IVU Softwareentwicklung GmbH

Kreuzbergweg 1a  
93133 Burglengenfeld

www.ivu-software.de

TRudi 1.0

SoftwareDokumentation

Dokumentation zur Softwareprüfung nach WELMEC Guide 7.2

Versionshistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Autor | Änderungen |
| 0.1 | 08.09.2017 | C. Schröder | Initiale Version |
| 0.2 | 12.09.2017 | C. Schröder | Änderungen nach Abstimmung mit PTB |
| 1.0 | 14.09.2017 | C. Schröder | Version zur Prüfung |
| 1.1 | 09.10.2017 | T. Müller | Ergänzungen und Klarstellungen |

Inhaltsverzeichnis

[1. Softwarestruktur 1](#_Toc495435547)

[1 Frontend 2](#_Toc495435548)

[1.1 Starten und Beenden der Anwendung 2](#_Toc495435549)

[1.1.1 Implementierung Integritätscheck 2](#_Toc495435550)

[1.1.2 Start und Stop des Backends 3](#_Toc495435551)

[1.1.3 Identität der Software 3](#_Toc495435552)

[2 Benutzeroberfläche 4](#_Toc495435553)

[2.1 Übersicht der Ansichten 4](#_Toc495435554)

[2.2 Beschreibung der Ansichten 4](#_Toc495435555)

[2.3 Backend 5](#_Toc495435556)

[2.3.1 Übersicht der Komponenten des Backends 5](#_Toc495435557)

[2.3.2 Gerätekommunikation 6](#_Toc495435558)

[2.3.3 Datenmodell 8](#_Toc495435559)

[2.3.4 Transparenzfunktion 8](#_Toc495435560)

[2.3.5 Flussdiagramme für TAF-Plugin TAF-2 10](#_Toc495435561)

[2.3.6 Flussdiagramme für TAF-Plugin TAF-1 13](#_Toc495435562)

[3 Bezug zu Anforderungen nach PTB 50.8 14](#_Toc495435563)

[3.1 Anforderungen nach Kapitel 5.3 – Kundendisplay als Anzeige Applikation 14](#_Toc495435564)

[3.2 Anforderungen nach Kapitel 7 – Transparenzsoftware 19](#_Toc495435565)

[3.2.1 Sonstige Anforderungen 20](#_Toc495435566)

1. Softwarestruktur

Die TRuDI 1.0 Anwendung besteht aus Frontend und Backend Prozess.

Das Frontend basiert auf dem Electron Framework zur Umsetzung plattformunabhängiger Desktop Anwendungen. Electron selbst wiederum nutzt den Chromium Browser für das Rendern von den in HTML beschriebenen Dialogen. Das Frontend prüft die Integrität der Anwendung, startet das Backend und zeigt die vom Backend erzeugten Views an.

Das Backend ist eine in C# 7 entwickelte ASP.NET Core MVC Webapplikation. Die Anwendungslogik ist vollständig im Backend umgesetzt.

- WAN und HAN Kommunikation

- HAN-Plugins

- TAF-Plugins

- XML Verarbeitung

- Verwaltung Anwendungsstatus

- Erstellung der Views

Backend

- Start und Stop des Backendprozess

- Integritätsprüfung

- Anzeige der Views

Frontend

<<uses>>

Abbildung 1 Softwarestruktur

Verwendete Frameworks

Tabelle 1 Relevante in TruDi verwendete Frameworks

|  |  |  |
| --- | --- | --- |
| Framework | Version | Lizenz |
| [.NET Core](https://www.microsoft.com/net/download/core) | 2.0 | MIT |
| [ASP.NET Core](https://docs.microsoft.com/en-us/aspnet/core/) | 2.0 | Apache 2.0 |
| Serilog | 2.5 | Apache 2.0 |
| [Electron](https://electron.atom.io/) | 1.7.8 | MIT |
| Node.js | 7.9.0 | MIT |
| Chromium | 58.0.3029.110 | BSD |
| V8 | 5.8.283.38 | BSD |
| Bootstrap | 3.3.7 | MIT |

# Frontend

## Starten und Beenden der Anwendung

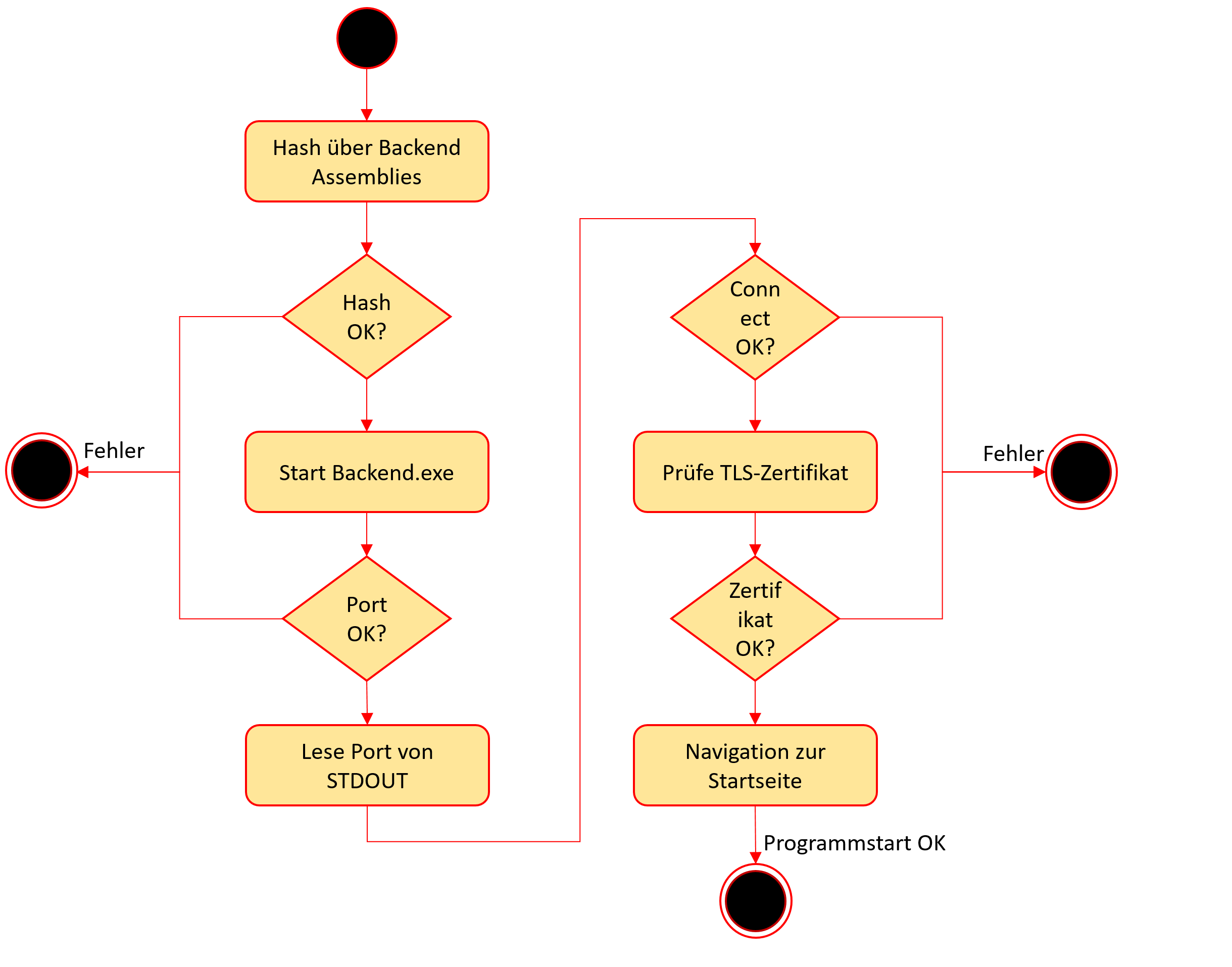


Abbildung 2 Lebenszyklus Frontend

### Implementierung Integritätscheck

Im Zuge der Erstellung einer Softwareversion[[1]](#footnote-1) wird für jede Datei des Backends ein RIPEMD-160 Hashwert berechnet. Der Hashwert wird zusammen mit dem relativen Pfad der Datei als 2-dimensionales Array in einer JSON-Datei abgespeichert. Diese Datei wird in das Frontend als Ressource in der Archivdatei app.asar hinterlegt. Das Frontend berechnet beim Start in dessen [*main-Routine*](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.Frontend/main.js) die Hashwerte über jede Datei des Backends und vergleicht anhand der Liste den berechneten Wert mit dem hinterlegten. Falls eine Datei fehlt oder ein Hashwert nicht übereinstimmt wird eine Fehlermeldung ausgegeben und die weitere Ausführung des Programms unterbunden.

Die HAN- und TAF-Adapter werden bei der Berechnung der Hashwerte ebenfalls berücksichtigt.

### Start und Stop des Backends

Wenn der Integritätscheck abgeschlossen ist, wird das Backend als Child-Process des Frontends gestartet.

Das Backend versucht für localhost einen freien Port zu finden und gibt diesen über STDOUT an den aufrufenden Prozess zurück. In der main()-Funktion des Frontends wird die Portnummer über die Ausgabemeldung des Backends eingelesen und anschließend die Startseite der Backend-Webapplikation aufgerufen. Dieses Verfahren soll sicherstellen, dass der Port unter dem das Backend erreichbar ist, nur dem Frontend bekannt ist.

Zusätzlich kommt zwischen Frontend und Backend kommt HTTPS zum Einsatz. Das Serverzertifikat des Backends wird dynamisch zur Laufzeit erzeugt. Als Common Name wird dabei der RIPEMD-160 Hashwert der Backend-Assembly eingetragen. Als Input für die Hashwertberechnung gehen alle Bytes derjenigen Assembly ein, welche die Klasse *Program* enthält. Das Frontend prüft beim Verbindungsaufbau das übermittelte Zertifikat und vergleicht den darin enthaltenen Common Name mit dem hinterlegten Erwartungswert.

Es ist ein Eventhandler eingerichtet um sicherzustellen, dass beim Beenden der Electron-Applikation gleichzeitig auch der Backend-Prozess beendet wird.

### Identität der Software

Im Dialog „Über TRuDI“ der Anwendung wird ein RIPEMD-160 Hashwert über die folgenden Frontend-Files des Installationsordners gebildet:

- TRuDI.exe

- app.asar

- electron.asar

Dieser Hashwert zeigt die Identität der Software und kann mit einem veröffentlichten Stand verglichen werden.

# Benutzeroberfläche

## Übersicht der Ansichten

Abbildung 3 Hierarchie der Anwendungsdialoge

## Beschreibung der Ansichten

Die Ansichten der Anwendung sind im [Handbuch zu TRuDI 1.0](https://bitbucket.org/dzgtrudi/trudi/src/master/doc/TRuDI%20Handbuch.docx) aufgeführt.

## Backend

### Übersicht der Komponenten des Backends

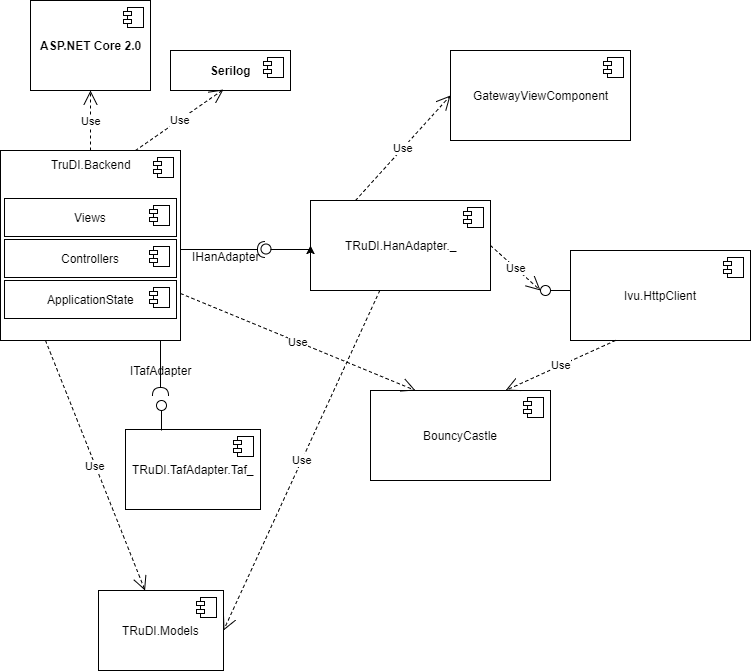


Abbildung 4 Pakete des TRuDI Backends

Das Backend ist eine ASP.NET Core 2.0 Konsolenanwendung. Aus dem Anwendungsframework werden folgende Dienste verwendet: *MvcCore*, *Views*, *RazorViewEngine* und *JsonFormatters*.

Für das Generieren des zur Kommunikation verwendeten TLS-Zertifikats und für die Berechnung von Hashwerten kommt die Bibliothek *BouncyCastle* zum Einsatz. Für das zentrale Logging wird *Serilog* verwendet. Beide Bibliotheken sind als NuGet Paket eingebunden.

Der Datenaustausch mit dem Frontend erfolgt über HTTPS, Websockets und dem Standard-Ausgabestrom.

Die ASP.NET Anwendung verwendet dem MVC-Paradigma folgend *Controller*, *Views* und *ViewComponents* für die Präsentation der Daten und die Verarbeitung von Benutzereingaben. Alle HTML-Inhalte werden durch die Controller erzeugt und per Razor-View-Engine bereits im Backend gerendert. Eine Manipulation des DOMs durch im Frontend laufenden Javascript Code findet dadurch nicht statt.

### Gerätekommunikation

Die Kommunikation mit SMGWs erfolgt über einen herstellerspezifischen Adapter welcher der Schnittstelle [*IHanAdapter*](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.Interface/IHanAdapter.cs) implementieren muss. In dieser Version sind Adapter für folgende Hersteller enthalten:

|  |  |  |
| --- | --- | --- |
| FLAG Kürzel | Hersteller | Adapter Type |
| PPC | PowerPlusCommunication AG | [HanAdapterPpc](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.Ppc) |
| THE | Theben AG | [HanAdapterThebenAG](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.ThebenAG) |
| LGZ | Landis+Gyr AG | [HanAdapterLandisGyr](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.LandisGyr) |
| DNT | Sagemcom Dr Neuhaus GmbH | [HanAdapterDrNeuhaus](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.DrNeuhaus) |
| DVL | devolo AG | [HanAdapterKiwigrid](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.Kiwigrid) |
| KIG | Kiwigrid GmbH |

Tabelle 2 Liste der HAN Adapter

Die Adapter werden als Assembly in der Klasse *HanAdapterRepository* registriert. Der Name der Assembly und der darin enthaltenen Implementierung von [*IHanAdapter*](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.Interface/IHanAdapter.cs) muss dabei folgenden Konventionen entsprechen, damit der Adapter geladen werden kann.

Name der Assembly:

TRuDI.HanAdapter.Example

Namespace und Name der Klasse welche [*IHanAdapter*](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.Interface/IHanAdapter.cs) implementiert:

**namespace** TRuDI.HanAdapter.Example

{

**public** **class** **HanAdapterExample** : IHanAdapter

{

*// ...*

}

}

„Example“ ist dabei jeweils durch den Namen des Herstellers zu ersetzen.

Das Repository garantiert, dass nur Adapter aus der Liste der HAN-Adapter geladen werden können. Der ApplicationState stellt sicher, dass maximal ein HanAdapter gleichzeitig aktiv sein kann.

Zur Gerätekommunikation steht jedem Herstelleradapter der *IVU.HttpClient* zur Verfügung. Diese Komponente entspricht derjenigen Schnittstelle des ASP.NET Framework Standards[[2]](#footnote-2). Die Implementierung wurde jedoch dahingehend angepasst, dass kryptorelevante Funktionen nicht direkt durch das Betriebssystem, sondern mit Unterstützung der Bibliothek *BouncyCastle* abgebildet werden. *BouncyCastle* wird deshalb benötigt, um auch unter Windows 7 Unterstützung für Elliptische-Kurven-Kryptographie unter Verwendung von Brainpool-Kurvenparametern zu bekommen[[3]](#footnote-3).



Abbildung 5 Adapterkommunikation

Bei den herstellerspezifischen Adaptern erfolgt der Verbindungsaufbau entweder nach TR03109-1, Kapitel 3.4.31 (HKS1) oder 3.4.3.2 (HKS2). Die Kommunikation erfolgt TLS-gesichert entsprechend den Vorgaben der TR03109-1, Kapitel 3.4.4.1. Die Anmeldedaten werden dabei von der Software nicht dauerhaft gespeichert. Sie werden nur vom HAN Adapter im Rahmen der Connect() Funktion zum Zweck des TLS-Verbindungsaufbaus verwendet.

Beim Verlassen des Programms wird eine evtl. bestehende Verbindung zum SMGW geschlossen.

### Datenmodell

Das zentrale Datenmodell des Backends bildet die Klasse [*ApplicationState*](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.Backend/Application/ApplicationState.cs). Beim Start der ASP.NET Anwendung wird ein Objekt des [*ApplicationState*](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.Backend/Application/ApplicationState.cs)als Singleton angelegt und per Dependency Injection allen Controllern übergeben. Der [*ApplicationState*](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.Backend/Application/ApplicationState.cs)verwaltet die Zustände der Benutzernavigation, der Dateneingaben sowie der Adapter-Kommunikation.

Die Objektmodelle für die Adapterkommunikation und die TAF-Berechnung sind in der Komponente [Models](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.Models/) zusammengefasst. Sie wird weiterhin für die Verarbeitung und Validierung aller vom Adapter oder per Datei geladenen Daten verwendet. Neben der reinen Schemaprüfung gegen die im Lastenheft referenzierten Schemas ([*Ar2418Validation*](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.XmlValidation/Ar2418Validation.cs)), enthält diese Komponente Model-Klassen zur Deserialisierung und Repräsentation der XML-Daten als C#-Objekte (*[XmlModelParser](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.XmlValidation/XmlModelParser.cs)*), sowie Funktionen zur Validierung der Rückgabewerte der IHanAdapter-Funktionen(*[ContextValidation](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.XmlValidation/ContextValidation.cs)*), sowie zur fachlichen Validierung des Datenmodells der XML-Daten (*[ModelValidation](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.HanAdapter.XmlValidation/ModelValidation.cs)*).

#### Abruf der Vertragsdaten

Vom HAN-Adapter müssen folgende Vertragsdaten bereitgestellt werden:

|  |  |  |  |
| --- | --- | --- | --- |
| Feld | Beschreibung | XML (AR2418) |  |
| TafId | Nummer des TAFs (z.B. TAF-2) |  | muss |
| TafName | Eindeutige Identifikation des TAF | tariffName | muss |
| Description | Kurze Beschreibung des TAF |  | optional |
| Meters | Liste der mit dem TAF verbundenen Zähler | meterId | muss |
| MeteringPointId | Zählpunktbezeichnung | usagePointId | muss |
| SupplierId | ID des Lieferanten | invoicingPartyId | muss |
| ConsumerId | ObjectID des Letztverbrauchers, dem die die Daten zugeordnet werden (Cosem Logical Device ohne .sm) | customerId | muss |
| Begin | Startzeitpunkt des Vertrags |  | muss |
| End | Endzeitpunkt des Vertrags |  | optional |

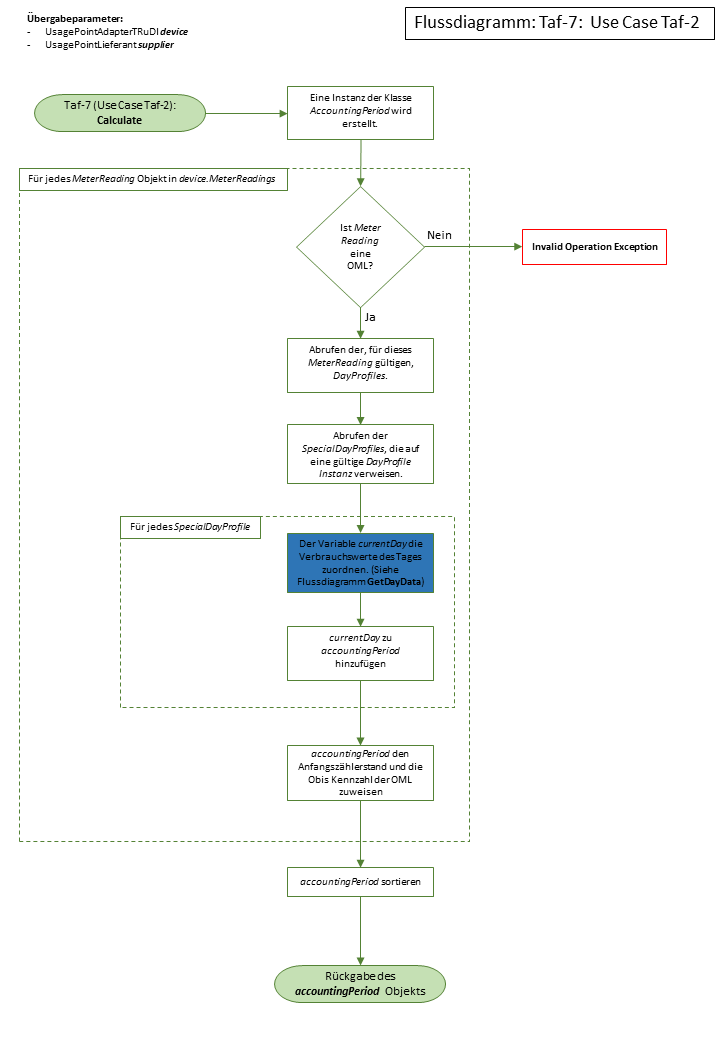
### Transparenzfunktion

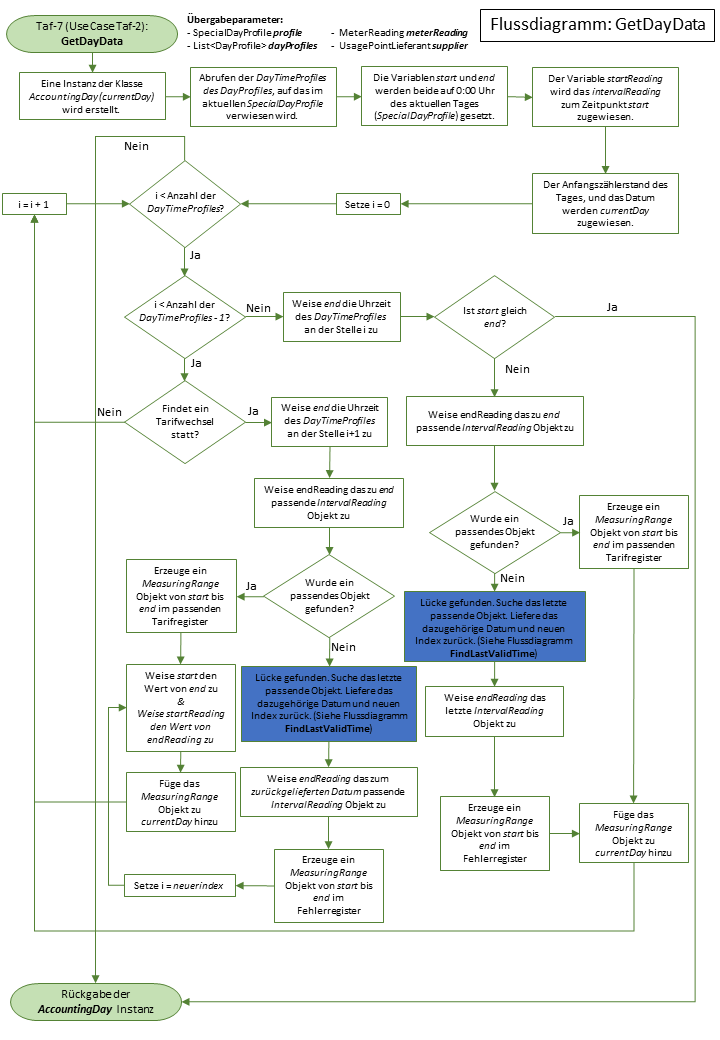
Analog zu *IHanAdapter* können TAF-Plugins eingebunden werden. Diese müssen das Interface [*ITafAdapter*](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.TafAdapter.Interface/ITafAdapter.cs) implementieren. Ein TAF-Plugin berechnet anhand einer Tarifdefinition (*[UsagePointLieferant](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.Models/BasicData/UsagePointLieferant.cs)*) und den Daten der aus dem SMGW abgelesenen originären Messwertliste (*[UsagePointAdapterTRuDI](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.Models/BasicData/UsagePointAdapterTRuDI.cs)*) die abgeleiteten Tarifregister. In dieser Programmversion sind TAF-Plugins für folgende Tarifanwendungsfälle implementiert:

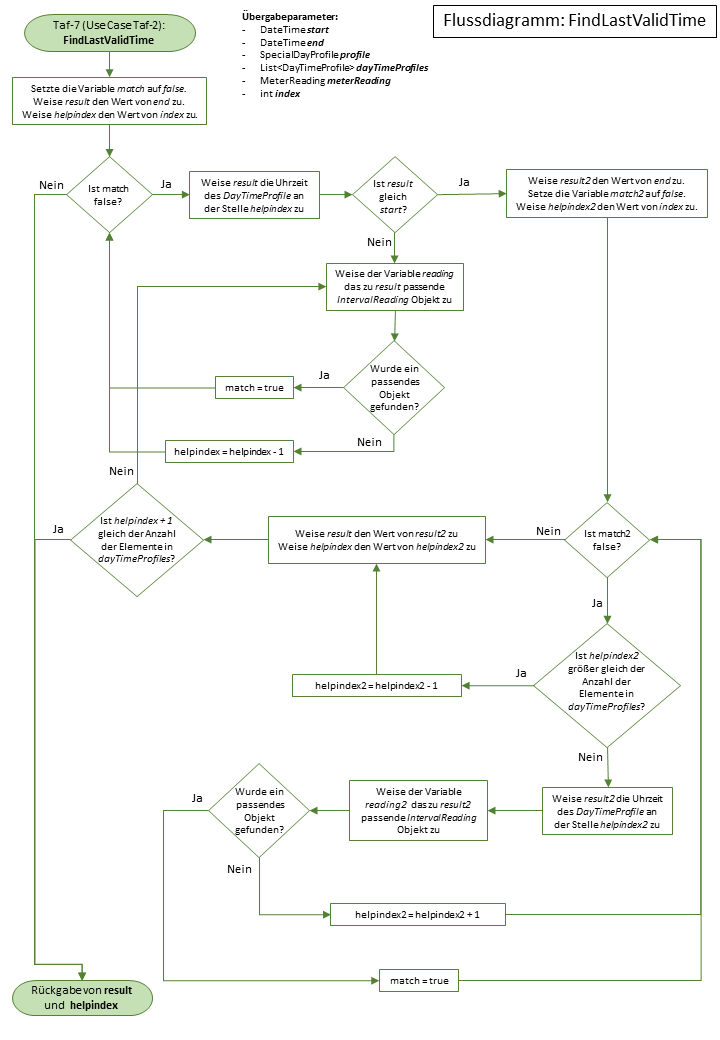
|  |  |  |
| --- | --- | --- |
| TAF-Id | Tarifbezeichnung | Assembly-Name |
| TAF-1 | Datensparsamer Tarif | TRuDI.TafAdapter.Taf1 |
| TAF-2 | Zeitvariabler Tarif | TRuDI.TafAdapter.Taf2 |

Tabelle 3 Liste der TAF-Plugins

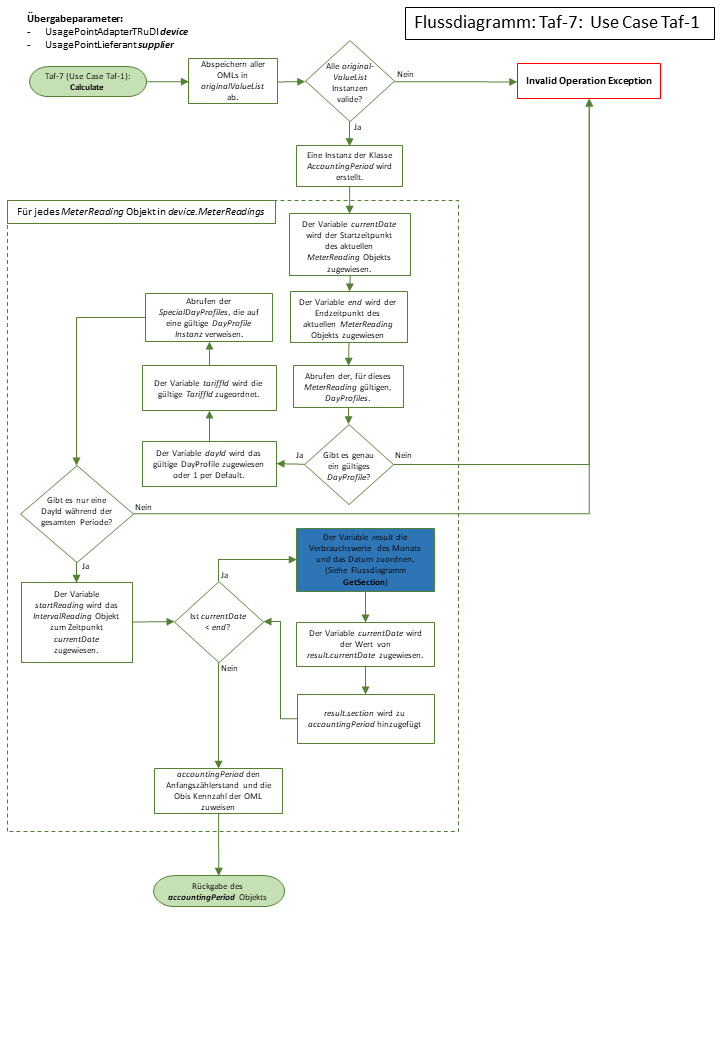
### Flussdiagramme für TAF-Plugin TAF-2







### Flussdiagramme für TAF-Plugin TAF-1



# Bezug zu Anforderungen nach PTB 50.8

## Anforderungen nach Kapitel 5.3 – Kundendisplay als Anzeige Applikation

**Kapitel 5.3.1 Anforderungen an die Software der separaten Anzeige-Applikation**

|  |  |  |
| --- | --- | --- |
| ID | Kurzbeschreibung | Kommentar |
| AN1.4 | Änderung der Zugangsdaten ermöglichen | Diese Funktion wird nicht für TRuDI benötigt. |
| AN1.8 | Anzeige aller Informationen von TR 3.4.2.1 und 4.1.3.2 | Zu TR 3.4.2.1   * **Systemzeit des SMGW**  Die Anzeige der Systemzeit ist nicht notwendig. Abgelesene Messwerte und Logdaten werden angezeigt und zeigen die Zeitstempel des geeichten Geräts. * **Aktuelle Zählerstände** Die aktuellen Zählerstände der Originären Messwertliste sowie die abgeleiteten Register bei TAF-1 und TAF-2 werden in der Ansicht „Abrechnungsdaten“ geliefert, wenn der Bediener zuvor den Abfragezeitbereich so eingestellt hat, dass dieser bis zum aktuellen Zeitpunkt reicht. Bei TAF-7 stehen aktuelle abgeleitete Register dann zur Verfügung, wenn sich der Abfragezeitbereich und der ausgerollte Kalender der Tarifdefinition bis zum aktuellen Zeitpunkt erstrecken. * **Aktuelle Tarifstufe je Auswertungsprofil** Die Anzeige der aktuellen Tarifstufe ist nicht notwendig. * **Historische Daten nach EER** Die historischen Daten werden anhand der originären Messwertliste berechnet und im Dialog „Abrechnungsdaten“ unter dem Reiter „Historische Messwerte“ angezeigt. * **Messwerte der letzten 24h** Siehe Spiegelstrich „Aktuelle Zählerstände“ * **Auflistung der für einen Letztverbraucher verfügbaren Datenstrukturen** Die im SMGW für den Letztverbraucher gespeicherten Daten werden gemäß den in Kapitel 4 beschriebenen Ansichten dargestellt.   Zu PTB 50.8, 4.1.3.2 (eichrechtlich relevant)   * **AG5.1 nur ein arithmetischer Rechenschritt.** Für die über die Anzeigefunktion dargestellten Werte ist keine weitere Berechnung vom Bediener vorzunehmen um die in Rechnung gestellten abgeleiteten Registerstände überprüfen zu können.  Zur Kontrolle der über die Transparenzfunktion angezeigten Registerstände für den Tarifanwendungsfall 1 und 2 ist keine Berechnung durch den Bediener nötig. * **AG2.3 Kennzeichnung fehlerhafter Wert** Unabhängig ob vom SMGW der Status im FNN oder PTB Format geliefert wird, erfolgt an der Oberfläche eine einheitliche Statusanzeige entsprechend der PTB-Definition (siehe PTB 50.8, Tabelle 4-2). Statuswerte nach FNN werden in die PTB-Darstellung konvertiert (siehe Klasse [StatusFnn](https://bitbucket.org/dzgtrudi/trudi/src/master/src/TRuDI.Models/BasicData/StatusFNN.cs)). Fehlerhafte Werte werden mit einem Warnsymbol und einer textuellen Darstellung des Fehlers gekennzeichnet. * **Kennung des Letztverbrauchers (alle TAF)** Die Kennung des Letztverbrauchers wird bei in der Ansicht „Tarifdaten“ als *Kunden ID* angezeigt. * **Kennung des Lieferanten (alle TAF)** Die Kennung des Lieferanten wird in der Ansicht „Verträge“(Anzeigefunktion) bzw. „Tarifdaten“ (Transparenzfunktion) als *ID des Rechnungsstellers* angezeigt. * **Gateway Kennzeichnung (alle TAF)** Die Kennzeichnung des Gateways wird in der Ansicht „Verträge“ (Anzeigefunktion) bzw. „Tarifdaten“ (Transparenzfunktion) zum Label *Smart Meter Gateway* angezeigt. * **Abrechnungszeitraum (alle TAF)** Der Abrechnungszeitraum wird in der Ansicht „Verträge“ (Anzeigefunktion) bzw. „Tarifdaten“ (Transparenzfunktion) durch Start- und Ende Zeitpunkt angezeigt. * **Zählerkennzeichnung (alle TAF)** Die ID des Zählers wird in der Ansicht „Abrechnungsdaten“ im Reiter Originäre Messwertliste unter dem Label *Zähler* angezeigt. * **Originäre Messwertliste (alle TAF)** Die Originäre Messwertliste wird in der Ansicht „Abrechnungsdaten“ angezeigt. Die hier angezeigten Zeitstempel sind die Abfragezeiten, wenn diese vom jeweiligen SMGW geliefert werden. * **Länge der Registrierperiode (alle TAF)** Die Länge der Registrierperiode wird aus der originären Messwertliste ermittelt und in der Ansicht „Abrechnungsdaten“ unter dem Label *Messperiode* angezeigt. * **Aktivierte Tarifanwendungsfälle (alle TAF)** Der aktivierte Tarifanwendungsfall wird in der Ansicht Abrechnungsdaten im Bereich der Zählpunkt-Daten unter dem Label *Tarifanwendungsfall* angezeigt. * **Tarifumschaltzeitpunkte (TAF 2)** Tarifumschaltzeitpunkte können nicht in TRuDI selbst überprüft werden. Die Tarifumschaltzeitpunkte sind Gegenstand des Liefervertrags und dort hinterlegt. Darin muss auch ein eindeutiger Bezeichner für das Tarifmodell angegeben sein.  Bei einem vom SMGW intern abgebildeten TAF 2 wird vom HAN-Adapter bei der von der Funktion *LoadData()* zurückgelieferten XML Struktur, das Element *tariffName* mit dem eindeutigen Bezeichner des Tarifmodells an die TRuDI übergeben. Dieser Bezeichner wird in der Ansicht „Abrechnungsdaten“ im Bereich des Zählpunkts als *Tarifbezeichnung* angezeigt. Der Anwender kann anhand des Bezeichners erkennen, ob im SMGW dasselbe wie im Vertrag vereinbarte Tarifmodell eingerichtet ist. * **Register der Tarifstufen** Die Abrechnungsregister werden in der Ansicht „Abrechnungsdaten“ im Bereich des Zählpunkts tabellarisch angezeigt. * **Datum und Tagesendstände (TAF 6)** Die Anzeige von TAF-6 Daten erfolgt über die Ansicht „Abrechnungsdaten“ nach vorangegangener Auswahl eines Tages. * **Anzeige zentrale Tarifierung (TAF 7)** Die Anzeige der über TAF 7 in TRuDI gebildeten Abrechnungsdaten erfolgt zunächst auf die gleiche Weise wie bei direkt aus dem SMGW ausgelesenen. Zusätzlich steht bei der Ansicht „Abrechnungsdaten“ der Reiter „Tarifdaten“ zur Verfügung. Hier sind die die Schaltzeiten und Tagessummen für jede Tarifstufe in Form eines Balkendiagramms und einer tabellarischen Darstellung aufgeführt. Die Tagessummenanzeige kann für jeden Tag des Abrechnungszeitbereichs aufgerufen werden. * **Fehlerstatus/Fehlernummer** Zu den in der Ansicht „Abrechnungsdaten“ angezeigten originären Messwertlisten wird je Messwert dessen Status nach PTB-Definition angezeigt. |
| AN1.9 | Verbindungsaufbau nach TR | Siehe Kapitel 2.3.2, letzter Absatz. |
| AN4.1 | Identifikator der eichrechtlich relevanten Software des SMGW | Die detaillierten Informationen zur Firmwareversion und Version einzelner Firmwarekomponenten des SMGW sind in der Ansicht „Details zum Smart Meter Gateway“ tabellarisch aufgeführt. |
| AN5.1 | Rückwirkungs- und Nebenwirkungsfreiheit | Die über die Benutzerschnittstelle eingegebenen Werte werden von der Anwendung geprüft. Es werden nur plausible Werte für die Weiterverarbeitung akzeptiert. Bei Eingabe von ungültigen Werten wird eine Fehlermeldung ausgegeben.  Die Eingabesteuerelemente der Anwendung wurden mit Strings aus einer Liste von bekannten Problemwerten[[4]](#footnote-4) getestet. |
| AN1.10 | Navigation und Benutzerführung | Die Navigation und Benutzerführung ist durch eine Location-Breadcrumb-Navigationsleiste mit Elementen für jede Ansicht übersichtlich umgesetzt. Die aktuelle Hauptansicht wird dabei in der Leiste farblich hervorgehoben.  **Lücken und Messwert-Stati**  Schnelles Auffinden von Lücken und Messwerten mit besonderen Stati ist über die Schaltfläche „Status-Übersicht anzeigen“ der jeweiligen Originären Messwertliste möglich. |
| AN1.11 | Sichere Authentifizierung | Die Absicherung der Authentifizierung folgt AN1.9 und damit den Vorgaben der TR03109-1. Durch die Dialogführung ist sichergestellt, dass keine Befehle an das SMGW vor erfolgreicher Authentifizierung gesendet werden. Die Prüfung der Authentizität der Anmeldeinformation wird vom SMGW durchgeführt. |
| AN6.3 | Rückwirkungsfreiheit | Die Rückwirkungsfreiheit wird durch eine TLS gesicherte Kommunikation zwischen Front- und Backend sichergestellt. |
| AN6.4 | Verbindungen nur zum SMGW | Durch den Verbindungsaufbau gemäß Kapitel 2.3.2 und des Umstands, dass stets nur ein HAN-Adapter aktiviert wird sichergestellt, dass nur Verbindungen zum zugehörigen Gerät aufgebaut werden. Weiterhin werden die Inhalte des während des TLS Verbindungsaufbaus aus dem SMGW ausgelesenen Zertifikats, in der Ansicht „Details zum Smart Meter Gateway“ angezeigt. |
| AN7.2 | Authentizität und Integrität | Die Integrität der Anzeige-Applikation wird durch den in Kapitel 1.1.1 beschriebenen Integritätscheck per Hashwertbildung sichergestellt. Die Authentizität ist durch die in der Ansicht „Über TRuDI“ aufgeführte Versionsbezeichnung und über die in Kapitel 1.1.3 beschriebenen Anwendungshashwerte nachprüfbar. |
| AN8.2 | Prüfung des TLS Zertifikats des SMGW | Die Inhalte des während des TLS Verbindungsaufbaus aus dem SMGW ausgelesenen Zertifikats, werden in der Ansicht „Details zum Smart Meter Gateway“ angezeigt.  Da das Zertifikat nicht gespeichert wird, muss diese Prüfung vom Benutzer immer manuell durchgeführt werden. |

## Anforderungen nach Kapitel 7 – Transparenzsoftware

|  |  |  |
| --- | --- | --- |
| ID | Kurzbeschreibung | Kommentar |
| TS1.1 | Anzeige der Daten gem. Tab. 4-17 (Kap. 4.1.3.2.4) | Siehe AN1.8 |
| TS1.2 | Einfache Vergleichbarkeit zur Rechnung | Ausgabe der Abrechnungsregister erfolgt immer in tabellarischer Form. Bei TAF-7 führt TRuDI die Berechnung der Tarifregister so durch, dass keine weiteren Berechnungsschritte zur Nachvollziehbarkeit der Abrechnung notwendig sind. Die Register werden mit OBIS-Codes eindeutig gekennzeichnet. In der Ansicht „Abrechnungsdaten“ wird zusätzlich zu den angezeigten OBIS-Codes eine textuelle Erläuterung angezeigt. |
| TS1.3 | Identifikator des Regelwerks | Der Identifikator des Regelwerks wird in der Ansicht „Abrechnungsdaten“ im Bereich des Zählpunkts unter dem Label *Tarifanwendungsfall* angegeben. |
| TS1.4 | Ergonomische Navigation und Benutzerführung | Das Layout und die verwendeten Steuerelemente entstammen dem verbreiteten Bootstrap-Framework. Dadurch ist sichergestellt, dass die Bedienelemente auf Linux und Windows Betriebssystemen gleichermaßen vertraut wirken. |
| TS2 | Identifikation der Software | Die Software ist über den Dialog „Über TRuDI“ durch die angezeigten Versions-Nummern und Hashwerte identifizierbar. |
| TS3 | Schutz gegen Verfälschung -> AN4 Benutzerschnittstelle | Die Software ist durch den veröffentlichten Hashwert und im Dialog „Über TRuDI“ angezeigten Hashwert geschützt. |
| TS4 | Schutz gegen Verfälschung -> AN6 Kommunikation | Durch den vom HAN-Adapter durchgeführten TLS-Verbindungsaufbau, sowie der TLS-gesicherten Kommunikation zwischen Back- und Frontend ist sichergestellt, dass keine Manipulation der Kommunikation zwischen den beteiligten Komponenten erfolgt ist. |
| TS5 | Schutz gegen Verfälschung -> Programmcode | Durch den in Kapitel 1.1.1 beschriebenen Integritätscheck wird der Schutz gegen absichtliche Verfälschungen sichergestellt. |
| TS6 |  | Datenablesung vom SMGW ist per TLS gesichert. Der Download der Tarifierungsdaten (Lieferanten.XML) |
| TS7.1 | Download Mechanismus | Nicht relevant, da kein Regelwerk nachgeladen werden kann. |
| TS8.1 | Authentizität/Integrität der herunter geladenen Software | Die Software kann anhand des in Kapitel 1.1.3 beschriebenen Verfahrens auf Authentizität hin überprüft werden. |
| TS8.2 | Benutzbarkeit über Live-CD | Nach Freigabe der Software wird eine Linux-Live-CD für das Betriebssystem Heise Meterix o.ä. erstellt. |
| TS8.3 | Prüfung des Regelwerks | Die in der Software im Rahmen der Transparenzfunktion hinterlegten Regelwerke für TAF-1 und TAF-2 entstammen keiner externen Quelle sondern sind integraler Bestandteil der Applikation. Über den Integritätscheck der Anwendung sind deren Integrität und Authentizität sichergestellt. |
| TS8.4 | Verarbeitung von Auswerteprofilen | Das Auswerteprofil wird durch einen eindeutigen Identifikator referenziert. Im Dialog „Tarifdaten“ wird dieser Bezeichner unter dem Label *Tarifbezeichnung* angezeigt. Zusätzlich werden die RIPEMD-160 und SHA-3 Prüfsummen der geladenen Datei angezeigt.  Beim Laden der Tarifdaten werden die in Kapitel 2.3.3 angegebenen Prüfungen durchgeführt. Schlägt mindestens eine der Prüfungen fehl, wird eine Fehlermeldung ausgegeben und die weitere Programmausführung abgebrochen. |
| TS8.5 | Prüfung der orig. Messwertliste | Die Software zeigt in der Ansicht „Abrechnungsdaten„ die ausgelesenen originären Messwertlisten an.  In der Ansicht „Details zum Smart Meter Gateway“ werden die wichtigsten Informationen zum TLS Zertifikats des SMGW dargestellt. |
| TS9 | Rückverfolgbarkeit des Herunterladens rechtlich relevanter Software | Es ist kein Nachladen von ausführbarem Code vorgesehen. |

### Sonstige Anforderungen

**Darzustellende Inhalte – Anzeige von aktuellen Informationen (4.1.3.2.1)**

|  |  |  |
| --- | --- | --- |
| ID | Kurzbeschreibung | Kommentar |
| AG7.1 | Tab 4-14 eichrechtlich relevante Anzeige: aktuelle metrologische Informationen für den Letztverbraucher auf dem Kundendisplay | Zu PTB 50.8, 4.1.3.1 (aktuelle Informationen)   * **Gateway Kennzeichnung (alle TAF)** siehe AN1.8 * **Zählerkennzeichnung (alle TAF)** siehe AN1.8 * **Zählerstand und OBIS** siehe AN1.8 * **Originäre Messwertliste (alle TAF)** siehe AN1.8 * **Länge der Registrierperiode (alle TAF)** siehe AN1.8 * **Aktivierte Tarifanwendungsfälle (alle TAF)** siehe AN1.8 * **Aktuelle Tarifstufe (alle TAF)** Funktion ist nicht durch TRuDI abgedeckt. * **Tarifumschaltzeitpunkte (TAF 2)** Gelöst durch Verweis per eindeutigem Bezeichner auf eine extern (Webseite des Lieferant, Vertragsunterlagen, etc.) gespeicherte Tarifdefinition. * **Register der Tarifstufen (TAF 2)** Die aktuellen Registerstände können durch Wahl des Ablesezeitbereichs bis zum aktuellen Zeitpunkt ausgelesen und angezeigt werden. * **Datum und Tagesendstände (TAF 6)** Die im SMGW vorhandenen Tagesendstände können in * **Anzeige zentrale Tarifierung (TAF 7)**  Die Anzeige entspricht der Darstellung von TAF 2. * **Fehlerstatus/Fehlernummer** siehe AN1.8 * **Kennung des Letztverbrauchers** siehe AN1.8 * **Kennung des Lieferanten** siehe AN1.8 * **Alle Parameter des Regelwerks** siehe TS8.4 |

1. Für Windows siehe: <https://bitbucket.org/dzgtrudi/trudi/src/master/src/build.cmd>

   Für Linux siehe: <https://bitbucket.org/dzgtrudi/trudi/src/master/src/build.sh> [↑](#footnote-ref-1)
2. Siehe <https://docs.microsoft.com/de-de/dotnet/api/system.net.http.httpclient?view=netcore-2.0> [↑](#footnote-ref-2)
3. Übersicht zu in .NET Core enthaltenen Kryptofunktionen je Betriebssystem: <https://github.com/dotnet/corefx/blob/master/Documentation/architecture/cross-platform-cryptography.md> [↑](#footnote-ref-3)
4. Siehe <https://github.com/minimaxir/big-list-of-naughty-strings/blob/master/blns.txt> [↑](#footnote-ref-4)