

BfSE GmbH

Geschäftsführer Dipl.-Inf. Gerhard Danzl

<http://www.systematische-entwicklung.de>

Mozartstr. 5

83435 Bad Reichenhall

Deutschland



Ingenieurbüro für Softwareentwicklung

Dipl.-Inf.(FH) Manfred Novotny

<http://software-novotny.de>

Graf-Chiemo-Str. 5

83339 Chieming

Deutschland



DER ROTE FADEN (THEMEN-ÜBERBLICK)

1. IoT Frameworks

1. Hintergrund & Historie NexusDataLink
2. Anforderungen

2. Use Cases

3. System-Architektur

4. Schnittstellen & Kommunikation

1. Kommunikation über Channels
2. Deklarative/Generische Kommunikation: Channels

5. Security

6. Ausblicke

IOT IST (EIGENTLICH) NICHTS NEUES

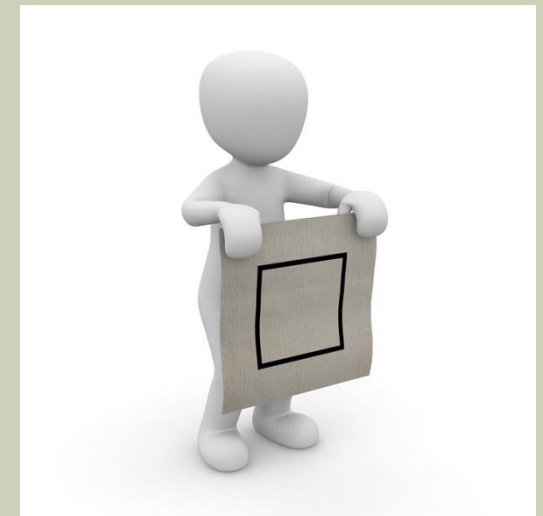


Ansätze zu finden
z.B. in SNMP, REST,...

Quelle: <https://pixabay.com>

ANFORDERUNGEN AN IOT FRAMEWORKS

- **Rahmenwerk für Device- und System-Kommunikation**
 - Swimlane für SW-Entwicklung
 - „Best Practices“
- **Reduktion von ...**
 - Komplexität
 - Aufwand (Zeit & Kosten)
 - Proprietärer Source Code
 - Bugs
- **Bereitstellung von ...**
 - Flexibilität / Anpassungsfähigkeit
 - Kommunikationsinfrastruktur
- **Maximierung von ...**
 - Stabilität
 - Sicherheit
- **Coverage aller Layer**
 - Device
 - Server (Cloud)
 - Client



DER ROTE FADEN (THEMEN-ÜBERBLICK)

1. IoT Frameworks

1. Hintergrund & Historie NexusDataLink
2. Anforderungen

2. Use Cases

3. System-Architektur

4. Schnittstellen & Kommunikation

1. Kommunikation über Channels
2. Deklarative/Generische Kommunikation: Channels

5. Security

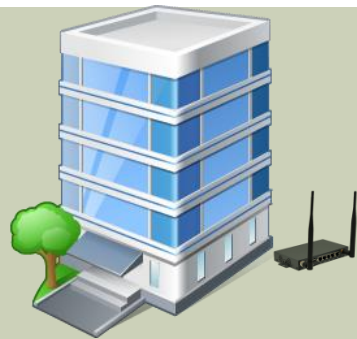
6. Ausblicke

USE CASE: TRINKWASSER- ÜBERWACHUNG



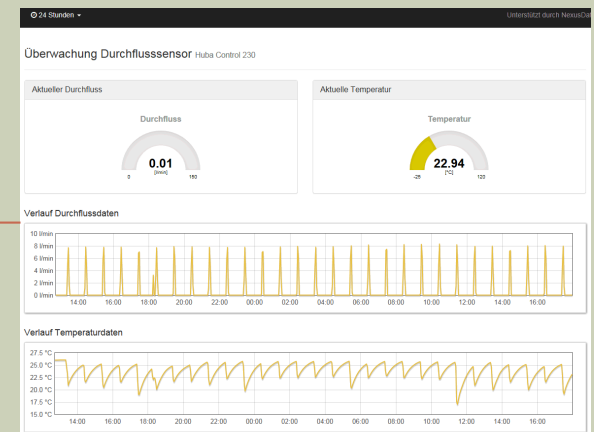
- Überwachung von Trinkwasserleitungen
- Legionellen-Prävention
- Anwendungsgebiete
 - Wellness-Einrichtungen
 - Sportanlagen
 - Private Immobilien
 - ...
- WLAN (private Cloud)
- WAN (public Cloud)

USE CASE: TRINKWASSER- ÜBERWACHUNG



Cloud
Server

Clients



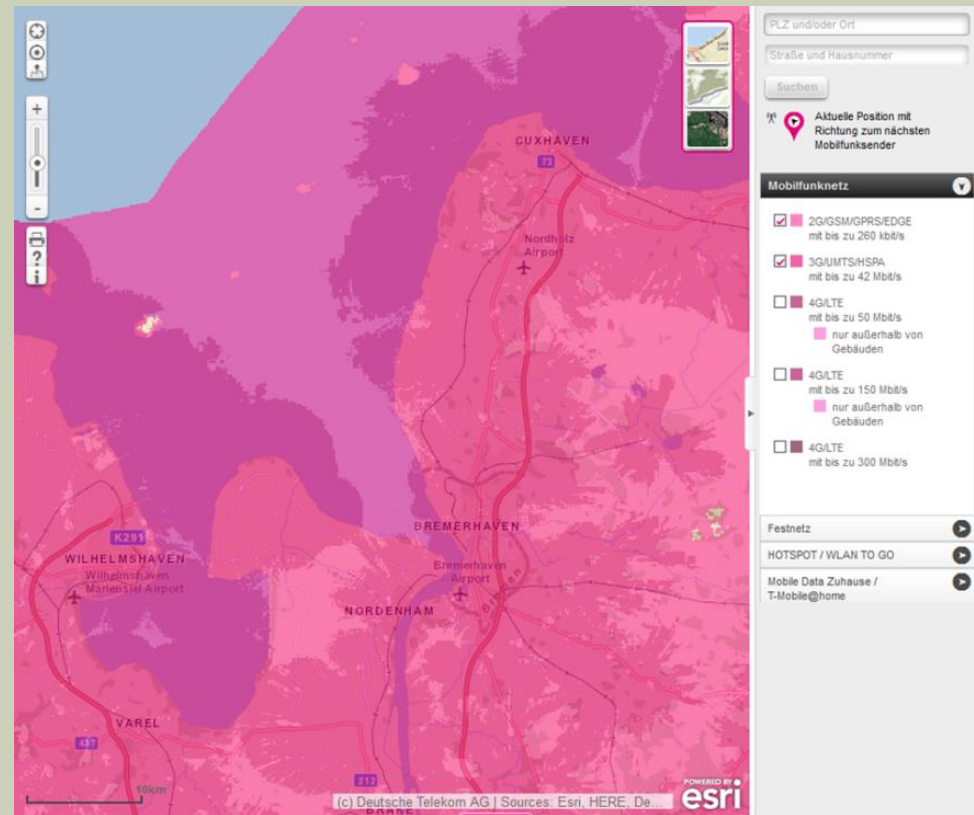
USE CASE: ERFASSUNG VON SCHIFFSDATEN

- Datenerfassung
 - Position (GPS)
 - Lage- & Beschleunigung (IMU)
 - Temperatur
 - Spannung/Leistung
- Datenübertragung via GSM (UMTS-Router)



USE CASE: ERFASSUNG VON SCHIFFSDATEN

- Datenerfassung
 - Position (GPS)
 - Lage- & Beschleunigung (IMU)
 - Temperatur
 - Spannung/Leistung
- Datenübertragung via GSM (UMTS-Router)



DER ROTE FADEN (THEMEN-ÜBERBLICK)

1. IoT Frameworks

1. Hintergrund & Historie NexusDataLink
2. Anforderungen

2. Use Cases

3. System-Architektur

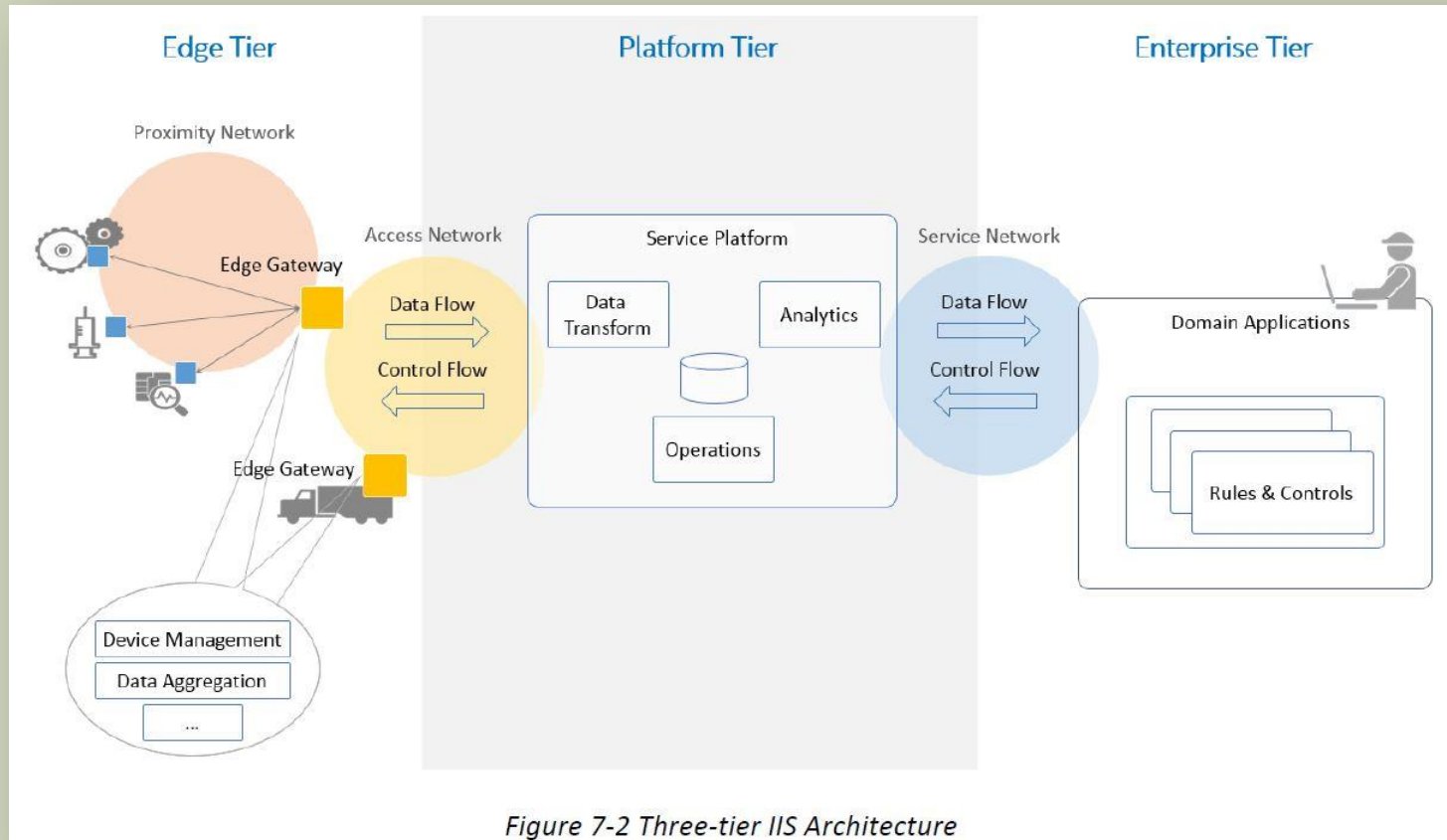
4. Schnittstellen & Kommunikation

1. Kommunikation über Channels
2. Deklarative/Generische Kommunikation: Channels

5. Security

6. Ausblicke

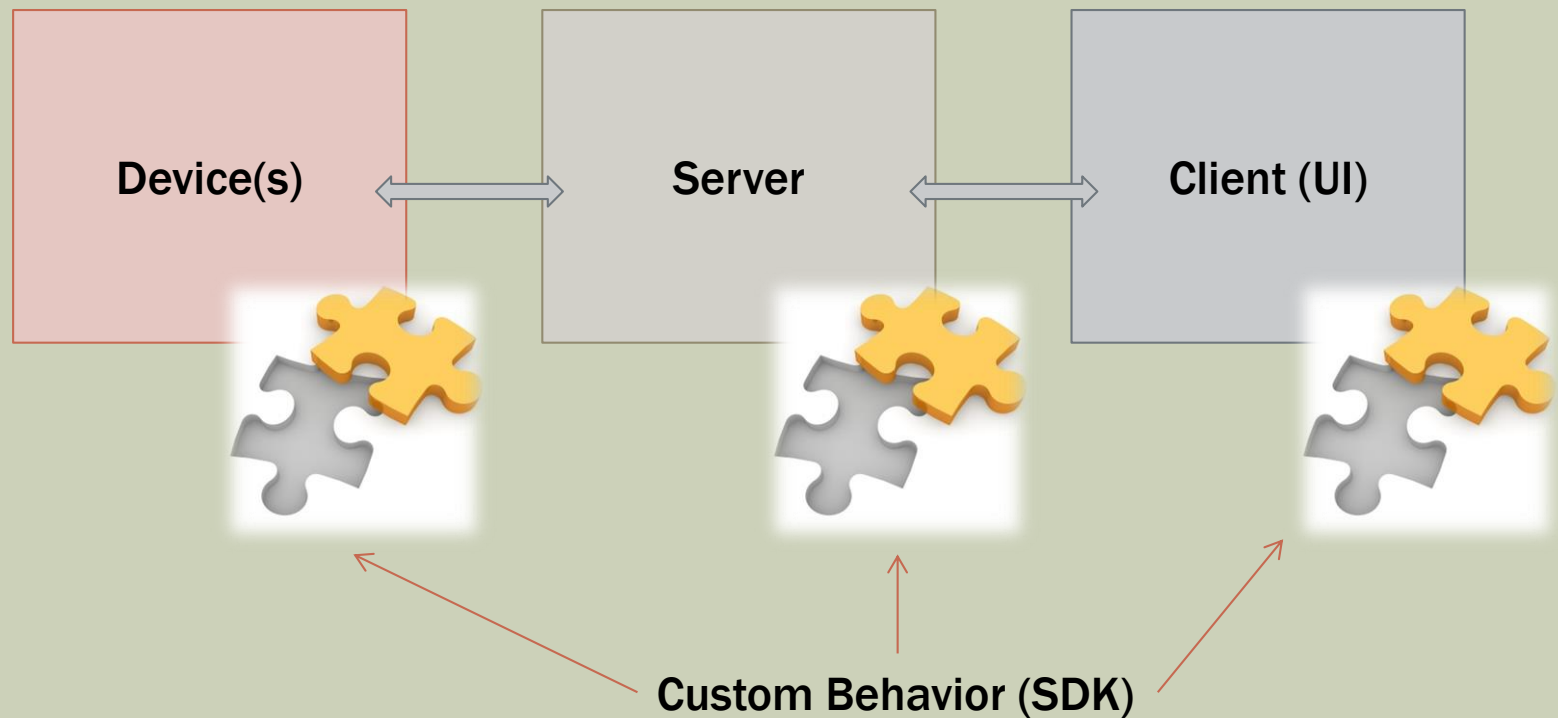
SYSTEM-ARCHITEKTUR: IIRA



Quelle: <http://www.iiconsortium.org/IIRA.htm>

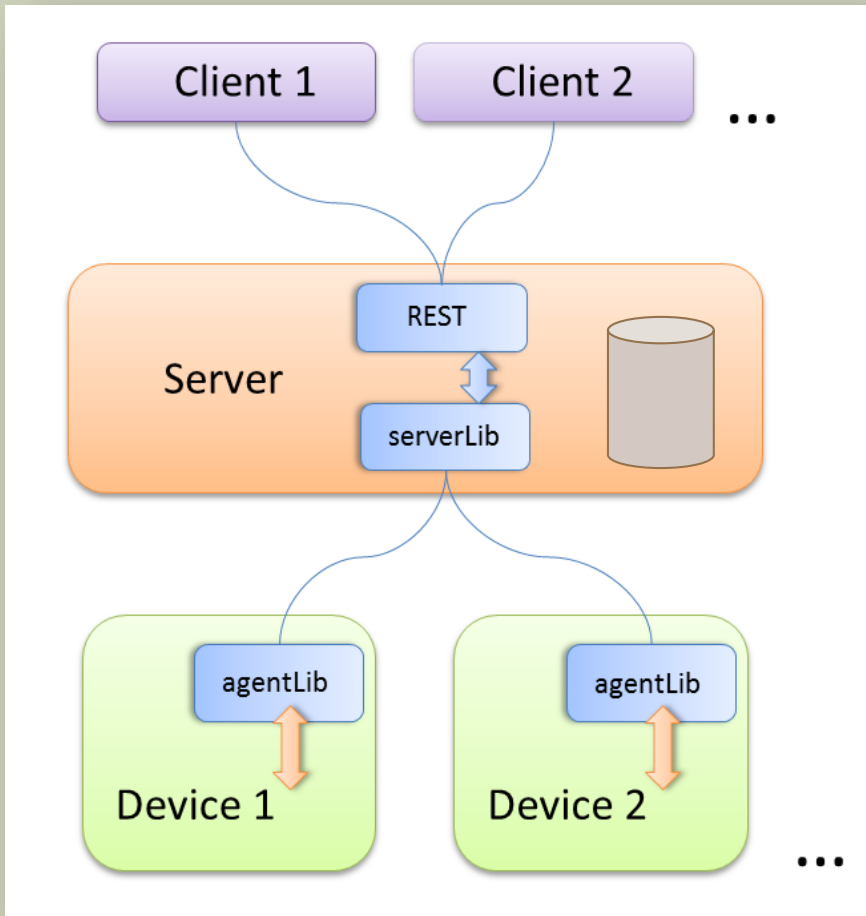
SYSTEM-ARCHITEKTUR

10.000 FT



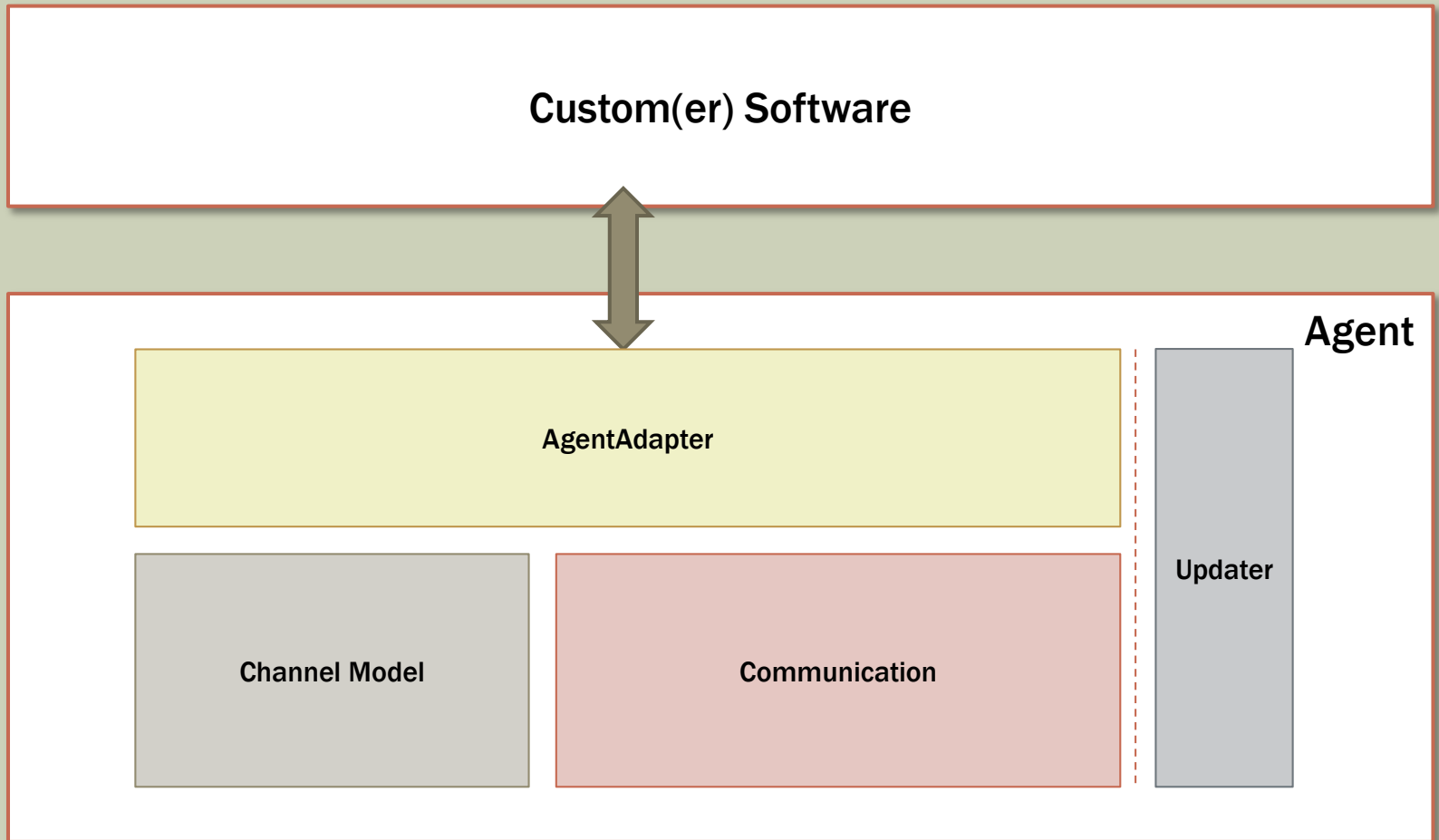
SYSTEM-ARCHITEKTUR

10.000 FT



- 3-Tier-Architecture
 - Device-Layer
 - Server/Cloud/Portal
 - Clients / UI
- Distributed Deployment

SYSTEM-ARCHITEKTUR AGENT



DER ROTE FADEN (THEMEN-ÜBERBLICK)

1. IoT Frameworks

1. Hintergrund & Historie NexusDataLink
2. Anforderungen

2. Use Cases

3. System-Architektur

4. Schnittstellen & Kommunikation

1. Kommunikation über Channels
2. Deklarative/Generische Kommunikation: Channels

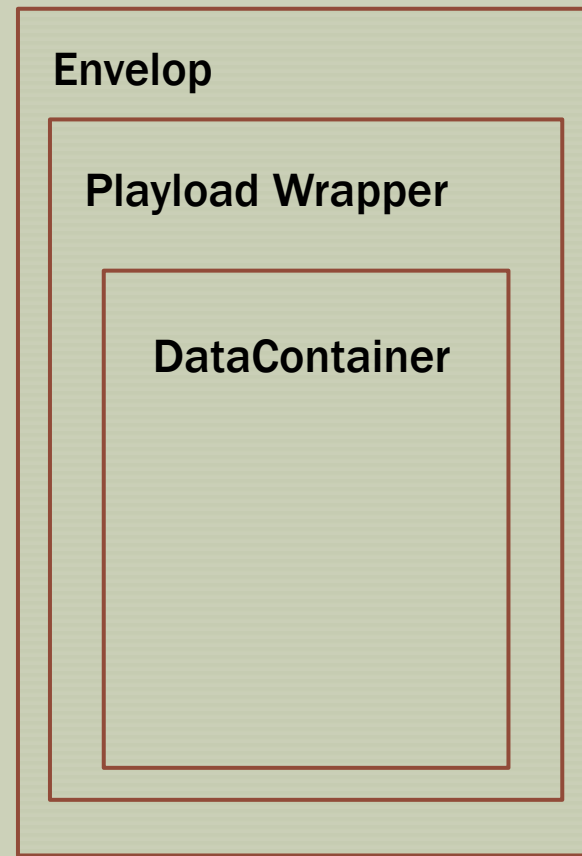
5. Security

6. Ausblicke

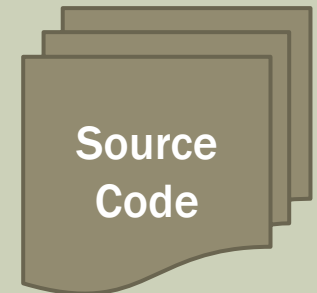
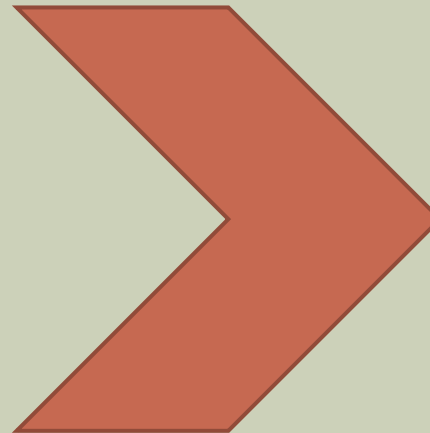
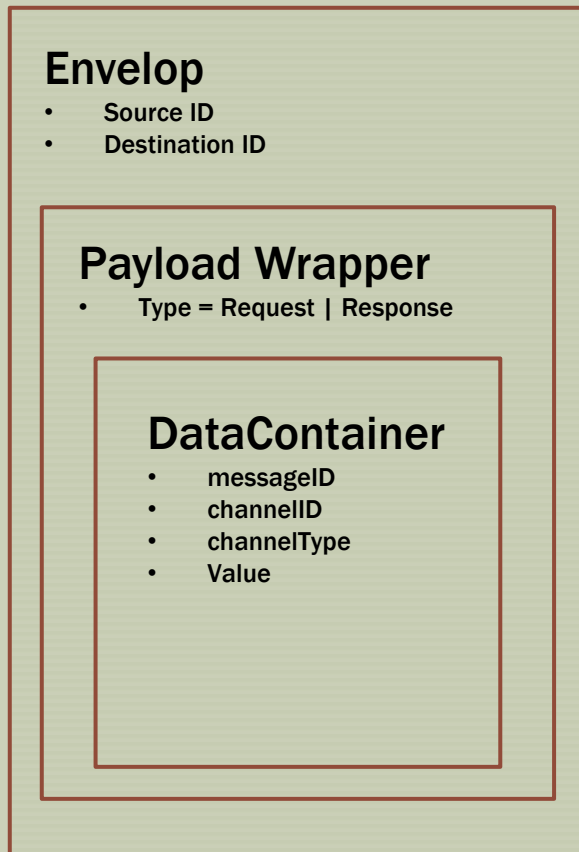
DATENPROTOKOLL DEVICE - SERVER

- Google's Protobuf
 - High Performance Serialization Framework
 - Deklarativer Ansatz (ASN.1)
 - Plattform-unabhängig
 - Support für C++, Java, JavaScript, ...
 - Lightweight
 - Hier: via TCP/IP

```
message Person {  
  required string name = 1;  
  required int32 id = 2;  
  optional string email = 3;  
}
```



DATENPROTOKOLL DEVICE - SERVER

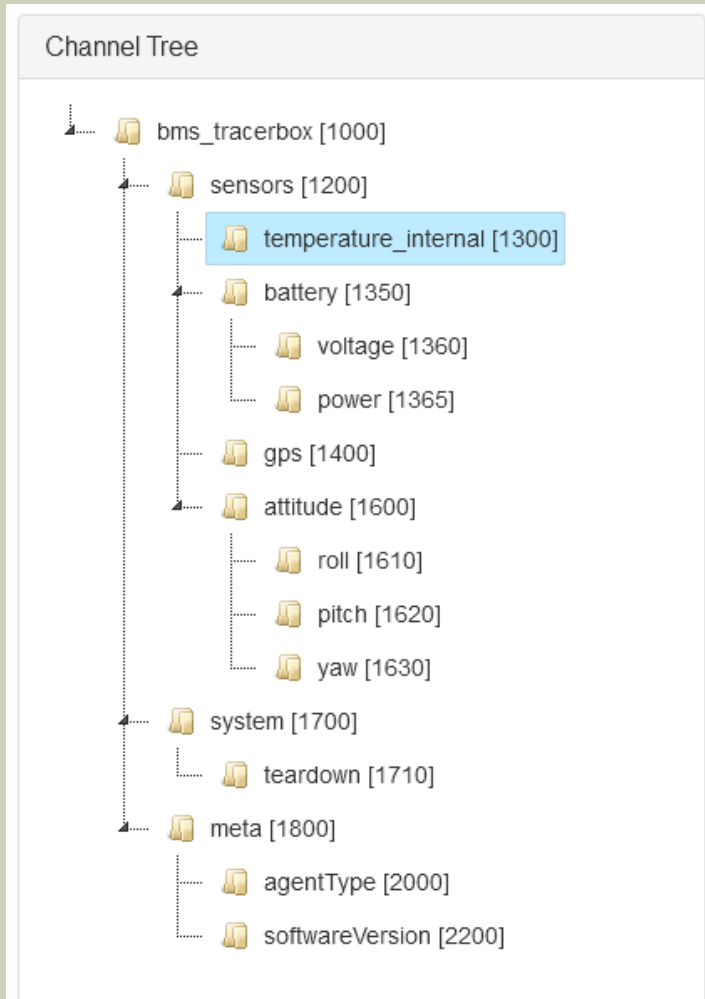


WAS SIND CHANNELS?

- Ein Channel...
 - Dient als Teil eines hierarchischen Trees der Modellierung der Schnittstelle
 - Beschreibt eine Eigenschaft des Systems
 - Hat einen Datentyp (Boolean, String, Integer, Float, Binary, Void)
 - Interagiert mit einer Gegenstelle (Server)
 - Read, Write
 - Wird deklariert

CHANNELS

[SYSTEM-DESIGN]



- Channels = hierarchisches Design der Datenstruktur des zu steuernden Systems

CHANNELS

[SYSTEM-DESIGN]

Channel ID **Parent Channel ID**

Must be unique and greater than 999

Channel Name

Up to 40 characters consisting of a-z, A-Z, 0-9, '-' and '_'

Full Channel Name (hierarchical)

Channel Description (optional)

Channel Type **Data Type** **Occurrence**

Transmission **Transmission Interval [ms]** **Data History** Record the data of this Channel

Resulting REST-URL

- Eigenschaften
 - ID + Name
 - Einordnung in Hierarchie
 - Datentyp
 - Request / Interval / Event
 - Beschreibung
 - Recorded Channel?

CHANNELS

[SYSTEM DESIGN]

Editor Code REST ChannelConfig

Channel Configuration System Properties

Channel Tree

- bms_tracerbox [1000]
 - sensors [1200]
 - temperature_internal [1300]
 - battery [1350]
 - voltage [1360]
 - power [1365]
 - gps [1400]
 - attitude [1600]
 - roll [1610]
 - pitch [1620]
 - yaw [1630]
 - system [1700]
 - teardown [1710]
 - meta [1800]
 - agentType [2000]
 - softwareVersion [2200]

Channel ID
Unique, numeric ID
Must be unique and greater than 999

Parent Channel ID
ID of the parent or 0

Channel Name
The channel's name
Up to 40 characters consisting of a-z, A-Z, 0-9, '-' and '_'

Full Channel Name (hierarchical)
The full name of the channel

Channel Description (optional)
Optional description

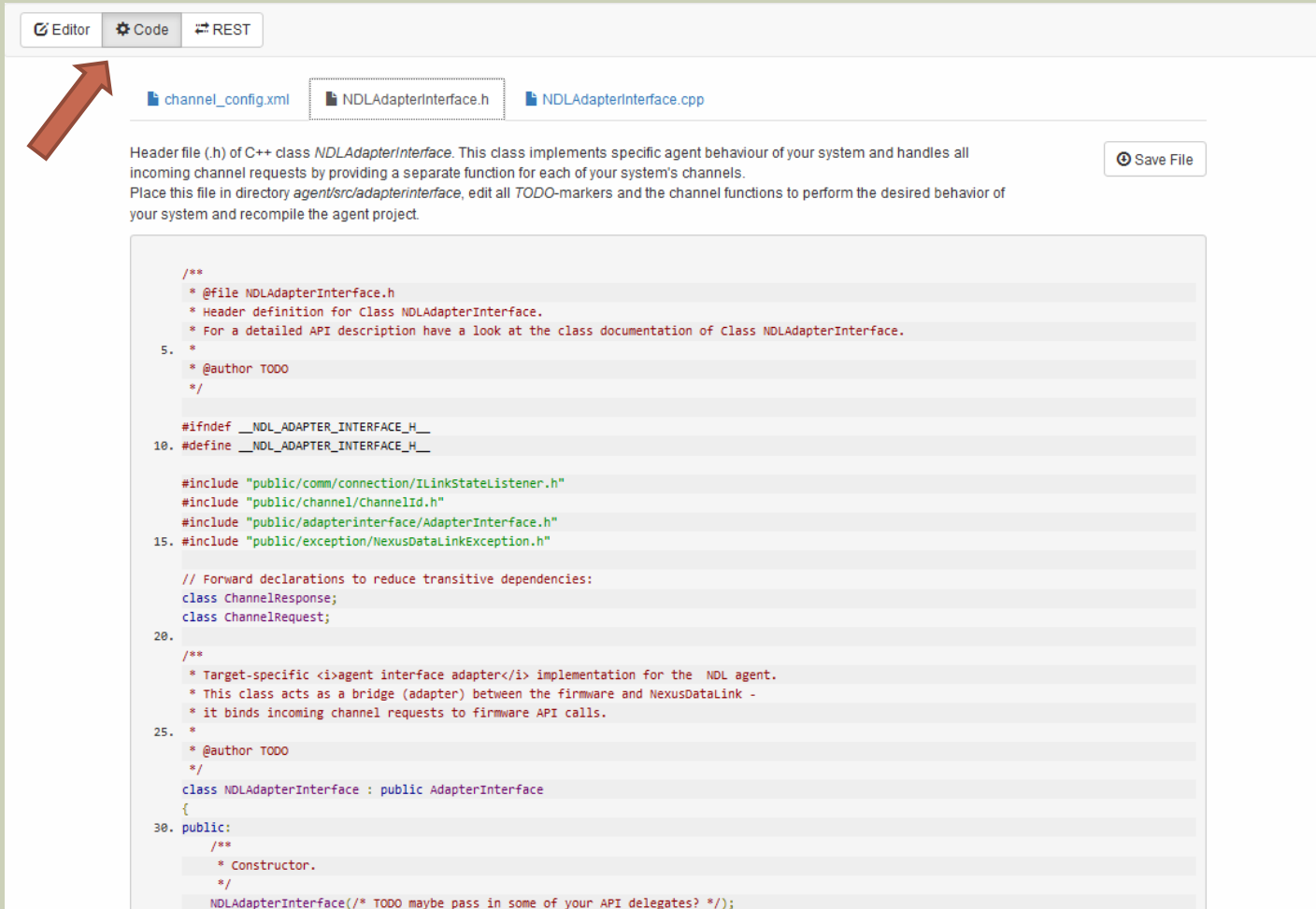
Channel Type [Dropdown] **Data Type** [Dropdown] **Occurrence** [Dropdown]

Transmission [Dropdown] **Transmission Interval [ms]** [Spinner] **Data History**
 Record the data of this Channel

Resulting REST-URL
The REST URL of this channel

Cancel Apply

CHANNELS [AUTO-GENERIERUNG]



The screenshot shows a code editor interface with three tabs: "Editor", "Code", and "REST". The "Editor" tab is active, and a red arrow points to it. The editor displays the header file `NDLAdapterInterface.h`. The file content includes a header comment, preprocessor directives for `__NDL_ADAPTER_INTERFACE_H__`, include statements for `IlinkStateListener.h`, `ChannelId.h`, `AdapterInterface.h`, and `NexusDataLinkException.h`. It also contains forward declarations for `ChannelResponse` and `ChannelRequest`, and the start of the `NDLAdapterInterface` class definition, which inherits from `AdapterInterface`.

Header file (.h) of C++ class `NDLAdapterInterface`. This class implements specific agent behaviour of your system and handles all incoming channel requests by providing a separate function for each of your system's channels.
Place this file in directory `agent/src/adapterinterface`, edit all `TODO`-markers and the channel functions to perform the desired behavior of your system and recompile the agent project.

```
/**
 * @file NDLAdapterInterface.h
 * Header definition for Class NDLAdapterInterface.
 * For a detailed API description have a look at the class documentation of Class NDLAdapterInterface.
5.  *
 * @author TODO
 */

#ifndef __NDL_ADAPTER_INTERFACE_H__
10. #define __NDL_ADAPTER_INTERFACE_H__

#include "public/comm/connection/IlinkStateListener.h"
#include "public/channel/channelId.h"
#include "public/adapterinterface/AdapterInterface.h"
15. #include "public/exception/NexusDataLinkException.h"

// Forward declarations to reduce transitive dependencies:
class ChannelResponse;
class ChannelRequest;
20.

/**
 * Target-specific <i>agent interface adapter</i> implementation for the NDL agent.
 * This class acts as a bridge (adapter) between the firmware and NexusDataLink -
 * it binds incoming channel requests to firmware API calls.
25.  *
 * @author TODO
 */
class NDLAdapterInterface : public AdapterInterface
{
30. public:
    /**
     * Constructor.
     */
    NDLAdapterInterface(/* TODO maybe pass in some of your API delegates? */);
```

CHANNELS

[AUTO-GENERIERUNG]

- Automatische Generierung einer Schnittstellenbeschreibung aus dem Channel System-Design

=> Deklarative Schnittstelle

```
<channel name="bms_tracerbox.sensors.temperature_internal" id="1300" channeltype="read">
  <description>
    Box-internal temperature in degree celsius
  </description>
  <occurrence value="single"/>
  <containertype value="float"/>
  <transmission value="request"/>
</channel>
```

CHANNELS

[AUTO-GENERIERUNG]

- Automatische Source Code Generierung der Adapterklasse
=> Deklarative Schnittstelle

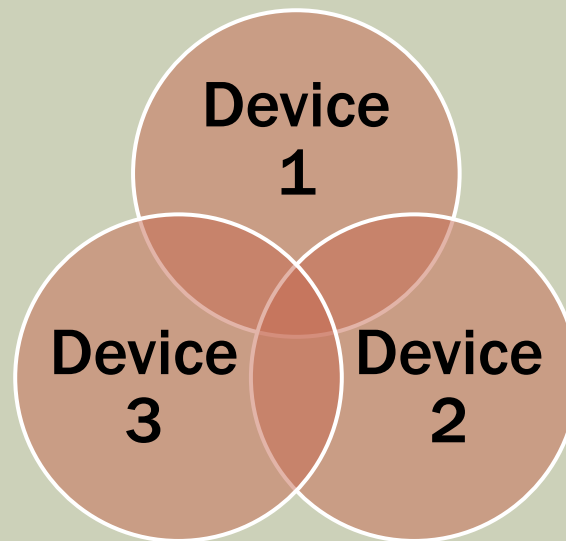
```
void NDLAdapterInterface::handle_1300_bms_tracerbox_sensors_temperature_internal(ChannelRequest* request,
                                                                                   ChannelResponse* responseProposal)
    throw (NexusDataLinkException) {

    // DataContainer-Type: float
    // Channel-Type       : read
    // Channel-Occurrence: single
    // Transsmission-Type: request

    // TODO provide value to be sent to server
    qreal value = 0.0;

    ((FloatContainer*)responseProposal->getDataContainer())->setValue(value);
}
```

STATISCHE CHANNELS

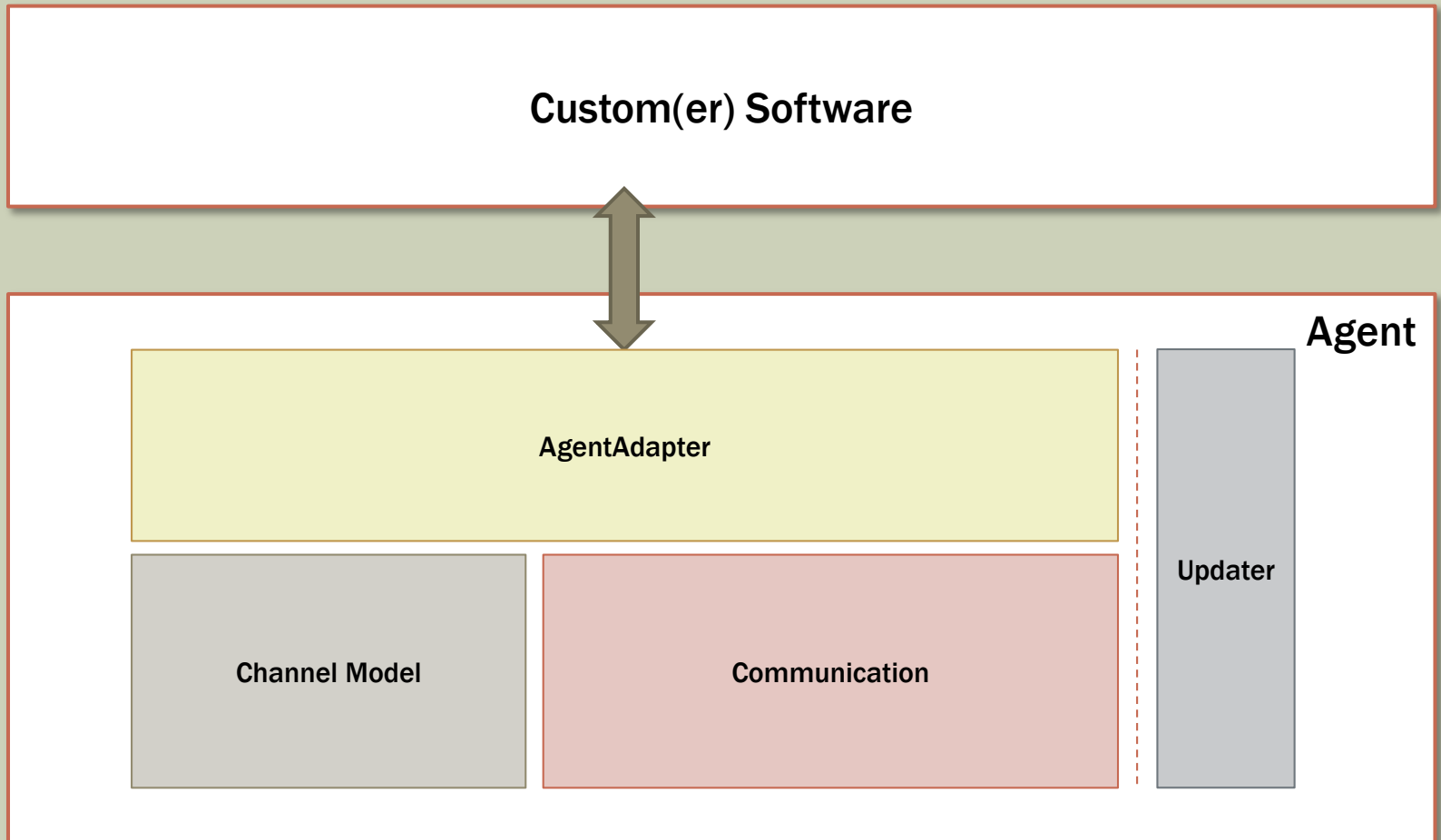


= gemeinsame Eigenschaften & Fähigkeiten

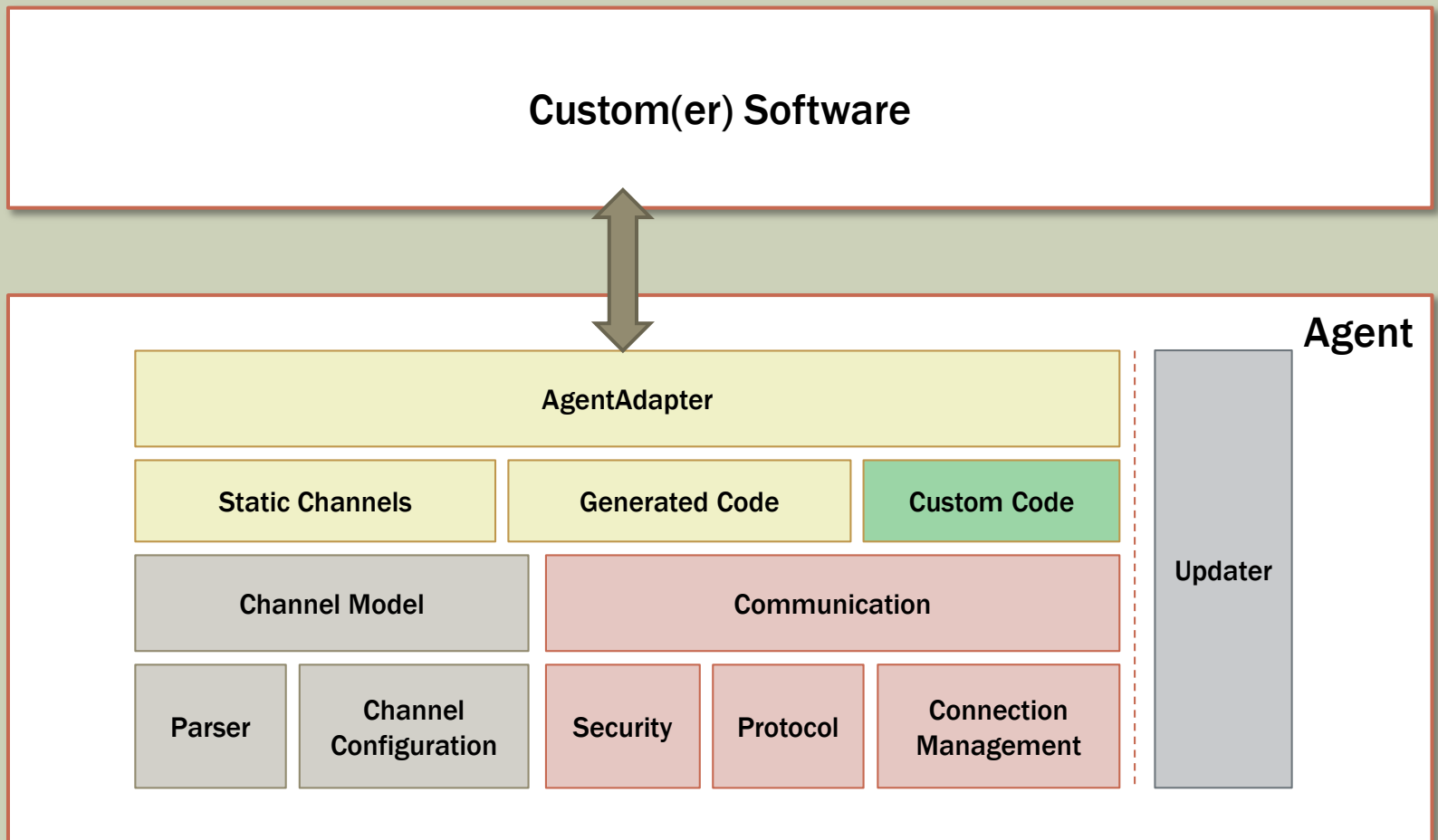
STATISCHE CHANNELS

Channel Name	Channel Type	Description
agent.static.protocolversion	read	Provides the agent's NDL protocol version
agent.static.id	read	Provides the agent's unique ID
agent.static.login	read	Provides the agent's login for authentication at the server
agent.static.password	read	Provides the agent's password for authentication at the server
agent.static.heartbeat	void	Heart-beat mechanism to detect connection state
agent.static.teardown	write	Perform a clean shutdown (value=0) or reboot (value=1) of the platform. Optional channel, that can be disabled if not supported.
agent.static.channelconfig	read	Provides the agent's channel configuration (XML)
agent.static.linkstate	write	Receives the link state (log-in state) from the server
agent.static.logfile	Read	Delivers the current content of the Agent's log file
agent.static.update	Read/write	Triggers and monitors the remote update of an NDL Agent
agent.static.position	read	The geo-position of the NDL Agent. Optional channel, that can be disabled if not supported.

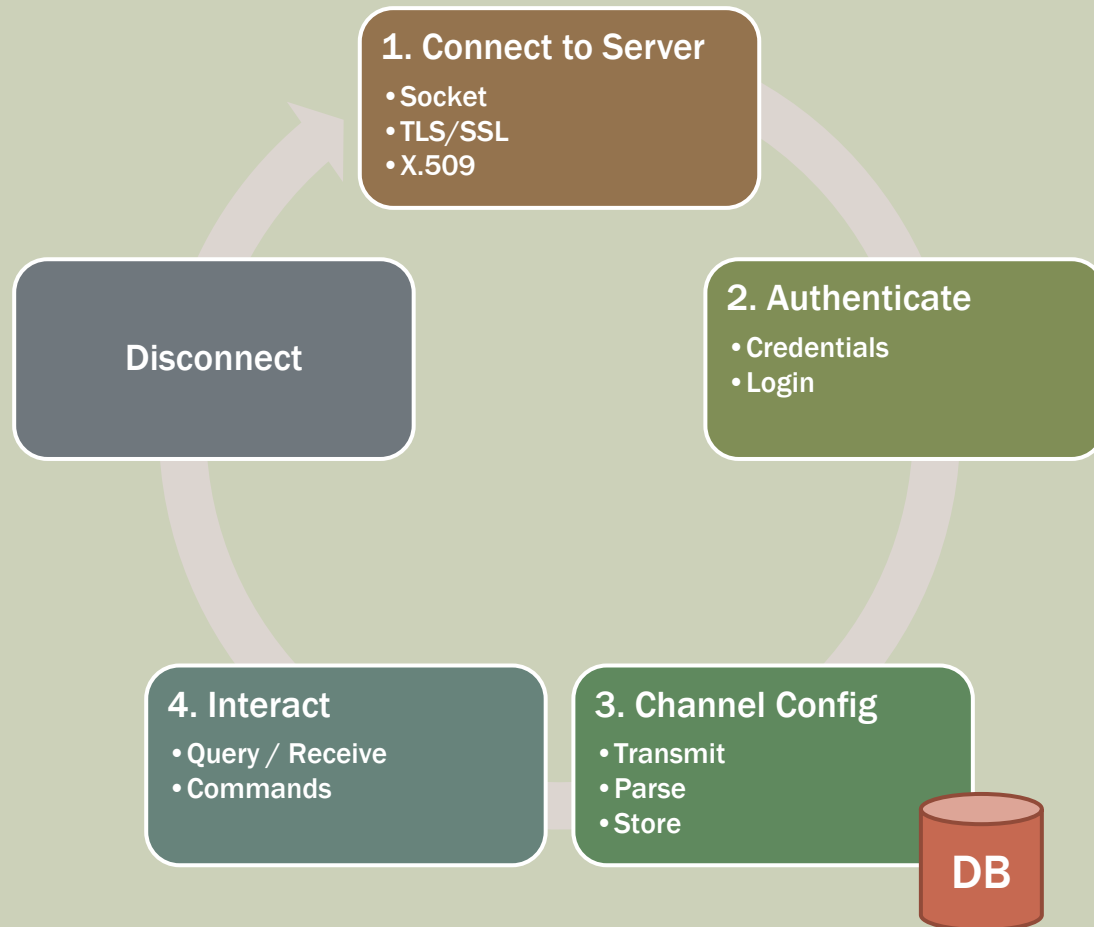
SYSTEM-ARCHITEKTUR CHANNELS



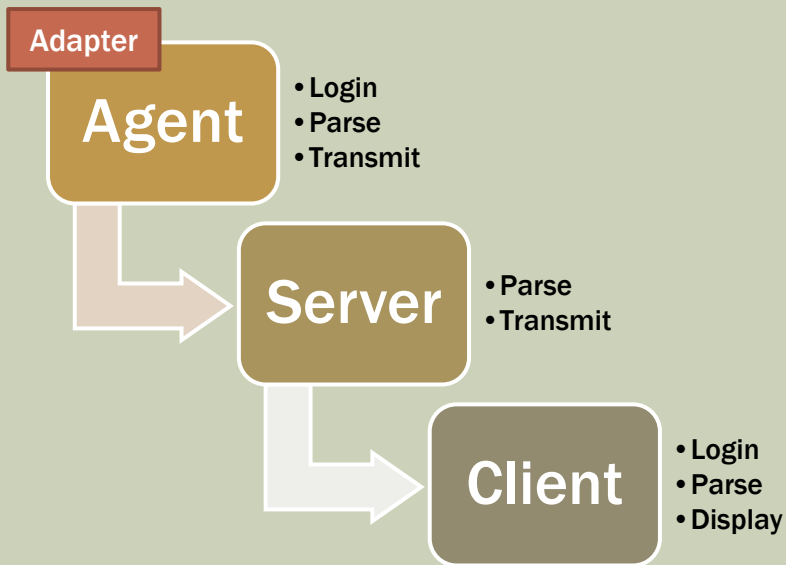
SYSTEM-ARCHITEKTUR CHANNELS



DEVICE CONNECTION LIFECYCLE



GENERIZITÄT DURCH INTERFACE PROPAGATION



```
<channel name="bms_tracerbox.sensors.temperature_internal" id="1300" channeltype="read">
  <description>
    Box-internal temperature in degree celsius
  </description>
  <occurrence value="single"/>
  <containertype value="float"/>
  <transmission value="request"/>
</channel>
```

```
<channel name="bms_tracerbox.sensors.temperature_internal" id="1300" channeltype="read">
  <description>
    Box-internal temperature in degree celsius
  </description>
  <occurrence value="single"/>
  <containertype value="float"/>
  <transmission value="request"/>
</channel>
```

```
<channel name="bms_tracerbox.sensors.temperature_internal" id="1300" channeltype="read">
  <description>
    Box-internal temperature in degree celsius
  </description>
  <occurrence value="single"/>
  <containertype value="float"/>
  <transmission value="request"/>
</channel>
```

GENERISCHE UI-CLIENTS

NexusDataLink Portal My Agents Documentation Tools John Smith

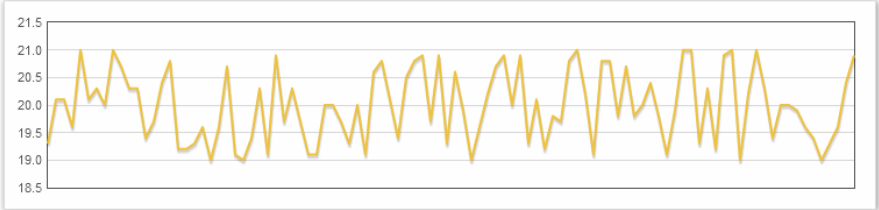
NDL-Agent-777 Details Interface **Monitor** Event-History Logs Update Map Shutdown

Please **double-click** on a Channel below to add it to a monitor slot (Channels that cannot be monitored are displayed as disabled items)

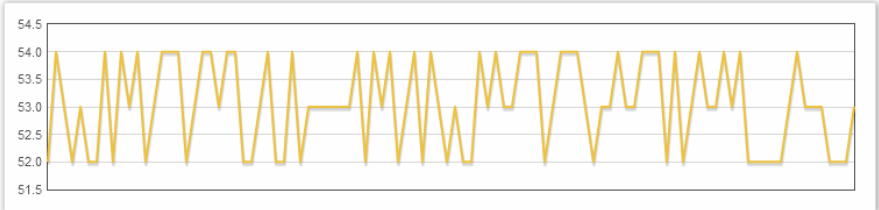
Channel Tree

- regime [100]
- update [50]
- position [100]
- exampleProject [1000]
 - sensors [1200]
 - temperature [1400]**
 - humidity [2600]
 - gpio [1800]
 - digital_input_port [2000]
 - actors [1600]
 - switch [2200]
 - valve [2800]
 - meta [3000]

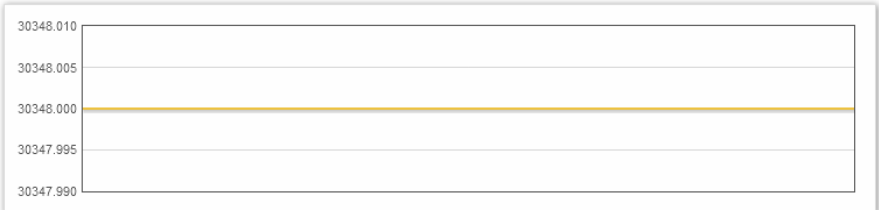
exampleProject.sensors.temperature (ID=1400)



exampleProject.sensors.humidity (ID=2600)



exampleProject.actors.valve (ID=2800)

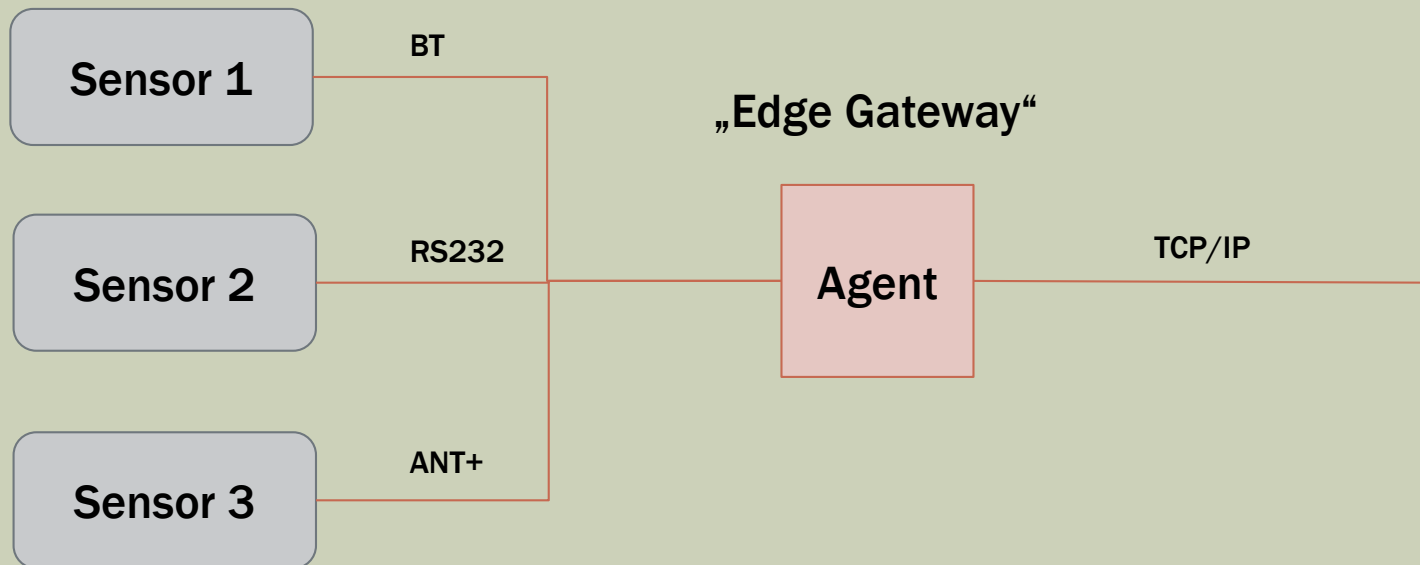


Select Monitoring Interval:

Every 500ms

▶ Start ■ Stop

ANBINDUNG „LOW LEVEL DEVICES“



DER ROTE FADEN (THEMEN-ÜBERBLICK)

1. IoT Frameworks

1. Hintergrund & Historie NexusDataLink
2. Anforderungen

2. Use Cases

3. System-Architektur

4. Schnittstellen & Kommunikation

1. Kommunikation über Channels
2. Deklarative/Generische Kommunikation: Channels

5. Security

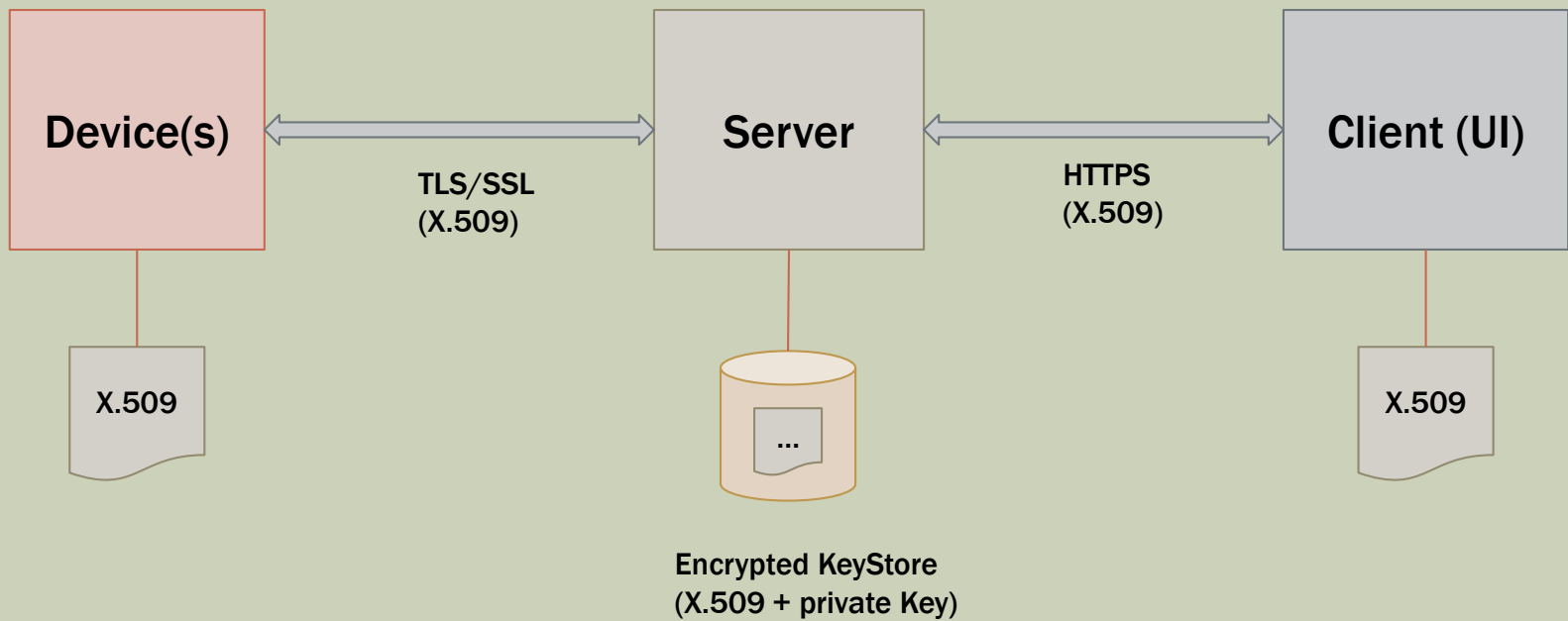
6. Ausblicke

SECURITY

X.509

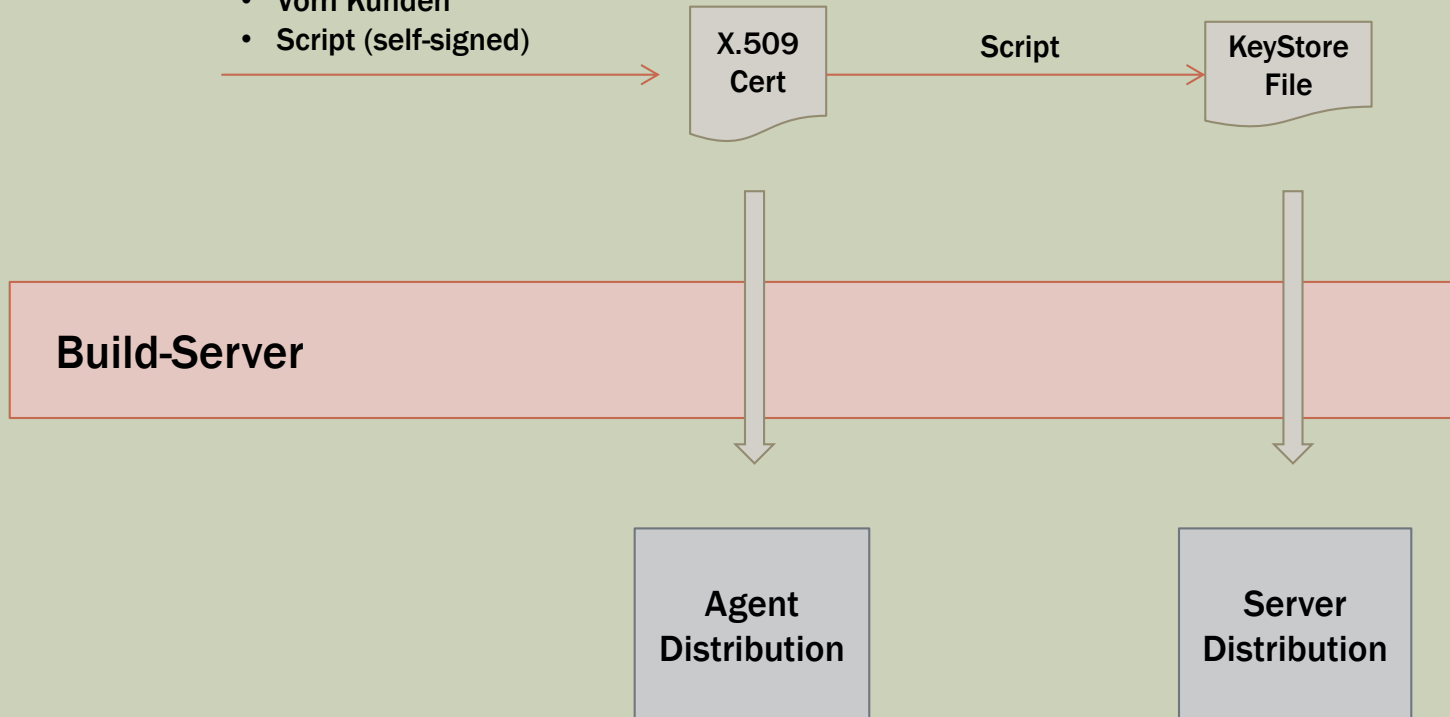
- X.509 = Zertifikat-basierende Verschlüsselung
 - Asymmetrisches Verfahren (pub. + priv. Key)
- Web-Technologie
- 2 Aufgaben:
 - Authentifizierung des Servers
 - Verschlüsselung der Kommunikation

SECURITY [ENCRYPTION]



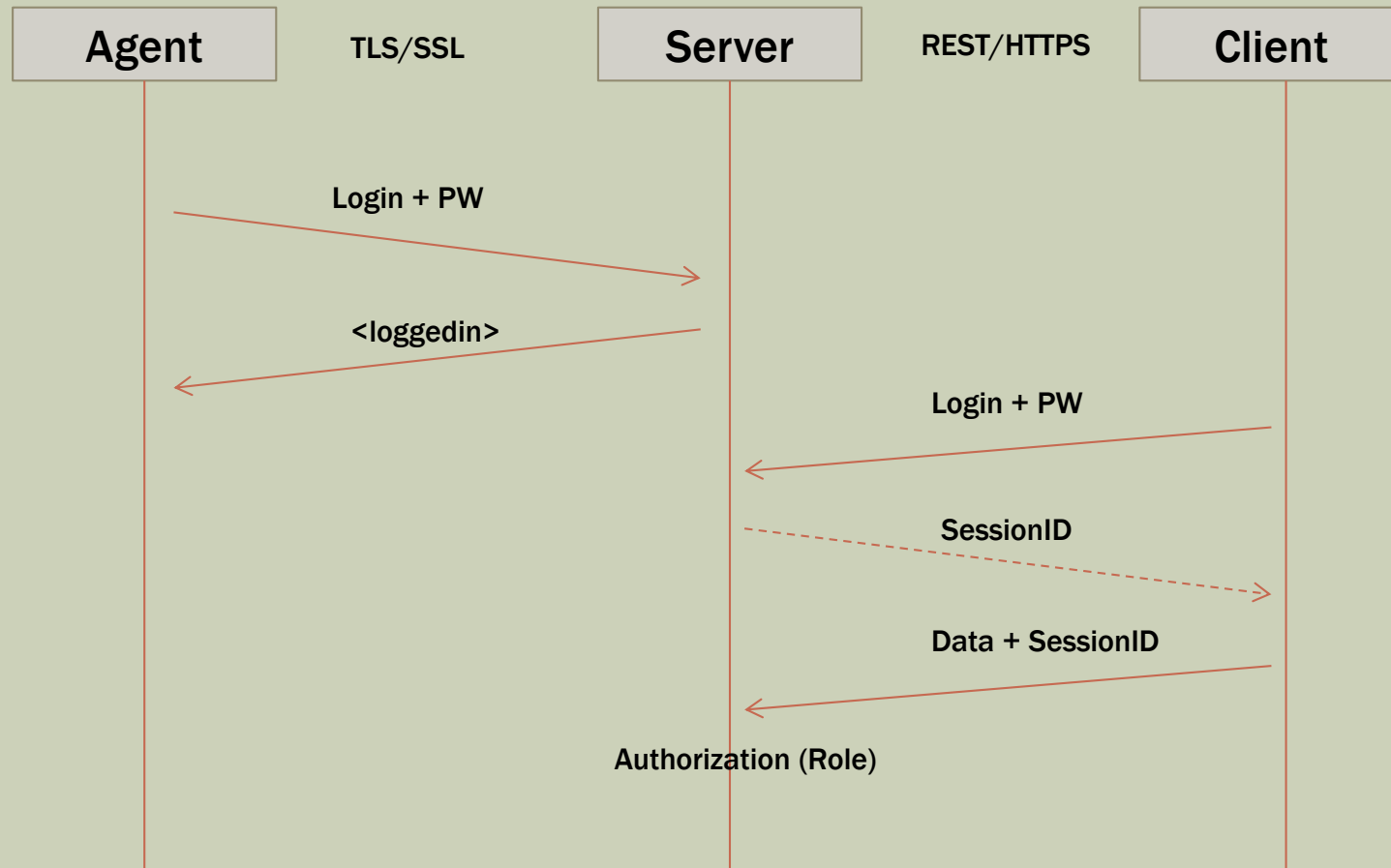
DYNAMISCHE EINBINDUNG VON ZERTIFIKATEN

- Von CA
- Vom Kunden
- Script (self-signed)



SECURITY

[AUTHENTICATION & AUTHORIZATION]



SECURITY

■ Lesetip:

- Linux Magazin 10/2015: Bedrohter Schwarm – Sicherheit für IoT-Geräte
 - <http://www.linux-magazin.de/Ausgaben/2015/10/IoT-Security>
 - Bedrohungen / Angriffe
 - Verschlüsselung & Co.
- iX Developer 1/2016: Entwickeln für das IoT
 - <https://shop.heise.de/katalog/ix-developer-internet-der-dinge-2015>
 - Security-Kapitel



DER ROTE FADEN (THEMEN-ÜBERBLICK)

1. IoT Frameworks

1. Hintergrund & Historie NexusDataLink
2. Anforderungen

2. Use Cases

3. System-Architektur

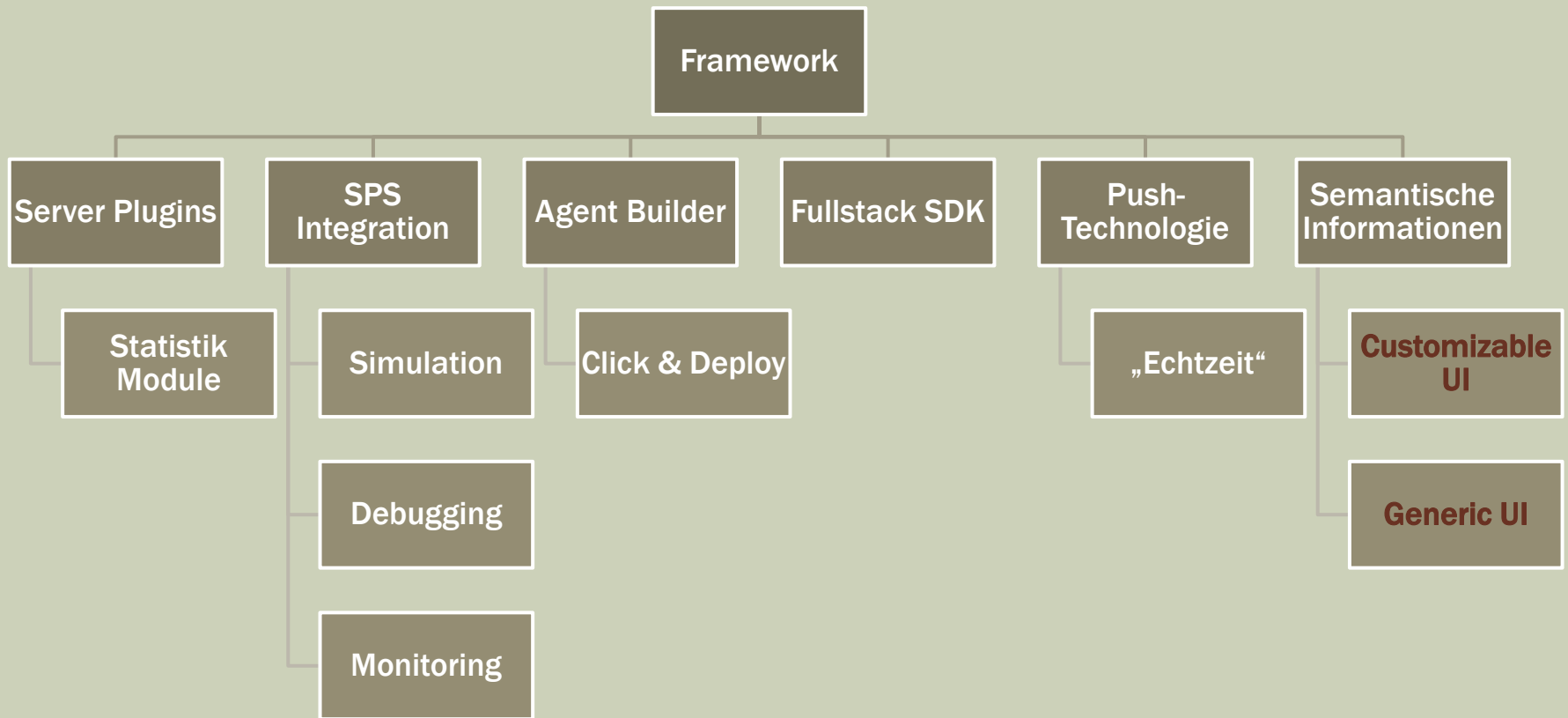
4. Schnittstellen & Kommunikation

1. Kommunikation über Channels
2. Deklarative/Generische Kommunikation: Channels

5. Security

6. Ausblicke

ZUKÜNFTIGE ENTWICKLUNGEN



DANKE FÜR IHRE AUFMERKSAMKEIT!

