.

- **Konfig-Konsistenz:** Falls am Angebot **sowohl** `BookingProcess` (mit `online`) **als auch** ein Deeplink hinterlegt ist, müssen Basis-URL und Template zusammenpassen (sonst Fehler).

#### 4.2.7 Praxisbeispiel (Stufe 2, Kiosk/App)

**Use-Case:** Auskunft zeigt eine Taxi-Teilstrecke; per Klick wird die Buchungsseite des Anbieters geöffnet, die Daten sind bereits vorbelegt.

- **Datenlage:** `FlexibleServiceJourney` inkl. `Extensions/vspv:BookingLinkTemplate` und Param-Spezifikation (siehe oben).
- **Auskunft → Link-Aufruf:** Die App setzt `jRef`, `oRef`, `dRef`, `dep`, `pax`, optional `ctx` (vom Auskunftssystem erzeugt) und signiert die Anfrage. (*HAFAS-Kompatibilität via `ctx`*)
- **Buchungsseite:** liest Parameter ein, zeigt Fahrtetails mit bereits berechnetem Preis (Tarifinfos stammen aus `FareFrame`; Stufe 2 kann Preise anzeigen, Zahlung erfolgt i. d. R. im externen System).

#### 4.2.8 Mappingtabelle

Fachliche Anforderung	Technische Datenstruktur	Beispiel/Regel
Deeplink je Fahrt	<code>FlexibleServiceJourney/Extensions/vspv:BookingLin</code> k...	Platzierung gemäß VDV 462-Extensions.
Generischer Deeplink je Angebot	<code>FlexibleLine/Extensions</code> <b>oder</b> <code>BookingProcess</code>	<code>vspv:BookingLinkTemplate</code> , Param-Spezifikation.
Rekonstruktions-Kontext	Query-Param <code>ctx</code> ( <code>opaque</code> )	Standardisiert aus HAFAS-Praxis.
Standard-Parameter	<code>jRef,oRef,dRef,dep,pax</code>	Pflicht; Formate siehe Kap. 3.3.
Sicherheit	<code>sig,ts,nonce</code>	Kurzlebige Links, HMAC-Signatur. (Profilvorgabe)
TRIAS-Anschluss	optional Codes statt URL-Freitext	Diskutiert für Stufe 2; verbindlich in Stufe 3.

Stufe 2 definiert **austauschbare, signierte Deeplinks** mit klaren Pflichtparametern und sauberer Verankerung im **NeTeX/VDV 462-Profil** (`Extensions/keyList`). Das adressiert die in SDGV dokumentierten Lücken (Basis-URL, Parameter, Austauschbarkeit) und bleibt kompatibel zu Bestandswelten (HAFAS/EFA), ohne bereits die Tiefenintegration zu verlangen.

## 4.3 Tiefenintegration (Integrationsstufe 3: TRIAS-Flows & REST-Bridge, SSO-Hinweis → Kap. 7)

### Zweck und Abgrenzung

Integrationsstufe 3 ermöglicht **vollständige Buchung innerhalb des ÖPNV-Auskunfts- bzw. Vertriebsfrontends** – ohne Medienbruch und ohne Wechsel in fremde Oberflächen. Technisch wird dies durch

1. ein **TRIAS-Profil** für *Verfügbarkeit, Angebotsbereitstellung/Quote, Buchung, Änderung, Stornierung, Status* sowie
2. eine **REST-Bridge** zur VSPV-Buchungs-API (Kap. 3.7) realisiert. **SSO** (Kap. 7) sorgt optional dafür, dass der Fahrgast ohne erneute Anmeldung durchgängig authentifiziert bleibt.

### 4.3.1 Zielbild & Architektur

#### Akteure/Schichten

- *Frontend* (Auskunft/Shop): UI, ruft TRIAS-Services auf.
- *TRIAS-Gateway* (Integrationsschicht): setzt TRIAS-Nachrichten auf **REST-Bridge** um.
- *VSPV-Buchungskern* (Kap. 3.7): REST-API `/availability`, `/quotes`, `/bookings`, `/payments`, Webhooks `/events`.
- *Anbietersysteme* (Disposition/Operator): erhalten bestätigte Buchungen, liefern Status/Ereignisse.

#### Prinzip

- **MUSS**: Jeder Frontend-Schritt (Verfügbarkeit→Angebot→Buchung→Status) ist als TRIAS-Operation definiert **und** 1:1 auf REST abbildbar.
- **SOLL**: Alle Vorgänge sind idempotent; Zeitouts, Wiederholungen und Duplikat-Erkennung sind spezifiziert.
- **KANN**: SSO-gestützte Kundenbindung (Token-Durchreichung, profilbasierte Zahlung).

### 4.3.2 Vorgangstypen (Pflichtenheft)

Vorgang	MUSS-Felder (aus Endnutzer-Sicht)	Ergebnis
<b>Verfügbarkeit</b>	Start/Ziel, Zeit(raum), Pax, Bedarfe (z. B. wheelchair, stretcher)	Liste <b>buchbarer Optionen</b> (mit serviceJourneyRef/flexibleServiceJourneyRef)

Vorgang	MUSS-Felder (aus Endnutzer-Sicht)	Ergebnis
<b>Angebot/Quote</b>	gewählte Option + Kontext (Preisrelevantes)	<b>Preis</b> , Konditionen, Reservierungsfenster, Quote-ID
<b>Buchung</b>	Quote-ID <b>oder</b> Parameter-Set; Kontakt-/DSGVO- Zustimmung; Zahlart	<b>Buchungs-ID</b> , Zahlungsstatus, ETA/Abholinfo
<b>Änderung</b>	Buchungs-ID, Änderung (Zeit, Ziel, Pax, Bedarf)	neue Quote → Bestätigung/Abbruch
<b>Stornierung</b>	Buchungs-ID, Grund (optional)	Stornobestätigung, Gebührenhinweis
<b>Status/Events</b>	Buchungs-ID	Zustandsautomat: CREATED → CONFIRMED → ASSIGNED → PICKUP → IN_PROGRESS → COMPLETED (oder CANCELLED/FAILED)

#### 4.3.3 TRIAS-Profil (Anfrage/Antwort – konzeptionell)

Hinweis: Die folgenden Bezeichner sind **Profilnamen**. Sie beschreiben **den semantischen Inhalt**, nicht zwingend die originalen TRIAS-Elementnamen. Die konkrete XML-Serialisierung wird im Profilanhang normiert.

##### A) AvailabilityRequest / AvailabilityDelivery

- **Input:** OriginRef, DestinationRef, When (Zeitpunkt/fenster), Pax, Needs[] (z. B. wheelchair, lyingTransport, assistanceRequired).
- **Output:** Liste AvailabilityOption mit
  - ServiceRef (FlexibleServiceJourneyRef/ServiceJourneyRef),
  - TimeWindow/PlannedTimes,
  - PriceIndicator (falls ermittelbar),
  - BookingProcessRef, PolicyHints (Lead-Time, Storno-Regeln).

##### B) OfferRequest / OfferDelivery (Quote)

- **Input:** ServiceRef **oder** zuvor erhaltene AvailabilityOptionId, plus konkrete Parameter (Zeit, Pax, Needs).
- **Output:** Offer mit Quoteld, TotalPrice, Currency, Conditions[], OfferValidUntil.

##### C) BookingRequest / BookingDelivery

- **Input:** Quoteld oder Angebotsparameter, Customer (minimal), PaymentMode (CARD|MOBILE|INVOICE|CARRIER), **DSGVO-Consents**.
- **Output:** BookingId, Status=CONFIRMED|PENDING, Zahlungsinfo (z. B. Token/Redirect, falls 3-DS).

#### D) CancelRequest / CancelDelivery

- **Input:** BookingId, Reason (optional).
- **Output:** Bestätigung, Gebührenhinweis (CancellationFee), Zeitstempel.

#### E) BookingStatusRequest / BookingStatusDelivery

- **Input:** BookingId.
- **Output:** Status + Fahrzeug/ETA, ggf. VehiclePlate/VehicleType (sofern zulässig).

### 4.3.4 REST-Bridge (Mapping auf Kap. 3.7)

Die TRIAS-Gateway-Schicht ruft folgende **kanonische REST-Endpunkte** des VSPV-Buchungskerns auf:

TRIAS-Operation	REST-Bridge	Minimal-Payload (JSON)	Antwort
AvailabilityRequest	POST /v1/availability	{ origin, destination, [[ optionId, serviceRef, when, pax, needs[] ] }	{ time, priceHint }
OfferRequest	POST /v1/quotes	{ optionId }	{ serviceRef, time, pax, needs, payer?, channel }
BookingRequest	POST /v1/bookings	{ quoteld, customer, payment, consents, channel }	{ bookingId, status, paymentAction?, eta? }
CancelRequest	POST /v1/bookings/{id}:cancel	{ reason? }	{ bookingId, status }
BookingStatusRequest	GET /v1/bookings/{id}	—	{ bookingId, status, vehicle?, eta? }

#### Kanal/Identitäten

- Header X-Booking-Channel: AUSKUNFT\_FRONTEND (für Stufe 3).
- OIDC-Token (falls SSO aktiv) → Kundenbindung, Zahlarten-Wallet (Kap. 7).

#### Idempotenz

- Jeder **schreibende** Call MUSS Idempotency-Key tragen; Server antwortet bei Wiederholung **identisch** (HTTP 409 nur bei Konfliktparametern).

## Webhooks/Events

- Der Buchungskern sendet **asynchron** Zustandsänderungen an registrierte Endpunkte (POST /events des Gateways).
- Ereignisse: `BOOKING_CONFIRMED`, `ASSIGNED`, `PICKUP_IMMINENT`, `TRIP_STARTED`, `TRIP_COMPLETED`, `CANCELLED`, `FAILED`.

### 4.3.5 Daten und Datenschutz (Auszug)

- **Minimalprinzip:** In Stufe 3 werden **nur** für die Durchführung notwendige Daten übertragen (Kontaktmittel, anonymisierte Kunden-ID).
- **Bedarfe** als **kodierte Flags** (keine Freitexte zu Gesundheitsdaten).
- **PCI-Out-of-Scope** für Auskunftsfreigabe, sofern Bezahlung **server-seitig** bzw. via Token/Redirect an PSP erfolgt.
- **Audit:** Jede Buchung/Änderung mit `correlationId` protokollieren.

### 4.3.6 Sicherheit (Kurzprofil; Details in Kap. 8 & Kap. 7)

- **Transport:** TLS 1.2+, HSTS.
- **AuthN/AuthZ:** OAuth 2.1/OIDC; Scopes für Stufe 3 mind. `availability:read`, `quote:write`, `booking:write`, `booking:read`, `events:read`.
- **SSO-Hinweis (Kap. 7):** Frontend erhält **OIDC-ID-Token** des Fahrgasts; TRIAS-Gateway tauscht gegen Backend-Access-Token (Token-Exchange) → REST-Bridge.
- **Signaturen:** Webhook-Signierung (HMAC); `ts/nonce` gegen Replay.
- **Raten & DoS:** Rate-Limiter, Backoff, Circuit-Breaker.

### 4.3.7 Fehler- & Zeitverhalten

#### Zeiten

- `availability` ≤ 1 s (P95), `quote` ≤ 2 s (P95), `booking` ≤ 5 s (P95) – Richtwerte.
- `OfferValidUntil` MUSS als ISO-Zeitpunkt geliefert werden.

#### Fehlercodes (REST)

- `400` fehlende/ungültige Felder, `401/403` Auth/Scope, `404` Ref unbekannt, `409` Idempotenz-Konflikt, `422` fachliche Regel verletzt (z. B. Lead-Time), `429` Rate-Limit, `5xx` temporär.  
**TRIAS-Fehler:** als strukturierte Fehlerliste mit maschinenlesbarem Code; Gateway mappt REST-Fehler → TRIAS-Fehlerdomäne.

### 4.3.8 Validierung & QS (Querschnitt)

- **Schema-Validierung:** TRIAS-XML und VSPV-REST-Schemas (OpenAPI).
- **Semantik-Prüfung:** `ServiceRef` muss existieren; Bedarfe ↔ Fahrzeug-/Dienstmerkmale konsistent; `OfferValidUntil` nicht verstrichen.

- **Messpunkte:** Erfolgsquote, P95-Latenzen, Event-Zustellrate, Idempotenztreffer, Abbruchquoten vor Zahlung.
- **Testprofile:** Referenzfälle (Standard-Taxi, Rollstuhl-Taxi, Festpreis/Taxameter, Kiosk-Buchung, Änderung, Storno).

#### 4.3.9 Praxisbeispiel (End-to-End, vereinfacht)

1. **AvailabilityRequest** (TRIAS) → Gateway → **POST /availability**  
Rückgabe: 2 Optionen (Taxi-Direkt, Taxi+Bus).
2. **OfferRequest** für Taxi-Direkt → **POST /quotes** → **Quoteld q\_123**.
3. **BookingRequest** mit **quoteld=q\_123, payment.mode="CARD"** → **POST /bookings**  
→ **BookingId b\_456, status=CONFIRMED, ETA 6 min**.
4. Webhook **ASSIGNED** mit Fahrzeugmerkmalen; später **TRIP\_STARTED / TRIP\_COMPLETED**.
5. **BookingStatusRequest** (bei Bedarf) → aktueller Status + Ankunftszeit.

#### 4.3.10 Normative Mindestanforderungen (Stufe 3)

- **MUSS:** Verfügbarkeit, Angebot, Buchung, Stornierung, Status **als TRIAS-Operationen** + REST-Mapping gemäß Tabelle 4.3-A.
- **MUSS:** Idempotenz-Schlüssel für alle Schreibvorgänge; korrelierbare Events.
- **MUSS:** DSGVO-konforme Datenminimierung; keine Freitext-Gesundheitsangaben.
- **SOLL:** SSO-Unterstützung per OIDC gemäß Kap. 7 (nahtloses Kundenerlebnis).
- **SOLL:** Webhook-Signaturen, Replay-Schutz, Rate-Limiting.
- **KANN:** Zahlungs-Pre-Authorisation im Buchungsschritt (je Zahlart), „Pay-Later“ für Kostenträgerfälle.

Mit Stufe 3 wird die Auskunft/Shop-App zum **vollwertigen Buchungskanal**. Das definierte **TRIAS-Profil** und die **REST-Bridge** zur VSPV-Buchungs-API stellen sicher, dass Verfügbarkeit, Angebot, Buchung, Änderung, Storno und Status **standardisiert, idempotent** und **sicher** abgewickelt werden. **SSO** (Kap. 7) rundet die Nutzererfahrung ab; **QS/Monitoring** und Validierung schließen Lücken zu Kap. 3.9.

## 5 Back-End-Integration in Betreiber-/Anbietersysteme (Disposition)

### 5.1 Datenschnittstellen zur Disposition (Verfügbarkeit, Zuweisung, Status)

#### 5.1.1 Zielbild und Abgrenzung

Dieses Kapitel standardisiert die **Back-End-Integration** zwischen dem **zentralen Buchungssystem** (VSPV-Kern; vgl. Kap. 3.7/4.3) und den **Betreiber-/Anbietersystemen** (Dispatch/Disposition von Taxi-, Mietwagen-, BTW-, Liegемietwagen-Anbietern). Es definiert drei Funktionsblöcke:

- **A. Verfügbarkeit:** Bereitstellung von Fahrzeug-/Kapazitätsinformationen an das zentrale System (pull/push).
- **B. Zuweisung (Dispatch):** Übergabe konkreter Buchungen zur Annahme/Zuordnung an Fahrzeuge.
- **C. Status & Telemetrie:** Rückmeldungen zum Buchungs- und Fahrzeugstatus (inkl. Ereignisse/ETA/Position).

**Nicht-Ziel:** Kundenschnittstellen (App/Kiosk), ÖPNV-Auskunft und TRIAS-Interaktion (→ Kap. 4); Rettungsleitstellen-spezifische Ergänzungen (→ Kap. 5.3).

#### 5.1.2 Grundsätze (normativ)

- **IDs/Referenzen:** Sämtliche Objekte referenzieren die in Kap. 3 festgelegten **stabilen VSPV-IDs** (OperatorRef, VehicleRef, ServiceJourneyRef, BookingId).
- **Idempotenz:** Alle **schreibenden** Operator-APIs verlangen **Idempotency-Key** (Header). Wiederholte Requests mit identischem Key liefern identische Antworten.
- **Zeitverhalten:** Annahme-/Ablehnungs-Entscheidungen für Zuweisungen müssen **innerhalb t\_assign ≤ 30 s** erfolgen (Vorgabewert; projektweit konfigurierbar).
- **Sicherheit:** mTLS + OAuth 2.1 Client Credentials; Webhooks mit HMAC-Signatur und **ts/nonce**.
- **Datenminimierung:** Nur für Durchführung erforderliche Daten (kein Freitext zu Gesundheitsdaten; Bedarfe als kodierte Flags, vgl. Kap. 3.5/3.7/4.3).
- **Transport:** HTTPS/TLS 1.2+; UTF-8; JSON für REST; optional MQTT 5.0 für Telemetrie.

#### 5.1.3 Schnittstellenübersicht

##### A) Verfügbarkeit (Operator → Zentralsystem)

Zweck: **Versorgungsgrad** ermitteln, Verfügbarkeiten planen, Angebote plausibilisieren.

## REST (Pull vom Zentralsystem)

- GET /v1/operators/{opId}/vehicles – Stammdaten/Capabilities (paging).
- GET /v1/operators/{opId}/availability?area=...&time=... – **aggregierte** Verfügbarkeit (z. B. Anzahl freier Fahrzeuge je Zone/Polygon und Zeitfenster).
- GET /v1/operators/{opId}/ranks – optionale Taxi-Stand-Queues (Position/Count pro StopPlace).

## REST (Push vom Operator)

- POST /v1/operators/{opId}/availability:publish – Aggregatsupply (zeit-/raumbezogen).
- POST /v1/operators/{opId}/vehicles/status – **Bulk**-Statusupdate (vehicle state, capability flags, optional position).
- POST /v1/operators/{opId}/ranks/occupancy – Standbelegung (frei/wartend, Queue-Länge).

## MQTT (optional, für hochfrequente Telemetrie)

- Topic: vspv/{opId}/vehicle/{vehicleId}/telemetry → { lat, lon, heading, speed, state, ts }
- QoS: 1; Rate: ≤ 1 Hz (Standard), ≤ 0.2 Hz bei geringer Dichte.

## Zulässige Fahrzeugzustände (vehicle.state)

OFFLINE | IDLE | QUEUED\_AT\_RANK | ASSIGNED | ENROUTE\_TO\_PICKUP | WAITING\_AT\_PICKUP | IN\_TRIP | UNAVAILABLE | OUT\_OF\_SERVICE

## B) Zuweisung (Zentralsystem → Operator)

Zweck: Übergabe konkreter **Buchungen** an das Dispositionssystem zur **Annahme/Zuordnung**.

## REST

- POST /v1/dispatch/proposals

Payload (vereinfacht):

```
{
  "proposalId": "prop_...",
  "bookingId": "b_...",
  "pickup": { "when": "2025-03-01T10:30:00+01:00", "placeRef":
    "VSPV:StopPlace:...", "windowMinus": "PT5M", "windowPlus": "PT10M" },
  "dropoff": { "placeRef": "VSPV:StopPlace:..." },
  "pax": 1,
  "needs": ["wheelchair" | "stretcher" | "escort" | "childSeat" ...],
  "price": { "type": "fixed|meter|estimate", "amount": 27.00, "currency":
    "EUR" },
}
```

```
"sla": { "decisionBy": "2025-03-01T10:00:30+01:00" },
"notes": { "channel": "KIOSK|APP|PHONE", "channelRef":
"VSPV:StopPlace:Klinikum-Hamm" }
}
```

- **POST /v1/dispatch/proposals/{proposalId}:accept**  
→ { "bookingId": "b\_...", "vehicleRef": "VSPV:Vehicle:...", "eta": "PT6M" }
- **POST /v1/dispatch/proposals/{proposalId}:reject**  
→ { "reason":  
"NO\_CAPACITY|OUT\_OF\_AREA|EQUIPMENT\_MISMATCH|TIME\_CONSTRAINT|TECHNICAL", "message": "..." }
- **POST /v1/dispatch/bookings/{bookingId}:cancel** – Storno aus Zentralsystem (inkl. Gebührenhinweis in Operator-Antwort).
- **GET /v1/dispatch/bookings/{bookingId}** – Statusabfrage (Operator-seitig).

### Entscheidungsautomat (Operator)

- Eingang **proposal** → **ACCEPT** (mit **vehicleRef**) **oder REJECT** bis **sla.decisionBy**.
- Keine Reaktion → **TIMEOUT** (Zentralsystem eskaliert/neu allokiert).

### C) Status, Ereignisse, Telemetrie (Operator → Zentralsystem)

Zweck: **Laufender Betrieb** und Kundenausspielung (ETA, Fortschritt).

### Webhooks (vom Operator zum Zentralsystem)

- Endpoint wird vom Zentralsystem pro Operator registriert.
- Ereignisse:
  - **BOOKING\_CONFIRMED** (nach **ACCEPT**)
  - **DRIVER\_ASSIGNED** (optional)
  - **VEHICLE\_ENROUTE** (ETA enthalten)
  - **ARRIVED\_AT\_PICKUP**
  - **TRIP\_STARTED**
  - **TRIP\_COMPLETED** (inkl. Meter/Festpreis-Endwert, sofern zulässig)
  - **BOOKING\_CANCELLED** (source=**OPERATOR|CUSTOMER|SYSTEM**; fee?)
  - **EXCEPTION** (z. B. no-show, Panne)

### REST (alternativ/zusätzlich)

- **POST /v1/operators/{opId}/events** – Bulk-Event-Upload (Batch, geordnet nach **ts**).

## 5.1.4 Datenmodelle (Auszug, normativ)

### Vehicle (Stammdatenauszug)

```
{
  "vehicleRef": "VSPV:Vehicle:XYZ-123",
  "typeRef": "VSPV:VehicleType:WAV-E",
  "capabilities": { "wheelchair": true, "stretcher": false, "escortSeat": true },
  "operatorRef": "VSPV:Operator:ABC",
  "registration": "DO-AB 1234",
  "state": "IDLE"
}
```

### Aggregierte Verfügbarkeit

```
{
  "areaRef": "VSPV:FlexibleStopPlace:Hamm-Innenstadt",
  "timeWindow": { "from": "2025-03-01T10:00:00+01:00", "to": "2025-03-01T11:00:00+01:00" },
  "supply": [
    { "vehicleTypeRef": "VSPV:VehicleType:Taxi-Std", "countIdle": 7, "countQueued": 3 },
    { "vehicleTypeRef": "VSPV:VehicleType:WAV-E", "countIdle": 1 }
  ]
}
```

### Taxi-Stand-Belegung (optional)

```
{
  "rankRef": "VSPV:StopPlace:TaxiRank-Hamm-Hbf",
  "queueLength": 4,
  "updatedAt": "2025-03-01T10:05:00+01:00"
}
```

### Event (Webhook/Batch)

```
{
  "bookingId": "b_456",
  "event": "ARRIVED_AT_PICKUP",
  "ts": "2025-03-01T10:28:30+01:00",
  "vehicleRef": "VSPV:Vehicle:XYZ-123",
  "position": { "lat": 51.389, "lon": 7.40 }
}
```

## 5.1.5 Zustandsautomaten & Mapping

**Buchungsstatus** (Zentralsystem, vgl. Kap. 4.3)

CREATED → CONFIRMED → ASSIGNED → PICKUP → IN\_PROGRESS → COMPLETED

Fehler-/Abbrüche: CANCELLED | FAILED | TIMEOUT

### Operator-Mapping (normativ)

Operator-Event	Buchungsstatus (zentral)
ACCEPT	CONFIRMED
DRIVER_ASSIGNED (opt.)	ASSIGNED
ARRIVED_AT_PICKUP	PICKUP
TRIP_STARTED	IN_PROGRESS
TRIP_COMPLETED	COMPLETED
REJECT / TIMEOUT	bleibt CREATED → Re-Allokation
BOOKING_CANCELLED	CANCELLED
EXCEPTION	FAILED (mit Code)

### Fahrzeugstatus ↔ Buchung

- ASSIGNED/ENROUTE\_TO\_PICKUP/WAITING\_AT\_PICKUP → spiegeln CONFIRMED/ASSIGNED/PICKUP.
- IN\_TRIP → IN\_PROGRESS.
- Rückkehr zu IDLE/QUEUED\_AT\_RANK nach COMPLETED.

#### 5.1.6 Fehler-/Timeout-Regeln

- **Annahme-Timeout:** Keine Antwort auf proposal bis decisionBy → TIMEOUT; Neustart Dispatch (anderer Operator).
- **Überbuchung:** 409 CONFLICT bei paralleler Fahrzeugbindung; Zentralsystem versucht nächste Option.
- **Validierung:** 422 bei Bedarfs-/Ausstattungs mismatch (wheelchair=true, Fahrzeug ohne Rampe).
- **Idempotenz:** Wiederholte accept/reject mit gleichem Idempotency-Key → identische Antwort (200/204).
- **Webhook-Zustellung:** Zustellversuch ≥ 3 Mal mit Exponential-Backoff; Signaturprüfung obligatorisch.

#### 5.1.7 Sicherheit & Compliance (Kurzprofil; Details Kap. 7/8)

- **AuthN/Z:** OAuth 2.1 (Client Credentials), scopes: vehicles:write, availability:write, dispatch:read|write, events:write.
- **mTLS** zwischen Systemen; **Webhook-HMAC** mit gemeinsamem Secret; ts/nonce gegen Replay.
- **Datenschutz:** Keine Freitext-Kommentare mit Gesundheitsinfos; Bedarfe nur als kodierte Flags.

- **Protokollierung:** Jede Entscheidung (accept/reject/cancel) mit `correlationId`.

### 5.1.8 Leistungskennzahlen & Betriebsgrenzen

- **Latenzen (P95):** `proposal` → `accept/reject` ≤ 30 s, Event-Zustellung ≤ 3 s, Telemetrie-Jitter ≤ 2 s.
- **Skalierung:** ≥ 10 Events/Sek./Operator (Baseline), Burst bis 100 Events/Sek.
- **Datenrückhalt:** Event-Historie ≥ 30 Tage, Telemetrie-Downsampling nach 24 h zulässig.

### 5.1.9 Praxisbeispiel (konkret, Kiosk-Krankenfahrt → Taxi/BTW)

1. Kioskbuchung im zentralen System erzeugt `bookingId=b_789` (Kap. 3.4).
2. Zentralsystem sendet `proposal` an Operator **MEDI-TAXI NRW** mit `needs=["stretcher"]`, `pickup=Klinikum Hamm 10:30±10 min`.
3. Operator antwortet `accept` mit `vehicleRef=VSPV:Vehicle:HAM-BTW-05`, `eta=PT8M`.
4. Operator sendet Events: `VEHICLE_ENROUTE (ETA)`, `ARRIVED_AT_PICKUP`, `TRIP_STARTED`, `TRIP_COMPLETED`.
5. Zentralsystem spiegelt Status in App/Kiosk (Kap. 4.3), löst Abrechnung gemäß Kap. 6.x an.

### 5.1.10 Mappingtabelle (fachlich → technisch)

Fachliche Anforderung	API/Objekt	Pflichtfelder (Auszug)
Aggregatsupply je Gebiet/Zeit	POST /availability:publish	areaRef, timeWindow, supply[]
Fahrzeuggestellung & Ausstattung	POST /vehicles/status	vehicleRef, state, capabilities
Zuweisung einer Buchung	POST /dispatch/proposals	bookingId, pickup, dropoff, needs, price, sla
Annahme/Ablehnung	:accept / :reject	vehicleRef bzw. reason
Laufende Ereignisse/ETA/Position	Webhook events / MQTT telemetry	bookingId, event, ts, vehicleRef, position?
Taxi-Stand-Queue (optional)	POST /ranks/occupancy	rankRef, queueLength

## 5.2 Integration von Dritt-Bestellern (Arztpraxis, Krankenhaus, Kiosk) über die zentrale Buchungs-API

### 5.2.1 Zweck und Geltungsbereich

Dieses Kapitel standardisiert die **delegierte Buchung** über **Institutionen** (z. B. Arztpraxis, Krankenhaus, Pflegeeinrichtung) und über **öffentliche Kiosksysteme** (z. B. am Bahnhof).

Die Buchungen laufen **immer** über die **zentrale VSPV-Buchungs-API** (vgl. Kap. 3.7/4.3) und werden danach über Kap. 5.1 an Betreiber/Anbieter disponiert. Im Fokus stehen:

- **Authentifizierung und Rollen** für Institutionen/Kioske,
- **kanalbezogene Pflichtdaten** (on-behalf-of),
- **Kostenträger-/Abrechnungsangaben** (ohne Freitext-Gesundheitsdaten),
- **idempotente Endpunkte** für Verfügbarkeit → Angebot/Quote → Buchung → Status → Storno.

Nicht Teil dieses Kapitels sind Rettungsleitstellen-Sonderregeln (→ Kap. 5.3).

## 5.2.2 Rollen, Objekte, Kanäle (normativ)

### Rollen

- **Organization** (Institution, z. B. Praxis/Klinik)
- **Facility** (Betriebsstätte einer Organization, optional)
- **Terminal** (registrierter Kiosk/Terminal an einem Ort)
- **Agent** (natürliche Person mit Delegationsrecht innerhalb der **Organization**; optional für Portal-Zugriff)

### Kanal-Kennungen (Header **X-Booking-Channel**)

- **ORG\_PORTAL** (Portal/API der Institution)
- **KIOSK** (öffentliches Terminal)
- **CALLCENTER** (zentraler Telefonarbeitsplatz; optional)

### Referenzen (Kap. 3.2/3.3)

- **organizationRef**, **facilityRef**, **terminalRef** (VSPV-IDs)
- **patientRef** (pseudonymisierte Kunden-/Fahrtbezug-ID)
- **costCarrierRef** (Kostenträger/Vertragsbezug, keine Freitextdiagnosen)

## 5.2.3 Sicherheit und Autorisierung

- **Transport:** TLS 1.2+; UTF-8.
- **AuthN/AuthZ:** OAuth 2.1 Client-Credentials für Maschinenzugriff (ORG\_PORTAL/KIOSK); optional OIDC-User-Token für **Agent**.
- **Scopes (mindestens):** **availability:read**, **quotes:write**, **bookings:write**, **bookings:read**, **bookings:cancel**.
- **mTLS** für Terminals (z. B. gerätegebundenes Zertifikat).
- **Rollenprüfung:** **organizationRef** und **facilityRef** müssen zum Token passen.

- **Audit:** Jede Buchung/Änderung mit `correlationId` protokollieren; Terminal-ID im Audit.

#### 5.2.4 Datenminimierung (DSGVO-konform)

- **Erlaubt (notwendig für Durchführung/Abrechnung):** Name/Telefon des Fahrgasts oder Kontaktperson, `patientRef` (pseudonym), Bedarfscodes (wheelchair, stretcher, `escort` ...), `costCarrierRef`, **genehmigungsbezogene Kennzeichen/Referenzen** (IDs/Token), Zeit/Ort.
- **Nicht zulässig:** Freitext-Gesundheitsangaben, Diagnosen, unstrukturierte PDF/Scans in der Buchung.
- **Belege/Anlagen:** ausschließlich **referenzieren** (`documentRef`) oder über gesonderte, zugriffskontrollierte Dokumentenablage (separate API/Scope), nicht im Buchungsobjekt inline speichern.

#### 5.2.5 Spezifika für Krankenfahrten mit Muster-4-Transportverordnung

**Zweck:** Standardisiert die **delegierte Buchung** medizinisch begründeter Fahrten auf Basis einer **Muster-4-Transportverordnung** (ausgestellt durch Praxis/Klinik), einschließlich **Genehmigungsstatus** der Krankenkasse und **künftiger QR-Übernahme** aus Praxissoftware.

##### Grundsätze (normativ):

- **MUSS:** Buchungen mit Kostenübernahme durch Krankenkassen führen ein strukturiertes Objekt `medicalOrdinance`.
- **MUSS:** Keine Freitext-Diagnosen; nur **kodierte Indikatoren** (z. B. Beförderungsart, Begründungskategorie, Genehmigungsstatus).
- **MUSS:** `billing.mode = "COST_CARRIER"` mit `costCarrierRef` (z. B. IK/vereinbarter Schlüssel) und `authorizationRef` (genehmigungs-/verordnungsbezogene Referenz).
- **SOLL:** Übernahme per **QR-Code** aus dem Muster-4-Dokument; Verifikation über Signatur/Hash sobald verfügbar.
- **SOLL:** Minimalprinzip – nur die für Durchführung/Abrechnung **zwingend** erforderlichen Felder.

#### 5.2.6 Datenmodell-Erweiterung `medicalOrdinance`

`medicalOrdinance` wird in `/quotes` und `/bookings` als strukturierter Bestandteil übergeben.

```
{
  "medicalOrdinance": {
    "type": "MUSTER_4_V1",
    "ordinanceId": "M4-20250301-ABCD1234",    // aus QR oder generiert
    "issueDate": "2025-03-01",
```

```

"validUntil": "2025-03-31",
"issuingPhysician": { "lanr": "1234567890", "bsnr": "123456789" },
"beneficiary": { "patientRef": "VSPV:Patient:KH-123456", "insuranceld":
"AOK-...", "coPayExempt": false },
"transport": {
  "mode": "TAXI|MIETWAGEN|BTW|LIEGENMIETWAGEN",
  "needs": ["wheelchair","stretcher","escort"], // kodierte Flags
  "pickupAssist": "NONE|DOOR_TO_DOOR|BED_TO_BED"
},
"indication": {
  "category":
"DIALYSIS|RADIATION|CHEMOTHERAPY|ADMISSION|DISCHARGE|TRANS
FER|OTHER_CODED",
  "codeSystem": "VSPV-M4-IND-1",
  "code": "ADMISSION"
},
"approval": {
  "status": "NOT_REQUIRED|PENDING|APPROVED|DENIED",
  "approvalRef": "GK-APP-2025-00042" // falls vorhanden
},
"qr": {
  "raw": "...", // QR-Payload (Base64 oder Klartext)
  "hash": "sha256:...", // idempotenz-/duplikat-Check
  "signature": "...", // optional
  "issuer": "KVDat|PraxisSW|..." // optional
}
}
}

```

#### Hinweise (normativ):

- `mode` bestimmt die **erforderliche Fahrzeug-/Ausstattungs-kategorie** (Zuweisung siehe Kap. 5.1).
- `indication.*` ist **kodiert**; keine Freitext-Diagnosen.
- `approval.status` **MUSS** gepflegt sein. `NOT_REQUIRED` nur bei genehmigungsfreien Fällen.
- `qr.hash` **SOLL** zur **Idempotenzprüfung** genutzt werden (Duplikate derselben Verordnung verhindern).

### 5.2.7 Endpunkte: Übergabe von Muster-4-Daten

#### A) Quote erweitern

POST /v1/quotes

```

{
  "optionId": "opt_9a1",

```

```

"channel": "ORG_PORTAL",
"organizationRef": "VSPV:Org:Klinikum-Hamm",
"billing": {
  "mode": "COST_CARRIER",
  "costCarrierRef": "VSPV:CostCarrier:IK123456789",
  "authorizationRef": "M4-20250301-ABCD1234"
},
"medicalOrdinance": { ... wie oben ... }
}

```

#### Regeln:

- **MUSS:** `billing.mode = COST_CARRIER` ⇒ `authorizationRef` und `medicalOrdinance` vorhanden.
- **SOLL:** `authorizationRef` ≙ `ordinanceld`.
- **SOLL:** `approval.status ≠ DENIED` (sonst 412 Precondition Failed).

#### B) Booking erweitern

POST `/v1/bookings` (mit `Idempotency-Key`)

- Payload enthält **identisches** `medicalOrdinance` wie in Quote (Server prüft Konsistenz).
- **Antwort** ergänzt Buchung um `medical.flags = { requiresQualifiedVehicle: true|false }` gemäß Mapping `mode/needs`.

### 5.2.8 QR-Code-Import (Praxissoftware/Kiosk)

**Ziel: Vorausfüllung** der Buchungsmaske aus dem QR-Code der Muster-4-Verordnung.

**API (Parsing-Service):** POST `/v1/medical/ordinances:parse`

```
{ "qrPayload": "..." }
```

#### Antwort:

```

{
  "ordinance": { ...medicalOrdinance... },
  "normalized": true,
  "warnings": ["SIGNATURE_MISSING"] // optional
}

```

#### Normative Regeln:

- **MUSS:** Der Parser normalisiert Felder (Datumsformate, Kodierungen).
- **SOLL:** `qr.hash` berechnen und in Quote/Booking übernehmen.
- **SOLL:** Wenn Signaturprüfung möglich ist, `issuer/signature` validieren und Flag setzen (`verified=true`).
- **MUSS:** Keine Übernahme nicht benötigter Patientendaten; **nur** strukturierte Mindestangaben.

### Kiosk-Flow (barrierearm):

1. QR einscannen → `ordinances:parse`.
2. Maske mit vorausgefüllten Feldern (`mode`, `needs`, `approval.status`, `authorizationRef`) anzeigen.
3. Zieladresse wählen, Kontakttelefon ergänzen → `availability/quotes/bookings`.

#### 5.2.9 Validierung (spezifisch Muster-4)

- **Schema:** `medicalOrdinance.type = MUSTER_4_V1`.
- **Pflichtfelder:** `ordinanceld`, `issueDate`, `issuingPhysician.lanr`, `transport.mode`.
- **Genehmigung:**
  - `approval.status = APPROVED` oder `NOT_REQUIRED`.
  - `PENDING` zulässig **nur**, wenn Policy dies vorsieht (Konfiguration); sonst 412.
- **Idempotenz:** Kombination (`authorizationRef` oder `ordinanceld` oder `qr.hash`) **MUSS** idempotent sein – Wiederholbuchungen verhindern.
- **Dublettenprüfung:** Falls `ordinanceld` bereits **verbraucht** (gebucht & gefahren), dann 409 CONFLICT.

#### 5.2.10 Fehlercodes (Ergänzungen)

- 412 `PRECONDITION_FAILED`: Genehmigung erforderlich, `approval.status` unzulässig.
- 409 `CONFLICT`: Duplikat derselben Verordnung (`qr.hash/ordinanceld` bereits verwendet).
- 422 `UNPROCESSABLE_ENTITY`: Inkonsistente Felder (z. B. `mode=LIEGENMIETWAGEN` ohne `needs["stretcher"]`).
- 400 `BAD_REQUEST`: QR-Payload nicht parsebar/ungültiges Format.

#### 5.2.11 Praxisbeispiel: Entlassfahrt mit QR-Übernahme (Klinikum Hamm)

1. Stationskiosk scannt Muster-4-QR → `ordinances:parse` liefert `medicalOrdinance` (`mode=MIETWAGEN`, `needs=["stretcher"]`, `approval.status=APPROVED`, `authorizationRef=M4-20250301-ABCD1234`).
2. `availability` (`ORG_PORTAL`) mit vorausgefüllten Parametern → Optionen.
3. `quotes` mit `billing.mode=COST_CARRIER`, `costCarrierRef=IK123456789`, `authorizationRef=M4-...`, `medicalOrdinance` anhängen → Preis/Konditionen.
4. `bookings` (Idempotency-Key = `qr.hash`) → `status=CONFIRMED`, `eta=PT8M`.
5. Events gemäß Kap. 5.1; Abrechnungsvorbereitung gemäß Kap. 6.

## 5.2.12 Ergänzung der Mappingtabelle (Krankenfahrten)

Fachliche Anforderung	API/Objekt	Pflichtfelder / Regeln
Muster-4-Verordnung übergeben	/quotes, /bookings	billing.mode=COST_CARRIER, authorizationRef, medicalOrdinance.*
Genehmigungsstatus prüfen	Validierung	approval.status ∈ {NOT_REQUIRED,APPROVED} sonst 412
QR-Code aus Praxissoftware übernehmen	/medical/ordinances:parse	qrPayload; Rückgabe medicalOrdinance, qr.hash
Idempotenz über Verordnung sichern	Header + Felder	Idempotency-Key <b>und</b> qr.hash/ordinanceld
Auswahl geeigneter Fahrzeugklasse (Taxi/BTW/Liegemietwagen)	Dispatch (Kap. 5.1)	Mapping transport.mode/needs → VehicleType/capabilities
Datenschutz (keine Diagnosen, nur Codes)	Payload	indication kodiert; keine Freitexte

## 5.2.13 Endpunkte für Dritt-Besteller (fachlich kanonisch)

Die folgenden Ressourcen sind **Spezialisierungen** der in Kap. 3.7 definierten Kern-APIs; zusätzliche Felder sind **kanalgebunden** (Organization/Kiosk).

### A) Verfügbarkeit

POST /v1/availability

```
{
  "origin": { "stopPlaceRef": "VSPV:StopPlace:Klinikum-Hamm" },
  "destination": { "stopPlaceRef": "VSPV:StopPlace:Hamm-Innenstadt" },
  "when": { "type": "DEPARTURE_AT", "time": "2025-03-01T10:30:00+01:00",
  "windowMinus": "PT0M", "windowPlus": "PT30M" },
  "pax": 1,
  "needs": ["stretcher"],
  "channel": "ORG_PORTAL",
  "organizationRef": "VSPV:Org:Klinikum-Hamm",
  "facilityRef": "VSPV:Facility:KH-Hamm-Zentrale",
  "onBehalfOf": { "patientRef": "VSPV:Patient:KH-123456" }
}
```

Antwort (gekürzt):

```
[
  {
    "optionId": "opt_9a1",
```

```
"serviceRef": "VSPV:ServiceJourney:Taxi-4711",
"time": { "planned": "2025-03-01T10:35:00+01:00" },
"priceHint": { "type": "fixed", "amount": 27.00, "currency": "EUR" },
"policyHints": { "leadTime": "PT0M", "cancelUntil": "PT10M" }
}
]
```

## B) Angebot/Quote

POST /v1/quotes

```
{
  "optionId": "opt_9a1",
  "channel": "ORG_PORTAL",
  "organizationRef": "VSPV:Org:Klinikum-Hamm",
  "facilityRef": "VSPV:Facility:KH-Hamm-Zentrale",
  "onBehalfOf": {
    "patientRef": "VSPV:Patient:KH-123456",
    "contact": { "name": "Max Mustermann", "phone": "+49 2335 123456" }
  },
  "billing": {
    "mode": "COST_CARRIER",
    "costCarrierRef": "VSPV:CostCarrier:GK-80-IK123456789",
    "authorizationRef": "VSPV:Auth:KH-OP-20250301-42"
  }
}
```

Antwort:

```
{ "quoteId": "q_123", "price": { "amount": 27.00, "currency": "EUR" },
  "validUntil": "2025-03-01T10:20:00+01:00", "conditions": [] }
```

## C) Buchung

POST /v1/bookings (Header: Idempotency-Key)

```
{
  "quoteId": "q_123",
  "channel": "ORG_PORTAL",
  "organizationRef": "VSPV:Org:Klinikum-Hamm",
  "facilityRef": "VSPV:Facility:KH-Hamm-Zentrale",
  "onBehalfOf": {
    "patientRef": "VSPV:Patient:KH-123456",
    "contact": { "name": "Max Mustermann", "phone": "+49 2335 123456" }
  },
  "billing": {
    "mode": "COST_CARRIER",
    "costCarrierRef": "VSPV:CostCarrier:GK-80-IK123456789",
    "authorizationRef": "VSPV:Auth:KH-OP-20250301-42",
    "caseRef": "KH-Asp-Entlassung-4711"
  }
}
```

```

    },
    "consents": { "dataProcessing": true, "termsAccepted": true }
  }
  Antwort:
  {
    "bookingId": "b_456",
    "status": "CONFIRMED",
    "eta": "PT8M",
    "links": { "status": "/v1/bookings/b_456" }
  }

```

## D) Änderung & Storno

- `POST /v1/bookings/{id}:modify` – Zeit/Ziel/Bedarfe ändern (liefert ggf. neue Quote → Confirm).
- `POST /v1/bookings/{id}:cancel` – mit optionalem Grund/Fee-Hinweis.
- `GET /v1/bookings/{id}` – Status/ETA/Vehicle (sofern zulässig).

### 5.2.14 Pflichtfelder nach Kanal

Feld	ORG_PORTAL	KIOSK
organizationRef	MUSS	–
facilityRef	SOLL	–
terminalRef	–	MUSS
onBehalfOf.patientRef	MUSS	KANN
onBehalfOf.contact	MUSS (mind. Telefon <b>oder</b> Ansprechpartner der Einrichtung)	MUSS (Telefon <b>oder</b> Vorname+Erkennungsmerkmal)
billing.mode	MUSS	MUSS
billing.costCarrierRef	MUSS bei COST_CARRIER	MUSS bei COST_CARRIER

**Hinweis:** Kiosk-Masken sind **barrierearm** und datenminimiert; nur zwingend erforderliche Felder.

### 5.2.15 Kostenträger & Abrechnung (Schnittstelle zu Kap. 6)

- `billing.mode`: CASH | CARD | MOBILE | INVOICE | COST\_CARRIER.
- `costCarrierRef`: kodierte Referenz (z. B. IK-Nr. o. vereinbarter Schlüssel).
- `authorizationRef`: **strukturierte** Genehmigungs-/Freigabe-ID (kein Freitext).
- `caseRef`: fallbezogene Referenz der Einrichtung (für Nachweise).
- Zahlungsfluss und spätere **E-Rechnung** werden in Kap. 6 spezifiziert; hier erfolgt nur die **Buchungsanreicherung**.

## 5.2.16 Validierung, Idempotenz, Fehlercodes

**Idempotenz:** Schreibende Calls verlangen Idempotency-Key.  
**Schema-/Semantikprüfungen (Auszug):**

- IDs und Feldgrößen gemäß Kap. 3.3,
- `needs[]` gegen verfügbare Dienst-/Fahrzeugmerkmale,
- `authorizationRef` nur wenn `billing.mode=COST_CARRIER`,
- `organizationRef/facilityRef/terminalRef` muss zum Token passen.

### Fehlercodes

- `400` Ungültige Felder/Format,
- `401/403` Auth/Scope/Policy verletzt (z. B. falscher Kanal),
- `404` Referenz unbekannt,
- `409` Doppelbuchung/Idempotenzkonflikt,
- `412` Policy-Verstoß (z. B. fehlende Genehmigung für `COST_CARRIER`),
- `422` Bedarfe vs. Angebot nicht erfüllbar,
- `429` Rate-Limit,
- `5xx` temporär.

## 5.2.17 Terminal-/Kiosk-Lifecycle

- **Registrierung:** `POST /v1/terminals` (durch Betreiber des zentralen Systems), Ausgabe `terminalRef`, Zertifikat/Client-ID.
- **Konfiguration:** zugeordnetes `StopPlaceRef`, Öffnungszeiten (vgl. Kap. 3.4 KioskAccess), Lokalisierung (Sprache, Barrierefrei-Modus).
- **Health:** `POST /v1/terminals/{terminalRef}:heartbeat` (optional), Offline-Queue mit späterem **Retry**.
- **Sicherheit:** mTLS, fest zugewiesener Scope `KIOSK`.

## 5.2.18 Praxisbeispiele

### A) Entlassfahrt aus Klinik (ORG\_PORTAL)

1. Klinikportal ruft `availability (needs=stretcher)`, Abfahrt 10:30).
2. `quote` liefert Festpreis 27 EUR (`COST_CARRIER + authorizationRef`).
3. `bookings` bestätigt (`CONFIRMED`, ETA 8 Min.).
4. Status-Events (Kap. 5.1) aktualisieren Anzeige am Stationsmonitor; Abrechnung vorbereitet (Kap. 6).

## B) Bahnhof-Kiosk (KIOSK)

1. Terminal mit `terminalRef` zeigt Zielauswahl; Fahrgast wählt Adresse, `pax=1`, Bedarf `wheelchair`.
2. `availability` → `quote` → `bookings` (Zahlart CARD), Ausdruck/QR optional.
3. Status-Anzeige am Kiosk bis Abholung; keine personenbezogenen Daten außer Kontakttelefon.

### 5.2.19 Mappingtabelle (fachlich → technisch)

Fachliche Anforderung	API/Objekt	Pflichtfelder/Regel
Delegierte Buchung durch Klinik/Arztpraxis	<code>/availability</code> → <code>/quotes</code> → <code>/bookings</code>	<code>channel=ORG_PORTAL</code> , <code>organizationRef</code> , <code>onBehalfOf</code>
Kiosk-Buchung am Bahnhof	wie oben	<code>channel=KIOSK</code> , <code>terminalRef</code> , reduzierte Felder
Bedarfscodes (z. B. Rollstuhl/Trage)	<code>needs[]</code>	Kodierte Flags, keine Freitexte
Kostenträger-Abrechnung	<code>billing.*</code>	<code>mode=COST_CARRIER</code> , <code>costCarrierRef</code> , <code>authorizationRef</code>
Idempotenz & Audit	Header + Protokoll	Idempotency-Key, <code>correlationId</code>
Datenschutz	Payload	keine Gesundheits-Freitexte; nur strukturierte Referenzen
Zustands-/ETA-Rückkanal	Kap. 5.1 Events/Webhooks	<code>bookingId</code> , <code>event</code> , <code>ts</code> , optional Position

## 5.3 Sonderfall: Anbindung von Rettungsleitstellen (niedrigschwellige Krankentransporte)

### Zweck und Geltungsbereich

Dieses Unterkapitel beschreibt die technische Einbindung von **Rettungsleitstellen** als **autorisierte Drittbesteller** für **nicht-qualifizierte Krankentransporte** im Gelegenheitsverkehr (Taxi/Mietwagen, Behindertentransportwagen, Liegemietwagen/Tragestuhl). Die Buchungen werden über die zentrale VSPV-Buchungs-API (vgl. Kap. 3.7) in die Betreiber-/Anbietersysteme übergeben. Die **Abrechnung gegenüber Kostenträgern** (z. B. Krankenkassen) wird in Kap. 6 spezifiziert; hier werden die **modellseitigen Pflichtattribute** festgelegt, die diese Abrechnung ermöglichen.

- **Rollenmodell und Berechtigungen**
- **Rolle** `MEDICAL_DISPATCH` (**Rettungsleitstelle**): darf stellvertretend für Patient\*innen Krankentransporte buchen/stornieren, Status abrufen.

- **Rolle** OPERATOR\_ADMIN (**Unternehmen/Anbieter**): pflegt Stammdaten (Fahrzeugtypen, Service-Angebote, Berechtigungen).
- **Rolle** KOSTENTRAEGER\_PORTAL (**optional**): Abruf abrechnungsrelevanter Leistungsdaten.
- **Berechtigter Leistungserbringer**: Nur Anbieter mit Unternehmensmerkmal „BERECHTIGTER\_LEISTUNGSERBRINGER“ dürfen medizinische Niedrigschwellen-Transporte annehmen.

### Unternehmensmerkmal (Pflicht)

**Ort der Modellierung:** Operator (ResourceFrame) und **spiegelnd** auf Service/VehicleType (für differenzierte Zulassungen).

### Schlüssel (KeyValue, vgl. 3.6):

- BERECHTIGTER\_LEISTUNGSERBRINGER = true|false (Bool, Pflicht auf Operator-Ebene)
- BERECHTIGUNG\_GUELTIG\_VON = ISO-Datum
- BERECHTIGUNG\_GUELTIG\_BIS = ISO-Datum (optional)
- LEISTUNGSKATALOG = Liste zulässiger Transportarten (siehe unten)
- **Transportkategorien (kontrolliertes Vokabular)**
- TRANSPORTKATEGORIE (Enum): TAXI | MIETWAGEN | BTW | LIEGEMIETWAGEN | TRAGESTUHL
- Zusätze als Flags:
  - WHEELCHAIR\_ACCESSIBLE = true|false
  - LYING\_TRANSPORT = true|false
  - ASSISTANCE\_REQUIRED = NONE|ONE\_PERSON|TWO\_PERSONS (z. B. Tragestuhl)

Diese Merkmale sind bei **VehicleType** und/oder **FlexibleServiceJourney** zu hinterlegen, um disponierbare Ressourcen passend zu filtern.

### Datenmodell-Erweiterungen (NeTeX/VDV-462, KeyValue/Extensions)

#### Operator (Ausschnitt)

```
<Operator id="VSPV:Operator:MedTaxi-Ruhr" version="1.0">
  <Name><Text language="de">MedTaxi Ruhr</Text></Name>
  <keyList>

  <KeyValue><Key>BERECHTIGTER_LEISTUNGSERBRINGER</Key><Value>t
  rue</Value></KeyValue>
    <KeyValue><Key>BERECHTIGUNG_GUELTIG_VON</Key><Value>2025-
  01-01</Value></KeyValue>
```

```

<KeyValue><Key>LEISTUNGSKATALOG</Key><Value>BTW,LIEGEMIETWA
GEN,TRAGESTUHL</Value></KeyValue>

<KeyValue><Key>IK_LEISTUNGSERBRINGER</Key><Value>123456789</V
alue></KeyValue>
</keyList>
</Operator>

```

### VehicleType (Ausschnitt)

```

<VehicleType id="VSPV:VehicleType:LMW-TRAGESTUHL" version="1.0">
  <Name><Text language="de">Liegemietwagen
(Tragestuhl)</Text></Name>
  <keyList>

<KeyValue><Key>TRANSPORTKATEGORIE</Key><Value>LIEGEMIETWAGE
N</Value></KeyValue>
  <KeyValue><Key>TRAGESTUHL</Key><Value>>true</Value></KeyValue>

<KeyValue><Key>LYING_TRANSPORT</Key><Value>>true</Value></KeyVal
ue>

<KeyValue><Key>ASSISTANCE_REQUIRED</Key><Value>TWO_PERSONS
</Value></KeyValue>
  </keyList>
</VehicleType>

```

### BookingProcess (Ausschnitt)

```

<BookingProcess id="VSPV:BookingProcess:MedicalDispatch"
version="1.0">
  <Name><Text language="de">Rettungsleitstelle –
Krankentransport</Text></Name>
  <keyList>

<KeyValue><Key>BOOKING_CHANNEL</Key><Value>MEDICAL_DISPATC
H</Value></KeyValue>

<KeyValue><Key>MIN_LEAD_TIME</Key><Value>PT10M</Value></KeyVal
ue>

<KeyValue><Key>ALLOWED_TRANSPORTKATEGORIEN</Key><Value>BTW
,LIEGEMIETWAGEN,TRAGESTUHL</Value></KeyValue>
  </keyList>
</BookingProcess>

```

## Buchung (Extensions, falls benötigt)

```
<Extensions>
  <vspv:MedicalDispatch>
    <vspv:CaseId>RL-CASE-2025-000317</vspv:CaseId>
    <vspv:PatientInitials>M.K.</vspv:PatientInitials>
    <vspv:KostentraegerCode>GKV:IKK-123456</vspv:KostentraegerCode>
    <vspv:AuthorisationRef>GEM-2025-AB-7711</vspv:AuthorisationRef>
  </vspv:MedicalDispatch>
</Extensions>
```

Hinweis: Personenbezogene Inhalte sind zu **minimieren** (Initialen/Referenzen statt Diagnosen). Vollständige personenbezogene Daten fließen nur, wenn abrechnungs- oder durchführungsbedingt erforderlich.

## REST-API-Spezifika für Rettungsleitstellen (Profil auf Basis 3.7)

### Authentifizierung & Rollen

- OAuth2/OIDC, **Client Credentials** für Leitstellen-Backend.
- Pflichtr scope: `availability:read`, `quote:write`, `booking:write`, `booking:read`.
- Header `X-Booking-Channel: MEDICAL_DISPATCH` (oder Feld `bookingChannel`).

### Pflichtfelder (zusätzlich zu 3.7)

- `bookingChannel = "MEDICAL_DISPATCH"`
- `actingOnBehalf` mit `{ "type": "EMERGENCY_DISPATCH", "name": "Rettungsleitstelle XYZ", "authorisationRef": "RL-CASE-..." }`
- `requirements` passend zur Transportkategorie (z. B. `lyingTransport=true`, `assistanceRequired`)
- `payer.type="KOSTENTRAEGER"`, `payer.code` (Kostenträgerkennung)
- `medical.flags` (nur Bool/Enums, keine Diagnosen): `infectionControl`, `mobilityImpairment`, `requiresStretcher`, ... (optional)

### Beispiel: `/v1/bookings` (POST)

```
{
  "quoteId": "q_01HAJ9...",
  "bookingChannel": "MEDICAL_DISPATCH",
  "customer": {
    "type": "PERSON",
    "name": "—",
    "actingOnBehalf": {
      "type": "EMERGENCY_DISPATCH",
      "name": "Rettungsleitstelle Ruhr",
      "authorisationRef": "RL-CASE-2025-000317"
    }
  }
}
```

```

},
"requirements": {
  "lyingTransport": true,
  "assistanceRequired": "TWO_PERSONS",
  "wheelchairAccessible": false
},
"payment": {
  "mode": "INVOICE",
  "payer": { "type": "KOSTENTRAEGER", "code": "GKV:IKK-123456" }
},
"medical": { "infectionControl": false },
"consents": { "termsAccepted": true, "dataProcessingAccepted": true }
}

```

### Geschäftsregeln (serverseitig, Validierung 422 bei Verstoß)

1. **Berechtigungsprüfung:** die zugewiesenen `Operator/VehicleType` müssen `BERECHTIGTER_LEISTUNGSERBRINGER=true` führen und die angefragte `TRANSPORTKATEGORIE` unterstützen.
2. **Kostenträgerprüfung:** `payer.code` muss bekannt/zugelassen sein; ggf. `AuthorisationRef` (Genehmigungskennzeichen) erforderlich.
3. **Datenminimierung:** keine Freitextdiagnosen; medizinische Infos nur als **kodierte Flags**.
4. **Zeitregeln:** Einhaltung `MIN_LEAD_TIME` aus `BookingProcess`.
5. **Zuweisung:** Nur Fahrzeuge mit passenden Merkmalen (`LYING_TRANSPORT`, `TRAGESTUHL`, `ASSISTANCE_REQUIRED`) dürfen disponiert werden.
6. **Audit:** Jede Buchung/Änderung ist revisionssicher mit `correlationId` und `actingOnBehalf` zu protokollieren.

### Prozessablauf (Sequenz)

1. **Verfügbarkeit** (`/availability`): Leitstelle übermittelt Start/Ziel, Zeitfenster, Anforderungen (z. B. `lyingTransport`).
2. **Quote** (`/quotes`): Tarif/Preis gemäß `FareStructure` bzw. Kostenträger-Regeln.
3. **Buchung** (`/bookings`): mit `bookingChannel=MEDICAL_DISPATCH`, `actingOnBehalf`, Kostenträgerdaten.
4. **Events/Webhooks:** `BOOKING_CONFIRMED` → `ASSIGNED` (Fahrzeug/ETA) → `TRIP_STARTED` → `TRIP_COMPLETED`.
5. **Abrechnungsvorbereitung:** Leistungsnachweis (Zeit/Ort, Fahrzeugtyp, Transportkategorie, Fallnummer); Übergabe an Kap. 6-Prozesse.

## Qualitäts- und Sicherheitsanforderungen

- **Scopes & Rollenpflicht:** ohne MEDICAL\_DISPATCH-Rolle keine Buchung dieser Kategorie.
- **Doppelbuchungsvermeidung:** Idempotency-Key obligatorisch.
- **Protokollierung:** alle Zugriffe/Änderungen mit correlationId; Aufbewahrungsfristen gemäß Betreiberpolitik.
- **Testprofile:** Bereitstellung von Beispiel-XML/JSON (Anhang) mit LMW/BTW/Tragestuhl-Fällen.

## Beispiel (NeTEx – kompaktes End-to-End)

```
<FlexibleServiceJourney id="VSPV:SJ:LMW-DO-2025-001" version="1.0">
  <Name><Text language="de">Liegemietwagen – Raum
  Dortmund</Text></Name>
  <keyList>

  <KeyValue><Key>TRANSPORTKATEGORIE</Key><Value>LIEGEMIETWAGE
  N</Value></KeyValue>

  <KeyValue><Key>ASSISTANCE_REQUIRED</Key><Value>TWO_PERSONS
  </Value></KeyValue>
  </keyList>
  <BookingProcessRef ref="VSPV:BookingProcess:MedicalDispatch"/>
  <OperatorRef ref="VSPV:Operator:MedTaxi-Ruhr" version="1.0"/>
  <VehicleTypeRef ref="VSPV:VehicleType:LMW-TRAGESTUHL"
  version="1.0"/>
</FlexibleServiceJourney>
```

# 6 Abrechnungslogik und Kostenträger-Integration

## 6.1 Zahlungsinformationen und Fahrpreisbildung

### Zweck und Geltungsbereich

Dieses Unterkapitel standardisiert die **Preisbildung** und die **Zahlungsdaten** für Gelegenheitsverkehre (Taxi/Mietwagen sowie Spezialfahrzeuge) im VSPV-Standard 101. Es definiert:

- die **Tarifrepräsentation** in NeTEx/VDV 462 (Profil) einschließlich Festpreis, zeit-/wegabhängiger Preisbildung und Zuschlägen,
- die **Zahlarten** samt Abwicklungslogik in der Buchung,
- die Datenspuren für **Kassenbeleg/Quittung** und spätere **Abrechnung** (Kap. 6.2/6.3).

- **Grundsätze (normativ)**
- **MUSS:** Tarif-/Preisobjekte werden in einem **FareFrame** geführt; operative Zahlungsabsichten in der **Buchung**.
- **MUSS:** Preisangaben sind **brutto** in **EUR** mit zwei Dezimalstellen.
- **MUSS:** Festpreise vs. laufende Preisbildung sind unterscheidbar modelliert.
- **SOLL:** Zuschläge/Zeitregeln werden **strukturiert** modelliert (keine Freitextsummen).
- **DARF:** dynamische Preise (Quote) zur Buchungszeit fixiert werden; die Quote wird referenziert.
- **Tarifrepräsentation (NeTeX/VDV 462 Profil)**

#### Objekte (Ausschnitt):

- **FareFrame** (Container)
- **FareProduct** (z. B. TAXI\_FIXED, TAXI\_METERED)
- **FareStructure** (Regelwerk: Zonen/Entfernungen/Zeiten/Zuschläge)
- **FareTable/PriceGroup** (konkrete Beträge)
- **Conditions** (Bedingungen, z. B. Gültigkeit, Mitnahme)

#### Beispiel (Festpreis):

```
<FareFrame id="VSPV:FareFrame:Taxi-NRW" version="1.0">
  <FareProducts>
    <FareProduct id="VSPV:FareProduct:TAXI_FIXED" version="1.0">
      <Name><Text language="de">Taxi Festpreis</Text></Name>
      <TypeOfFareProductRef ref="VSPV:ToFP:TaxiFixed"/>
    </FareProduct>
  </FareProducts>
  <FareStructures>
    <FareStructure id="VSPV:Fare:CGN-City-Fixed" version="1.0">
      <Name><Text language="de">Festpreis CGN↔City</Text></Name>
      <PriceGroups>
        <PriceGroup id="VSPV:PriceGroup:CGN-City">
          <Amount>35.00</Amount><Currency>EUR</Currency><PriceType>fixed<
/PriceType>
        </PriceGroup>
      </PriceGroups>
      <ValidBetween><FromDate>2025-01-01</FromDate></ValidBetween>
    </FareStructure>
  </FareStructures>
</FareFrame>
```

### Beispiel (zeit-/wegabhängig, Quotenfähig):

```
<FareStructure id="VSPV:Fare:Taxi-Metered" version="1.0">
  <Name><Text language="de">Taxi Zeit/Weg</Text></Name>
  <Conditions>
    <Condition><Text language="de">Grundpreis + €/km + €/min;
    Nachtzuschlag</Text></Condition>
  </Conditions>
  <keyList>

  <KeyValue><Key>BASE_FEE_EUR</Key><Value>4.50</Value></KeyValue>
  <KeyValue><Key>PER_KM_EUR</Key><Value>2.10</Value></KeyValue>

  <KeyValue><Key>PER_MIN_EUR</Key><Value>0.55</Value></KeyValue>

  <KeyValue><Key>NIGHT_SURCHARGE_EUR</Key><Value>3.00</Value><
  /KeyValue>

  <KeyValue><Key>NIGHT_FROM</Key><Value>22:00</Value></KeyValue>
  <KeyValue><Key>NIGHT_TO</Key><Value>06:00</Value></KeyValue>
  </keyList>
</FareStructure>
```

**Hinweis:** Parameterisierte Fahrpreisregeln werden **profilkonform** als KeyValue-Paare hinterlegt, solange kein dediziertes Attribut im VDV-Profil existiert. Festpreise nutzen **PriceGroup**. Für dynamische Preise wird zusätzlich eine **Quote** im Buchungsfluss erzeugt (s. u.).

### Zahlungsarten und Buchungsdaten

#### Zahlarten

(enum)

CASH | CARD | MOBILE | IN\_APP | INVOICE | COST\_CARRIER

#### Zahlungsvereinbarung in der Buchung (Auszug):

```
"payment": {
  "mode": "CARD",
  "capture": "AFTER_SERVICE", // ON_BOOKING | AFTER_SERVICE
  "providerRef": " PSP-or-merchant-id ",
  "tokenRef": " tok_abc123 ", // optional für IN_APP
  "receiptRequested": true
}
```

#### Quote (Preisfixierung zur Buchung):

```
"quote": {
  "quoteId": "q_123",
  "fareRef": "VSPV:Fare:Taxi-Metered",
  "amount": 27.00, "currency": "EUR",
  "validUntil": "2025-03-01T10:20:00+01:00"
```

}

### Normative Regeln:

- **MUSS:** Bei **INVOICE** oder **COST\_CARRIER** ist **keine** sofortige Zahlungsabwicklung zu triggern; stattdessen Abrechnung (Kap. 6.2).
- **SOLL:** **receipt** wird als strukturierter Datensatz bereitgestellt (Zeit/Ort, Beträge, MwSt-Sätze, Zahlungsart).
- **MUSS:** Rundung **kaufmännisch** auf zwei Dezimalstellen.

## 6.2 Abrechnung mit Kostenträgern (Krankenkassen, Kommunen)

### Zweck und Geltungsbereich

Dieses Unterkapitel standardisiert die **abrechnungsrelevanten Daten** und den **Prozessfluss** für Drittkostenträger. Es verbindet Buchung/Leistung mit **Anspruchs-/Rechnungsobjekten** und definiert Validierungen, ohne medizinische Freitexte zu speichern.

- **Grundsätze (normativ)**
- **MUSS:** Kostenträgerfälle werden strukturiert in einem **BillingFrame** (VSPV-Erweiterung) geführt.
- **MUSS:** Ein **Claim** (Leistungsanspruch) verweist auf **Booking, ServiceJourney/FlexibleServiceJourney, CostCarrier, Authorization**.
- **MUSS:** Keine Diagnosen/Freitexte; nur **kodierte** Indikatoren (vgl. Kap. 5.2 Muster-4).
- **SOLL:** Statusmodell: **OPEN → SUBMITTED → ACKNOWLEDGED → APPROVED/REJECTED → PAID**.

### Datenobjekte (VSPV-BillingFrame)

#### Übersicht:

- **CostCarrier** (Stammdaten/IK oder vereinbarter Schlüssel)
- **Contract** (Leistungs-/Vergütungsvereinbarung)
- **Authorization** (z. B. Verordnungs-/Genehmigungs-Ref; vgl. Kap. 5.2)
- **Claim** (Anspruch mit Positionen)
- **ClaimLineItem** (Einzelposition Fahrt/Zuschlag/Wartezeit)
- **Invoice** (aggregierte Abrechnungsrechnung; optional pro Claim oder Bündel)
- **Settlement** (Zahlungseingang/Verrechnung)

### XML-Ausschnitt (vereinfachtes Profil):

```
<BillingFrame id="VSPV:BillingFrame:NRW" version="1.0"
xmlns:vspv="http://vspv.dev/ns">
```

```

<CostCarriers>
  <vspv:CostCarrier id="VSPV:CostCarrier:IK123456789" version="1.0">
    <Name><Text language="de">Beispiel-Krankenkasse</Text></Name>
  </vspv:CostCarrier>
</CostCarriers>

<Contracts>
  <vspv:Contract id="VSPV:Contract:IK123456789-2025" version="1.0">
    <CostCarrierRef ref="VSPV:CostCarrier:IK123456789"/>
    <ValidBetween><FromDate>2025-01-01</FromDate></ValidBetween>
    <keyList>

<KeyValue><Key>TARIFF_MODEL</Key><Value>TAXI_CONTRACT_2025</
Value></KeyValue>
    </keyList>
  </vspv:Contract>
</Contracts>

<Claims>
  <vspv:Claim id="VSPV:Claim:2025-000042" version="1.0">
    <BookingRef ref="b_456"/>
    <CostCarrierRef ref="VSPV:CostCarrier:IK123456789"/>
    <AuthorizationRef ref="M4-20250301-ABCD1234"/>
    <Status>OPEN</Status>
    <LinItems>
      <vspv:ClaimLinItem id="VSPV:CLI:1">
        <ServiceRef ref="VSPV:ServiceJourney:Taxi-4711"/>

<Price><Amount>27.00</Amount><Currency>EUR</Currency></Price>
        <Category>TRANSPORT</Category>
      </vspv:ClaimLinItem>
    </LinItems>
  </vspv:Claim>
</Claims>
</BillingFrame>

```

### Prozessfluss (vereinfacht)

1. **Buchung:** `billing.mode = COST_CARRIER` + `authorizationRef` + strukturierte **medicalOrdinance** (Kap. 5.2).
2. **Leistung:** Fahrtstatus/Ereignisse (Kap. 5.1) erzeugen **leistungszeitliche** Daten (Abhol-/Ankunftszeiten, ggf. Wartezeiten).
3. **Claim-Erstellung:** System erzeugt `Claim (OPEN)` inkl. Positionen (Transport/Zuschläge).
4. **Übermittlung:** `Claim` → `SUBMITTED`, Quittierung `ACKNOWLEDGED`.

5. **Prüfung:** Ergebnis APPROVED oder REJECTED (mit rejectionCode).
6. **Rechnung/Zahlung:** Invoice erzeugen, PAID nach Zahlungseingang; Settlement dokumentiert.

### API-Ausschnitte (REST)

**Create Claim** – POST /v1/billing/claims

```
{
  "bookingRef": "b_456",
  "costCarrierRef": "VSPV:CostCarrier:IK123456789",
  "authorizationRef": "M4-20250301-ABCD1234",
  "lineItems": [
    { "category": "TRANSPORT", "amount": 27.00, "currency": "EUR" },
    { "category": "ESCORT", "amount": 0.00, "currency": "EUR" }
  ],
  "meta": { "contractRef": "VSPV:Contract:IK123456789-2025" }
}
```

**Submit** – POST /v1/billing/claims/{id}:submit → status=SUBMITTED

**Acknowledge** (eingehend) – POST /v1/billing/claims/{id}:ack

**Decision** – POST /v1/billing/claims/{id}:decide (APPROVED|REJECTED, optional rejectionCode)

**Invoice** – POST /v1/billing/invoices (ein Claim oder Bündel)

### Validierung (normativ)

- **MUSS:** bookingRef, costCarrierRef, authorizationRef gültig und referenziert.
- **MUSS:** Summenkonsistenz ( $\sum \text{lineItems} = \text{invoice.total}$ ).
- **MUSS:** Währungen homogen (EUR).
- **MUSS:** Kein Claim-Duplicate: Idempotenz via (bookingRef, authorizationRef).
- **SOLL:** Zeit-/Entfernungswerte konsistent zu Ereignissen (Kap. 5.1).
- **DARF NICHT:** Freitext-Gesundheitsangaben.

## 6.3 Clearing und Erlösaufteilung (ÖPNV-Taxi, Pooling, Fördermodelle)

### Zweck

Dieses Unterkapitel standardisiert die **Erlösverteilung** zwischen beteiligten Akteuren (z. B. Verkehrsunternehmen, Anbieter/Unternehmer, Plattform, Kommune/Fördertopf) auf Basis transparenter Regeln.

### Grundsätze (normativ)

- **MUSS:** Clearing basiert auf **deklarativen Regeln** (Split-Regeln) in einem **ClearingAgreement**.

- **MUSS:** Jeder abrechenbare Leistungssatz erzeugt **Settlement-Einträge** pro Partner.
- **SOLL:** Abweichungen (Rabatt, Co-Payment) sind **pro Buchung** nachvollziehbar.

### Datenobjekte (VSPV-Clearing-Erweiterung im BillingFrame)

- **ClearingAgreement** (Regelwerk)
- **SplitRule** (Prozentsatz, Fixanteil, Cap/Floor)
- **Settlement** (Abrechnung pro Partner, Periodenbündel)
- **ClearingReport** (maschinenlesbare Auswertung)

### Beispiel (Split-Regeln):

```
<vspv:ClearingAgreement id="VSPV:Clear:TaxiBus-2025" version="1.0">
  <Rules>
    <vspv:SplitRule id="VSPV:Rule:OpTaxi">
      <AppliesTo>TRANSPORT</AppliesTo>
      <PartnerRef ref="VSPV:Partner:Taxi-Unternehmer"/>
      <Percent>85</Percent>
    </vspv:SplitRule>
    <vspv:SplitRule id="VSPV:Rule:Platform">
      <AppliesTo>TRANSPORT</AppliesTo>
      <PartnerRef ref="VSPV:Partner:Plattform"/>
      <Percent>15</Percent>
    </vspv:SplitRule>
  </Rules>
</vspv:ClearingAgreement>
```

### Settlement-Eintrag (JSON):

```
{
  "bookingRef": "b_456",
  "lineItemRef": "VSPV:CLI:1",
  "partnerRef": "VSPV:Partner:Taxi-Unternehmer",
  "amount": 22.95,
  "currency": "EUR",
  "period": "2025-03"
}
```

- **Validierung & Reports**
- **MUSS:**  $\sum \text{partner amounts} = \text{lineitem.amount}$ .
- **SOLL:** Periodenberichte (`/v1/billing/clearing/reports?period=YYYY-MM`) maschinenlesbar (CSV/JSON).
- **MUSS:** Jede Anpassung (Kulanzen, Stornos) erzeugt **Korrekturbuchungen** (negative Settlement-Einträge).

## 6.4 Fehlercodes, Idempotenz, Audit (kapitelübergreifend)

**Idempotenz (Schreiben):** `Idempotency-Key` + natürliche Schlüssel (z. B. `authorizationRef`, `bookingRef`).

**Fehlercodes (Ergänzung):**

- `409 CONFLICT` – Duplikat (Claim/Buchung/Clearing-Satz bereits vorhanden).
- `412 PRECONDITION_FAILED` – fehlende Genehmigung/ungültiger Zustand.
- `422 UNPROCESSABLE_ENTITY` – Summen-/Regelinkonsistenz.

**Audit (MUSS):** Jede finanzrelevante Mutation wird mit `correlationId`, Zeitstempel, Akteur, Altdaten/Neudaten protokolliert.

## 6.5 Mappingtabelle Kapitel 6

Fachliche Anforderung	Technische Datenstruktur / API	Konventionen
Festpreis Taxi	FareFrame → FareStructure + PriceGroup	EUR, 2 Dez., Gültigkeit
Zeit-/wegabhängiger Tarif	FareStructure + KeyValue-Parameter	BASE_FEE, PER_KM, PER_MIN, Zuschläge
Preisfixierung bei Buchung	quote in /quotes & /bookings	validUntil, Betrag brutto
Zahlung in App/Karte/Bar/Rechnung/Kostenträger	payment.mode	capture-Regel, Receipt optional
Kostenträger-Abrechnung	BillingFrame → Claim/LinItem/Invoice	bookingRef, costCarrierRef, authorizationRef
Genehmigungsprüfung (aus Kap. 5.2)	Claim-Validierung	APPROVED/NOT_REQUIRED
Clearing zwischen Partnern	ClearingAgreement + SplitRule + Settlement	Summe Partner = Position
Idempotenz und Dubletten	Header + natürliche Schlüssel	409 bei Wiederholung
Audit/Protokollierung	Änderungslog	Pflicht für finanzrelevante Aktionen

---

Kapitel 6 führt eine **saubere Trennung** ein: **Tarif/Preislogik** im FareFrame, **Zahlungsabsicht** in der **Buchung**, **Drittkostenträger-Abrechnung** in einem **VSPV-BillingFrame** (Erweiterung) mit **Claims/Invoices/Settlement** und **Clearing**-Regeln. Alle

Objekte referenzieren die in Kap. 3 definierten IDs und beachten die Feld-/Zeichensatzregeln aus Kap. 3.3.

## 7 Nutzermanagement und Authentifizierung

Dieses Kapitel spezifiziert Identitäten, Rollen-/Rechtekonzepte und Authentifizierungsverfahren für alle Interaktionen mit den in Kap. 3–6 beschriebenen Ressourcen (z. B. `/availability`, `/quotes`, `/bookings`, `/billing`, `/medical/ordinances:parse`). Ziel ist ein einheitliches, interoperables Sicherheitsmodell für Endkunden, Kiosksysteme, Organisationen (Arztpraxis/Krankenhaus), Betreiber/Unternehmer und Kostenträger.

### Grundsätze (normativ):

- **MUSS:** Autorisierung erfolgt rollen- und scope-basiert (RBAC + OAuth2-Scopes).
- **MUSS:** Identitäten werden über OIDC/JWT repräsentiert (Endnutzer, Kiosk, Organisationskonto, System-zu-System).
- **MUSS:** Delegierte Buchungen (z. B. Klinik → Patient) werden technisch kenntlich gemacht (`actingOnBehalfOf`).
- **SOLL:** SSO zwischen ÖPNV-Frontends und externen Buchung-/Abrechnungsdiensten per OIDC Authorization Code Flow (mit PKCE).
- **DARF NICHT:** Speicherung sensibler Freitexte (medizinische Details) im Identitätskontext; nur kodierte Flags gemäß Kap. 5.2/6.

### 7.1 Benutzerrollen und Rechte

#### 7.1.1 Rollen (RBAC)

Die folgenden **Rollen** sind systemweit definiert. Rollen gewähren **Baseline-Berechtigungen**; **Scopes** schränken/erweitern den Zugriff auf API-Ebenen.

Rolle	Beschreibung	Typische Aktionen
PASSENGER	Endkunde/Fahrgast	Verfügbarkeit prüfen, Angebote anfragen, buchen, stornieren, Quittung abrufen
KIOSK_DEVICE	Registriertes Kiosksystem (Bahnhof, Klinik, Praxis)	Verfügbarkeit/Angebote abfragen, Buchungen im Namen von Endkunden anlegen (delegiert)
ORG_STAFF	Autorisiertes Personal einer Organisation (Arztpraxis/Krankenhaus)	Delegierte Buchungen (Krankenfahrten), QR-Import Muster-4, Statusabrufe eigener Buchungen

Rolle	Beschreibung	Typische Aktionen
OPERATOR_DISPATCH	Betreiber/Unternehmer/Disposition	Fahrtenstatus pflegen, Fahrzeug-/Ressourcenstatus liefern, Buchungen annehmen/ablehnen gemäß Kap. 5.1
COST_CARRIER_AGENT	Kostenträger (Kasse/komm. Träger)	Claims/Invoices lesen, Entscheidungen/Acknowledgements liefern (Kap. 6.2)
PLATFORM_ADMIN	Plattformadministration	Mandanten-/Org-Verwaltung, Rollenvergabe, Schlüsselmanagement, Audits

**Hinweis:** Rollen sind **mandantenfähig**. Rechte gelten nur innerhalb des zugeordneten **Organisationskontexts**.

### 7.1.2 Scopes (feingranular)

Scopes werden als **OAuth2-Scope-Strings** vergeben und in Access-Tokens transportiert.

#### Lese/Schreib-Scopes (Auszug):

```
read:availability create:quote create:booking read:booking
cancel:booking
read:events write:events read:billing write:claim decide:claim
parse:ordinance read:org administer:org administer:platform
```

#### Beispiel Rollenzuordnung (empfohlen):

- **PASSENGER:** read:availability create:quote create:booking read:booking cancel:booking
- **KIOSK\_DEVICE:** wie **PASSENGER** plus parse:ordinance (Muster-4-QR)
- **ORG\_STAFF:** parse:ordinance create:booking read:booking cancel:booking read:org
- **OPERATOR\_DISPATCH:** read:booking write:events read:events read:availability
- **COST\_CARRIER\_AGENT:** read:billing write:claim decide:claim
- **PLATFORM\_ADMIN:** administer:platform administer:org (restriktiv vergeben)

#### Normative Regeln:

- **MUSS:** Jeder API-Aufruf erfordert mindestens einen passenden Scope.
- **SOLL:** Scopes sind **Least-Privilege** konfiguriert; UI/Funktionen dürfen nur anbietbare Aktionen rendern.
- **DARF NICHT:** Es dürfen keine impliziten Rechte außerhalb der im Token enthaltenen Scopes angenommen werden.

### 7.1.3 Delegation („acting on behalf“)

Delegierte Buchungen (z. B. Klinik bucht für Patient) **MÜSSEN** im Token/Request modelliert werden.

#### JWT-Claims (Beispiel):

```
{
  "iss": "https://id.vspv.example",
  "sub": "VSPV:User:OrgStaff-4711",
  "aud": "vspv-api",
  "scope": "parse:ordinance create:booking read:booking",
  "role": "ORG_STAFF",
  "org": "VSPV:Org:Klinikum-Hamm",
  "obo": "VSPV:Passenger:PID-9a28",      // actingOnBehalfOf
  (pseudonym)
  "act": { "sub": "VSPV:User:OrgStaff-4711" }, // Actor nach IETF-Pattern
  "exp": 1735737600,
  "jti": "1c8f..."
}
```

#### Request-Kopf (alternativ/ergänzend):

```
X-Acting-On-Behalf-Of: VSPV:Passenger:PID-9a28
```

#### Regeln:

- **MUSS:** Jede delegierte Aktion enthält **obo** (pseudonymisierte Fahrgast-ID).
- **SOLL:** Die tatsächliche Identität des Patienten bleibt **pseudonymisiert**; Mapping liegt im geschützten Mandantenkontext (Kap. 7.3).

## 7.2 Authentifizierungsverfahren

### 7.2.1 Endnutzer & Kiosk: OIDC Authorization Code (mit PKCE)

- **Flow:** Authorization Code + **PKCE** (S256) für Apps/Web.
- **Tokens:** Kurzlebige **Access-Tokens** (empf. 10–15 Min.), **Refresh-Tokens** (max. 24 h; rotierend).
- **Claims (Minimum):** **sub**, **role**, **scope**, **org** (bei Kiosk/Org-Kontext), **obo** (falls Delegation).
- **Sitzung:** Serverseitige Session nur als UI-Bequemlichkeit; **nicht** als Autorisationsquelle.

#### Kiosk-Spezifik:

- **Geräte-Enrollment:** mTLS oder signierter Gerätecode → Ausstellung eines **Device-Credentials** (Bindung an Standort/StopPlaceRef).
- **Gerätetoken:** Kurzlebig (≤ 12 h), nur Scopes **read:availability create:quote create:booking parse:ordinance read:booking cancel:booking**.

- **Rate-Limits:** Pro Gerät/Org konfigurierbar (Missbrauchsvermeidung).

## 7.2.2 System-zu-System: mTLS-Client-Credentials

- **Flow:** OAuth2 Client Credentials **mit mTLS** (Client-Zertifikate; SAN enthält org/systemId).
- **Lebensdauer:** Access-Tokens ≤ 10 Min., **keine** Refresh-Tokens.
- **Verwendung:** Disposition (Kap. 5.1), Abrechnung (Kap. 6.2/6.3), Batch-Importe.
- **Key-Rotation:** Zertifikats- und JWK-Rotation **MUSS** unterstützt werden.

### Beispiel-Token (gekürzt):

```
{
  "sub": "VSPV:System:Dispatcher-OB",
  "role": "OPERATOR_DISPATCH",
  "scope": "read:booking write:events read:events",
  "org": "VSPV:Org:Taxi-OB"
}
```

## 7.2.3 SSO-Handshakes (Verweis auf Kap. 4.3)

- **Richtung:** ÖPNV-Plattform ↔ Externer Buchungs-/Abrechnungsdienst.
- **Verfahren:** OIDC **Authorization Code mit PKCE; pairwise subject identifiers** zur Korrelationstrennung.
- **Parameter (MUSS):** state, nonce, exact redirect URIs; Token-Austausch serverseitig.
- **Session-Weitergabe:** Nur per Token-Austausch; **keine** Weitergabe von Passwörtern/Cookies.
- **Abmeldung:** Front-/Back-Channel Logout **SOLL** unterstützt werden.

## 7.3 Datenschutz und Datenminimierung

### 7.3.1 Prinzipien

- **MUSS: Datensparsamkeit** – nur für Buchung/Erfüllung/Abrechnung erforderliche personenbezogene Daten verarbeiten.
- **MUSS: Trennung** von Identitätsdaten und Buchungs-/Leistungsdaten; Verknüpfung über **pseudonyme Schlüssel**.
- **DARF NICHT:** Speicherung von medizinischen Freitexten; ausschließlich **kodierte Indikatoren** (Kap. 5.2).
- **SOLL: Pseudonymisierung** (PassengerID), sektorierte sub-Werte (pairwise) im SSO.
- **MUSS: Rechte-/Rollenprüfung** vor allen lesenden/schreibenden Aktionen.

### 7.3.2 Identitäts- und Personenobjekte (Minimalsatz)

#### Endkunde (Profile-Ausschnitt):

```
{
  "passengerId": "VSPV:Passenger:PID-9a28",
  "contact": { "phone": "+49...", "email": "...@..." }, // optional
  "preferences": { "wheelchair": true } // nur kodierte Flags
}
```

#### Organisationskonto (Praxis/Klinik):

```
{
  "orgId": "VSPV:Org:Klinikum-Hamm",
  "users": [
    { "userId": "...", "role": "ORG_STAFF", "login": "..." }
  ]
}
```

#### Regeln:

- **MUSS:** `passengerId` ist **pseudonym**; Klardaten getrennt (verschlüsselt, separat).
- **SOLL:** Kontaktfelder optional; für Kiosk-Buchungen kann eine erreichbare Telefonnummer abgefragt werden.
- **MUSS:** Aufbewahrung/Löschung konfigurierbar (Policy); Protokollierung von Zugriffen (Audit).

### 7.3.3 Delegationsnachweis (Krankenfahrten)

- **MUSS:** Bei `ORG_STAFF`-Buchungen für Patienten wird **Delegation** protokolliert (`obo`, `org`, `userId`, Zeitstempel).
- **SOLL:** Verknüpfung zum **Muster-4**-Datensatz (`authorizationRef`, `medicalOrdinance`) ohne Klardiagnose.
- **DARF NICHT:** Weitergabe von personenbezogenen Daten an unbeteiligte Rollen (z. B. `OPERATOR_DISPATCH` erhält nur erforderliche Transportmerkmale).

### 7.3.4 Audit & Nachvollziehbarkeit

Jede sicherheits-/abrechnungsrelevante Änderung wird mitgeführt:

```
{
  "auditId": "a_7f2...",
  "when": "2025-09-01T09:12:03+02:00",
  "actor": { "sub": "VSPV:User:OrgStaff-4711", "org": "VSPV:Org:Klinikum-Hamm", "obo": "VSPV:Passenger:PID-9a28" },
  "action": "BOOKING_CREATE",
  "resource": "/v1/bookings",
  "result": "SUCCESS",
  "correlationId": "c_3ab..."
}
```

}

- **MUSS:** Audit-Logs sind **unveränderlich** (WORM-fähig) und mandantensepariert.
- **SOLL:** Export-/Auskunftsprozesse für Betroffenenrechte vorgesehen.

## 7.4 Tokenformate, Lebensdauern, Schlüssel

**JWT-Signatur:** JWS (RS256/ES256). **JWKs** veröffentlicht; Rotation **SOLL** halbjährlich oder anlassbezogen erfolgen.

### Lebensdauern (empfohlen):

- Access-Token Endnutzer/Kiosk: 10–15 Min.
- Refresh-Token Endnutzer/Kiosk: ≤ 24 h (Rotationspflicht).
- System-Tokens (mTLS Client Credentials): ≤ 10 Min., **ohne** Refresh.  
**Idempotenz:** Schreibaufrufe **MÜSSEN** **Idempotency-Key** unterstützen (siehe Kap. 5/6).  
**Clock Skew:** Toleranz ± 2 Min. bei Tokenprüfung.

## 7.5 Mappingtabelle Kapitel 7

Fachliche Anforderung	Technische Umsetzung	Konventionen
Rollenbasiertes Rechemodell	RBAC + OAuth2-Scopes	Least-Privilege, Mandantenbezug org
Delegierte Buchung (Klinik → Patient)	JWT-Claim obo + Header X-Acting-On-Behalf-Of	Pseudonyme IDs, Audit Pflicht
Endnutzer/Kiosk-Login	OIDC Code Flow + PKCE	Access ≤ 15 Min., Refresh ≤ 24 h
System-zu-System-Zugriff	OAuth2 Client Credentials + <b>mTLS</b>	Tokens ≤ 10 Min., Zert-Rotation
SSO zu externen Diensten	OIDC, pairwise sub, state/nonce	Verweis Kap. 4.3; kein Passwort-Sharing
QR-Import Muster-4	Scope parse:ordinance	Keine Freitexte; Hash/Idempotenz
Datenschutz	Pseudonymisierung, Trennung Klardaten	Nur kodierte Flags; Löschpolitik
Audit	Unveränderliche Logs, correlationId	Export für Betroffenenrechte

Kapitel 7 definiert ein **klares, interoperables Identitäts- und Autorisierungsmodell:** RBAC + Scopes, OIDC/OAuth2 für Menschen und Geräte, mTLS für Systeme, delegierte Buchungen transparent über obo, strikte Datenminimierung und revisions sichere Audits.

Es schließt nahtlos an die Prozess- und Datenobjekte aus Kap. 3–6 an und bereitet die Security-Vertiefung in Kap. 8 vor.

## 8 Sicherheits- und Zugriffskonzepte

Dieses Kapitel definiert verbindliche technische und organisatorische Sicherheitsmaßnahmen für alle in Kap. 3–7 beschriebenen Prozesse und Schnittstellen. Es ergänzt die Authentifizierungs- und Rollenmodelle aus Kap. 7 um **Autorisierung**, **Protokollierung/Monitoring** sowie **Datenschutz & Datensicherheit**. Ziel ist eine konsistente, mandantenfähige und revisionssichere Sicherheitsarchitektur über den gesamten Lebenszyklus von Buchung, Durchführung und Abrechnung.

### 8.1 Datenautorisation und Zugriffskontrolle

#### 8.1.1 Grundprinzipien (normativ)

- **MUSS:** Zugriffskontrolle nach dem **Least-Privilege-Prinzip** (Rollen + Scopes, vgl. Kap. 7).
- **MUSS: Mandantentrennung** (Organisationskontext `org`) auf allen Ebenen: Datenhaltung, Caches, Queues, Logs.
- **MUSS: Transportverschlüsselung** (TLS 1.3; HSTS aktiviert) für alle externen Schnittstellen.
- **MUSS:** System-zu-System-Kommunikation via **mTLS** (gegenseitige Zertifikatsprüfung, SAN enthält `org/systemId`).
- **MUSS:** Durchsetzung der **Scope-Prüfung** pro Endpoint & HTTP-Methode; keine impliziten Rechte.
- **SOLL:** Ergänzende **ABAC**-Filter (Attribut-basiert), z. B. `org`, `obo` (delegierte Buchung), `role`, `region`.
- **DARF NICHT:** Speicherung von Zugangsdaten im Klartext; Secrets ausschließlich über KMS/HSM.

#### 8.1.2 Autorisierungsregeln pro API-Bereich (Auszug)

API-Bereich	Operation	Erforderliche Scopes	Zusätzliche Bedingungen
/availability	GET	read:availability	org muss zum Mandanten des Tokens passen
/quotes	POST	create:quote	ggf. Rate-Limit pro deviceId/org
/bookings	POST	create:booking	bei Delegation: obo erforderlich (Kap. 7)

API-Bereich	Operation	Erforderliche Scopes	Zusätzliche Bedingungen
/bookings/{id}	GET	read:booking	Zugriff nur auf Buchungen der eigenen org <b>oder</b> des eigenen sub
/events (Status)	POST	write:events	nur OPERATOR_DISPATCH/mTLS-Clients
/billing/claims	POST	write:claim	COST_CARRIER_AGENT oder autorisierte Plattform
/billing/claims/{id}:decide	POST	decide:claim	Entscheidung nach Vertrags-/Regelwerk (Kap. 6)

### Normative Ergänzungen

- **MUSS:** Token-Validierung inkl. Signatur, `aud`, `iss`, Ablauf (`exp`), Nonce/State (bei OIDC), **Replay-Schutz** über `jti`.
- **SOLL: Idempotency-Keys** (Kap. 6) für alle mutierenden Operationen; Wiederholungen → `409 CONFLICT`.
- **MUSS: Input-Validierung** (XSD/JSON-Schema), Größenlimits, Anti-XXE (XML-Parser ohne externe Entitäten).
- **SOLL: IP-Allowlisting** optional für Partner-Integrationen; anwendungsseitiges **Rate-Limiting** und **DoS-Schutz**.

#### 8.1.3 Schlüssel- & Geheimnisverwaltung

- **MUSS:** Zentrales **KMS/HSM** für JWT-Keys, mTLS-Zertifikate, Datenbankschlüssel; **Rotation** mind. halbjährlich oder anlassbezogen.
- **MUSS:** Trennung von **Betriebsgeheimnissen** (PSP-Tokens, SSO-Client-Secrets) je Mandant/Umgebung; kein Geheimnis in Code/Images.
- **SOLL:** SBOM/Dependency-Pinning; Build-Provenance signiert (Supply-Chain-Härtung).

#### 8.1.4 Datenklassifikation & Zugriffspfade

- **Klassen:** `PUBLIC` | `INTERNAL` | `CONFIDENTIAL` | `PERSONAL` | `HIGHLY_SENSITIVE`.
- **Regeln:** Zugriff auf `PERSONAL/HIGHLY_SENSITIVE` ausschließlich mit **personenbezogenem** Scope und **Zweckbindung**; strikte Protokollierung (Kap. 8.2).
- **Pflicht:** Alle Export-/Reporting-Schnittstellen implementieren **Feld-Maskierung** und **Aggregationsspannen**.

## 8.2 Protokollierung und Monitoring

### 8.2.1 Audit-Logging (normativ)

- **MUSS:** Unveränderliche **Audit-Logs** (WORM-fähig) für sicherheits-/finanzrelevante Aktionen: Authentifizierung, Rollen-/Scope-Änderungen, Buchungs-Create/Update/Cancel, Event-Writes (Status), Claim/Invoice/Settlement-Mutationen.
- **MUSS:** Standardfelder: timestamp, actor.sub, actor.org, role, scopes, obo?, action, resource, method, status, correlationId, idempotencyKey?, remoteAddr, mTLS.fingerprint?.
- **DARF NICHT:** Speicherung von Passwörtern, vollständigen Zahlungs-Tokens oder medizinischen Freitexten in Logs.
- **SOLL: Zeitquelle** über NTP/PTP synchronisiert (Drift  $\leq \pm 2$  Min.; Kap. 7 Tokenprüfung berücksichtigt Skew).

#### Beispiel-Audit-Eintrag (JSON):

```
{
  "ts":"2025-09-01T09:12:03+02:00",
  "actor":{"sub":"VSPV:User:OrgStaff-4711","org":"VSPV:Org:Klinikum-
Hamm","role":"ORG_STAFF","obo":"VSPV:Passenger:PID-9a28"},
  "action":"BOOKING_CREATE",
  "resource":"/v1/bookings",
  "status":"SUCCESS",
  "correlationId":"c_3ab...",
  "idempotencyKey":"idem_9f1..."
}
```

### 8.2.2 Betriebs-Monitoring & SIEM

- **MUSS:** Zentrale **Metriken:** Verfügbarkeit (SLO  $\geq 99,9$  %), p95-Latenz je API, Fehlerraten (4xx/5xx), AuthN/AuthZ-Fehler, Queue-Tiefe, Event-Backlogs, PSP-Antwortzeiten.
- **SOLL: Anomalieerkennung** (z. B. sprunghafte create:booking in kurzer Zeit, ungewöhnliche Claim-Ablehnungsquote).
- **MUSS:** Integration in **SIEM** (Korrelation von Security-Events), Alarme mit Schwellwerten/Heuristiken.
- **SOLL: Distributed Tracing** (W3C TraceContext), Weitergabe von correlationId über Systemgrenzen (Kap. 5/6).

### 8.2.3 Incident-Response & Verwundbarkeiten

- **MUSS:** Definierte **Schweregrade** (SEV-1 bis SEV-4) mit Reaktionszeiten, Eskalationspfaden und Kommunikationsplan (inkl. Partner-Benachrichtigung).

- **MUSS: Vulnerability-Management:** kontinuierliches Scanning (Container, Dependencies), CVE-Bewertung, Patch-SLA nach Kritikalität.
- **SOLL:** Regelmäßige **Penetrationstests** und **Red-Team-Übungen**; Findings werden getrackt bis zur Abstellung.

#### 8.2.4 Aufbewahrung

- **Audit-Logs:** min. 12 Monate (empfohlen 24), danach **revisionsicher archivieren** oder datenschutzkonform löschen/anon.
- **Metriken/Traces:** min. 90 Tage (Fein-), 12 Monate (Aggregat).
- **Partner-Spezifika:** Vertraglich abweichende Fristen sind in der Datenhaltungs-Policy zu dokumentieren.

### 8.3 Datenschutz und Datensicherheit

#### 8.3.1 Kryptografie & Speichersicherheit

- **MUSS: Verschlüsselung im Transit** (TLS 1.3) und **im Ruhezustand** (AES-256-GCM o. ä.) für Datenbanken, Suchindizes, Backups.
- **SOLL: Feld-/Spaltenverschlüsselung** (z. B. Telefonnummern, E-Mail, Zahlungstoken), Schlüsselverwaltung über KMS/HSM.
- **MUSS: Pseudonymisierung** von Fahrgast-IDs (PassengerID), Trennung Klardaten ↔ Buchungs-/Leistungsdaten (separate Datenspeicher, getrennte Zugriffsrollen).
- **SOLL: Salting/Hashing** für Identifikatoren, die in externen Systemen referenziert werden (Kollisionsschutz).

#### 8.3.2 Datenminimierung & Zweckbindung

- **MUSS:** Erhebung ausschließlich **zweckgebundener** Daten (Buchung/Erfüllung/Abrechnung).
- **DARF NICHT:** Speicherung medizinischer Freitexte; nur **kodierte Flags/Referenzen** gemäß Kap. 5.2.
- **SOLL: Privacy by Default** in UIs/APIs (z. B. opt-in für Quittungsversand per E-Mail; Maskierung im Self-Service).

#### 8.3.3 Löschung, Sperrung, Aufbewahrung

- **MUSS:** Konfigurierbare **Lösch- und Sperrfristen** pro Datenklasse; technische Durchsetzung (geplante Jobs, „hard delete“ oder starke Anonymisierung).
- **MUSS:** Historisierung (Kap. 3.3) bleibt für **nicht-personenbezogene** Teile erhalten; personenbezogene Bezüge werden entfernt/entkoppelt.
- **SOLL:** Nachvollziehbare **Löschprotokolle** im Audit-Log (ohne Offenlegung gelöschter Inhalte).

### 8.3.4 Sicherer Datenaustausch & Eingabevalidierung

- **MUSS: Schema-Validierung** (NeTeX/VDV-XSD, JSON-Schema); Abweisung nicht valider Payloads.
- **MUSS: Schutz gegen XXE, Entity-Expansion, Deserialization-Angriffe;** Deaktivierte DTDs.
- **MUSS: Längen-/Zeichenprüfungen** gemäß Kap. 3.3 (IDs, Namen, Freitexte); harte **Payload-Größenlimits** pro Endpoint.
- **SOLL: Content-Security-Policy** und **WAF-Regeln** für öffentliche Endpunkte; Ratelimits pro `sub/org/device`.

### 8.3.5 Backups, Wiederanlauf & Resilienz

- **MUSS: Verschlüsselte Backups** (Off-Site, getrennte Schlüssel), Integritätsprüfungen, **regelmäßige Restore-Tests**.
- **SOLL: Zielwerte RPO ≤ 15 Min., RTO ≤ 2 Std.** für Kernsysteme (Buchung, Events, Billing).
- **MUSS: Chaos-/Failure-Drills** (z. B. Datenbank-Failover, Queue-Stau) mindestens halbjährlich.

### 8.3.6 Secure-Development-Lifecycle (SDL)

- **MUSS: Sicherheitsanforderungen in Design-Reviews, Threat-Modeling** (z. B. STRIDE) pro Schnittstelle.
- **MUSS: Static/Dynamic Application Security Testing (SAST/DAST)** in der CI/CD-Pipeline; **laC-Scans**.
- **SOLL: Secrets-Scanning** für Repos/Images; **4-Augen-Prinzip** bei sicherheitsrelevanten Änderungen.
- **MUSS: Versionierte OpenAPI/Schema-Artefakte;** Breaking-Changes nur in Major-Releases (Versionierung Kap. 3.3/3.9).

## 8.4 Mappingtabelle Kapitel 8

Anforderung	Umsetzung	Nachweis/Prüfung
Autorisierung nur mit passenden Scopes	Enforcer vor Business-Logik, Tabellen wie 8.1.2	automatisierte API-Tests, Pen-Tests
Mandantentrennung	org-Kontext in Token, Datenpartitionierung	Architektur-Review, Versuch unberechtigter Zugriffe
mTLS S2S	Client-Zertifikate, SAN-Prüfung, Kurzzeittokens	TLS-Handshake-Logs, Zert-Inventar

<b>Anforderung</b>	<b>Umsetzung</b>	<b>Nachweis/Prüfung</b>
Unveränderliche Audits	WORM-Speicher, vollständiger Audit-Satz	Audit-Export, Integritäts-Checks
DoS/Abuse-Schutz	Rate-Limits, WAF, Backoff	Lasttests, SIEM-Alerts
Verschlüsselung	TLS 1.3, AES-256-at-rest, Feldverschlüsselung	KMS-Policies, Krypto-Reports
Datenminimierung	Pseudonyme IDs, keine Freitexte	Schema-Validierung, Dateninventar
Wiederanlauf	Backups, RPO/RTO-Ziele, Restore-Drills	Protokolle, Drill-Berichte
SDL	SAST/DAST/IaC-Scans, 4-Augen CI/CD-Berichte, Release-Gate	

## 9 Anhang A – ID-/Namensraum-Registry

**Zweck (normativ):** Eindeutige, kollisionsfreie Identifikatoren gem. Kap. 3.2/3.3.

**MUSS:** Jeder vergebene **Namensraum (prefix)** ist registriert. **SOLL:** id\_pattern dokumentiert die ID-Struktur. **DARF NICHT:** Mehrfachvergabe desselben Präfixes.

**Spalten:** prefix | owner\_org | contact | scope | id\_pattern | notes

prefix	owner_org	contact	scope	id_pattern	notes
VSPV	VSPV- Plattform	<a href="mailto:tech@vspv.example">tech@vspv.example</a>	global	VSPV:{Objekttyp};{Schlüssel}	Kern- Namensraum
NRW	Aufgabenträger NRW	<a href="mailto:data@nrw.example">data@nrw.example</a>	regional	NRW:{Objekttyp};{Region}{-Nummer}	regionaler Bestand
OB	Stadt Oberhausen	<a href="mailto:dv@oberhausen.example">dv@oberhausen.example</a>	lokal	OB:{Objekttyp};{YYYY}{-laufend}	Lokale Pilotdaten
DIVA	Bestandswelt DIVA	<a href="mailto:diva@provider.example">diva@provider.example</a>	migration	DIVA:{Objekttyp};{AltID}	nur Migration/Mapping
WEST F	WestfalenTarif	<a href="mailto:info@wt.example">info@wt.example</a>	tarif	WESTF:{Objekttyp};{Code}	Tarifobjekte

**Pflege & Änderung:** Änderungen nur durch Registry-Owner; Aufnahme/Änderung via Pull-Request im Annex-Repo.

## 10 Anhang B – KeyValue-Schlüssel-Registry

**Zweck (normativ):** Einheitliche Zusatzattribute für <keyList> (Kap. 3.6).

**MUSS:** Schlüssel **ASCII-Uppercase** mit \_ ([A-Z0-9\_]+). **SOLL:** Wohldefinierte Datentypen. **DARF**

**NICHT:** Umlaute/Leerzeichen in key.

**Spalten:** key | datatype | domain\_object | allowed\_values | required\_if | version | status | notes

key	datatype	domain_object	allowed_values	required_if	version	status	notes
FESTPREIS	decimal(9,2)	FlexibleServiceJourney, FareProduct	>= 0 (EUR)	pricing_model=fixed	1.0	stable	Preis ohne Währungssymbol
WHEELCHAIR_ACCESS	boolean	VehicleType, Booking	true	false	transport_needed=wheelchair	1.0	stable
MAX_DETOUR_TIME	duration(ISO-8601)	FlexibleServiceJourney	PT0M...PT60M	pooling=true	1.0	stable	Max. zul. Umwegzeit
POOLING_ALLOWED	boolean	FlexibleServiceJourney	true	false	-	1.0	stable
BEFOERDERUNGSART	string	Service/Booking	PBefG\$47,PBefG\$49,Krankenfahrt	-	1.0	stable	ASCII-Schreibweise ohne Umlaut
LEGACY_ID	string	*	frei	Migration=ja	1.0	stable	Ursprungs-ID aus Altsystem
KOSTENTRAGER_ID	string	Booking/Billing	[A-Z0-9\-]{3,20}	payer=insurancance	1.0	<b>transitional</b>	in 6.2 strukturiert abbilden
AUTHORIZATION_CODE	string	Billing	frei	genehmigungspflichtig	1.0	<b>transitional</b>	Muster-4/Genehmigung, später strukturiert

**Hinweis:** Wo strukturierte Felder existieren (z. B. BillingFrame, Kap. 6.2), **SOLLEN** KeyValue-Schlüssel nur übergangsweise verwendet werden.

## 11 Anhang C – XML-Erweiterungen/Namensräume

**Zweck:** Verwendung von <Extensions> (Kap. 3.6).

**Spalten:** prefix | namespace\_uri | schema\_location | owner\_org | version | status

prefix	namespace_uri	schema_location	owner_org	version	status
vspv	urn:vspv:ext:1	/schema/vspv-ext-1.xsd	VSPV	1.0	stable

## 12 Anhang D – Feldgrößen-/Zeichen-Master

**Zweck (normativ):** Einheitliche Längen/Zeichensätze gem. Kap. 3.3.

**MUSS:** Ablehnung bei Verstößen; **SOLL:** lokale Vorvalidierung.

**Spalten:** path | max\_len | charset | required | rules

path	max_len	charset	required	rules
//@id	50	[A-Za-z0-9:_- ]	yes	keine Leer-/Steuerzeichen
//@version	20	[A-Za-z0-9:_- ]	yes	SemVer empfohlen
//Name/Text	70	UTF-8	yes(de)	ISO 639-1 language
//Description/Text	255	UTF-8	no	\n erlaubt; keine Emojis
//Code	20	[A-Za-z0-9]	no	keine Leerzeichen
//TelephoneAccess/PhoneNumber	20	[0-9+ ]	no	international (+49...)
//Email	100	[A-Za-z0-9@._-]	no	RFC-konform
//Url	255	ASCII	no	RFC-konform, keine Leerzeichen

## 13 Anhang E – API-Endpunkte × Scopes-Matrix

**Zweck (normativ):** Durchsetzung AuthZ (Kap. 7).

**MUSS:** Passender Scope pro Operation; **SOLL:** Idempotency-Key für Schreibzugriffe.

**Spalten:** method | path | purpose | required\_scopes | roles | idempotency | pii\_class

method	path	purpose	required_scopes	roles	idempotency	pii_class
GET	/v1/availability	Fahrzeug/ Slot- Verfügbar- keit	read:availability	PASSENGER,KIOSK,ORG_STAFF	–	none
POST	/v1/quotes	Angebote ermitteln	create:quote	PASSENGER,KIOSK,ORG_STAFF	empfohlen	PERSONAL
POST	/v1/bookings	Buchung anlegen	create:booking	PASSENGER,KIOSK,ORG_STAFF	<b>erforderlich</b>	PERSONAL
GET	/v1/bookings/{id}	Buchung lesen	read:booking	PASSENGER,KIOSK,ORG_STAFF,OPERATOR_DISPATCH	–	PERSONAL
POST	/v1/bookings/{id}:cancel	Buchung stornieren	cancel:booking	PASSENGER,KIOSK,ORG_STAFF	<b>erforderlich</b>	PERSONAL
POST	/v1/events	Status/Ereignisse schreiben	write:events	OPERATOR_DISPATCH	<b>erforderlich</b>	INTERNAL
POST	/v1/billing/claims	Leistungsanspruch melden	write:claim	COST_CARRIER_AGENT	<b>erforderlich</b>	PERSONAL/HIGHLY_SENSITIVE
POST	/v1/billing/claims/{id}:decide	Entscheidung/Kasse	decide:claim	COST_CARRIER_AGENT	<b>erforderlich</b>	PERSONAL
POST	/v1/medical/orphanances:parse	Muster-4- QR einlesen	parse:orphanance	KIOSK,ORG_STAFF	empfohlen	HIGHLY_SENSITIVE

## 14 Anhang F – Fehler-/Reason-Codes

**Zweck (normativ):** Einheitliche Serverantworten.

**MUSS:** Stabiler error\_code, klare message\_template, Hinweis zur Abhilfe.

**Spalten:** error\_code | http\_status | category | message\_template | remediation | retrieable

error_code	http_status	category	message_template	remediation	retrieable
UNAUTHORIZED	401	auth	Nicht angemeldet/Token fehlt.	Login/OIDC-Flow starten.	yes
INVALID_SCOPE	403	authz	Fehlender Scope: {scope}.	Rechte/Scopes prüfen.	no
VALIDATION_ERROR	422	validation	Feld {field}: {reason}.	Payload korrigieren.	no
RATE_LIMITED	429	throttle	Ratelimit erreicht.	Backoff/Retry-After beachten.	yes
BOOKING_DUPLICATE	409	business	Idempotency-Key bereits verwendet.	Status prüfen/neuen Key nutzen.	no
AVAILABILITY_EXHAUSTED	409	business	Keine Kapazität in {window}.	Zeitfenster/Anzahl anpassen.	no
ORDINANCE_INVALID_QR	422	validation	Verordnung (QR) ungültig.	Neues Bild/Original scannen.	yes
PAYMENT_DECLINED	402	payment	Zahlung abgelehnt: {reason}.	Zahlungsart wechseln.	maybe
CLAIM_MISSING_DOCS	400	billing	Nachweise fehlen: {list}.	Dokumente ergänzen.	yes
CLAIM_REJECTED_RULE	409	billing	Regelverstoß: {ruleId}.	Prüfen/Begründung liefern.	no

## 15 Anhang G – Buchungsstatus-/Event-Matrix

**Zweck (normativ):** Einheitlicher Lebenszyklus und Ereignisemission (Kap. 5/6).

**MUSS:** Nur erlaubte Transitionen; **SOLL:** Ereignisse idempotent.

**Spalten:** state | allowed\_transitions | emitted\_event | producer | sla\_target

state	allowed_transitions	emitted_event	producer	sla_target
REQUESTED	→ CONFIRMED, → REJECTED	booking.requested	Plattform	sofort
CONFIRMED	→ ASSIGNED, → CANCELLED	booking.confirmed	Plattform	≤ 30 s
ASSIGNED	→ IN_PROGRESS, → CANCELLED	booking.assigned	Betreiber/Disposition	≤ 10 min
IN_PROGRESS	→ COMPLETED, → CANCELLED	booking.started	Betreiber/Disposition	–
COMPLETED	–	booking.completed	Betreiber/Disposition	–
REJECTED	–	booking.rejected	Plattform/Betreiber	Grundcode Pflicht
CANCELLED	–	booking.cancelled	Plattform/Betreiber	Grundcode Pflicht

**Stornogründe (Auszug, normativ):** CUSTOMER\_REQUEST, NO\_SHOW, OPERATOR\_CANCELLED, MEDICAL\_CONFLICT, PAYMENT\_FAILED.

## 16 Anhang H – FareProducts-/Tarif-Katalog

**Zweck:** Klare Tarifmodelle (Kap. 3.4/6).

**Spalten:** code | name | pricing\_model | neTex\_element | combinable\_with | notes

code	name	pricing_model	neTex_element	combinable_with	notes
TAXI_METERED	Taxi Taxameter	metered	FareProduct/PriceUnit	-	Zuschläge separat
TAXI_FIXED_ZONE	Taxi Festpreis Zone	fixed	FareProduct/PriceGroup	Bus/RE Kombi	Kap. 4.1/6.1

## 17 Anhang I – Kostenträger-Datenfelder

**Zweck (normativ):** Einheitliche Abrechnung Dritter (Kap. 6.2).

**MUSS:** Nur minimal erforderliche personenbezogene Felder; **SOLL:** Trennung Identität ↔ Leistungsdaten.

**Spalten:** field | type | source | required\_if | constraints | retention | privacy\_class | notes

field	type	source	required_if	constraints	retention	privacy_class	notes
payer_type	enum	Buchung	Kostenträger beteiligt	`INSURANCE	MUNICIPAL	COMPANY`	Abrechnung +X
insurance_id	string	Praxis/Kiosk	payer_type=INSURANCE	[A-Z0-9\-\_]{6,20}	Vertraglich	PERSONAL	Versicherten -Kennz.
authorization_code	string	Kasse/Verordnung	genehmigungspflichtig	frei (≤ 50)	Vertraglich	HIGHLY_SENSITIVE	Genehmigungs-ID
ordinance_ref	string	QR-Parser	Krankenfahrt	Hash/UID (≤ 50)	Abrechnung+X	HIGHLY_SENSITIVE	Referenz Muster-4
medical_flags	set	QR-Parser	bei Bedarf	`WHEELCHAIR	STRETCHER	ESCORT`	Abrechnung +X
payer_contract_id	string	Plattform	immer	[A-Z0-9\-\_]{3,30}	Vertragslaufz.	INTERNAL	Vertragsreferenz
claim_id	string	Plattform/Kasse	bei Einreichung	UUID	Abrechnung+X	INTERNAL	eindeutiger Anspruch
invoice_ref	string	Plattform	bei Rechnungsstellung	frei (≤ 50)	GoBD	INTERNAL	Rechnungsnummer

**Validierung (normativ):**

- Pflichtfelder gemäß required\_if.
- Keine Speicherung medizinischer Freitexte; nur kodierte Flags/Referenzen.
- Pseudonyme **PassengerID** anstelle von Klarnamen (siehe Kap. 7.3).

## 18 Anhang J – TRIAS-Nachrichten-Matrix

**Zweck:** Interop Echtzeit/Booking-Flows (Kap. 3.8/4.3).

**Spalten:** message | direction | trigger | expected\_response | latency\_target

message	direction	trigger	expected_response	latency_target
AvailabilityRequest	Auskunft → Betreiber	Angebotsabfrage	AvailabilityResponse	≤ 2 s
BookingRequest	Auskunft → Betreiber	Stufe-3-Buchung	BookingResponse	≤ 3 s
CancelRequest	Auskunft → Betreiber	Storno	CancelResponse	≤ 2 s

## 19 Anhang K – Prüf- & Konformitätsmatrix

**Zweck:** Nachweis „konform/nicht konform“ (Kap. 3.9/8).

**Spalten:** rule\_id | normative\_text | test\_id | validator | severity

rule_id	normative_text	test_id	validator	severity
R-ID-01	IDs ohne Leerzeichen, max. 50	T-ID-01	XSD+Custom	error
R-AUTH-05	Scope-Prüfung MUSS erfolgen	T-AUTH-0		

## 20 Anhang L – Beispiel-Feeds & Payloads

**Zweck:** Referenz/Onboarding (NeTeX `PublicationDelivery`, OpenAPI-Samples).

**Inhalt:** Minimal-Set für **Taxi-Festpreis**, **Taxi-Metered**, **delegierte Praxisbuchung**; je ein validiertes XML/JSON mit Kommentaren.

## 21 Verwaltungs- und Konformitätshinweise (für alle Anhänge)

- **Versionierung:** SemVer pro Anhang (Start 1.0.0).
- **Änderungsprozess:** Änderungsanträge via Pull-Request; Review durch Registry-Owner; Deprecation mit Übergangsfrist  $\geq 1$  Release.
- **Konformitätsprüfung:**
  - Schema-Validierung gegen Annex-Tabellen (CSV/JSON).
  - API-Tests (Scopes/Fehlercodes/Status-Flüsse) gegen Anhang E/F/G.
  - Statische Checks (IDs/Zeichen) gegen Anhang D.
  - Datenschutz-Checks gegen Anhang I.





