Konfiguration von OSGi-Anwendungen

mit dem Configuration Admin Service

Christoph Schmidt-Casdorff, Thorsten Vogel - iks GmbH

Agenda

- Konfiguration
- Konfiguration und OSGi
- Konfiguration und *Declarative Services*
- Konfiguration und *Spring Dynamic Modules*

Wo sind wir ?

- Konfiguration
- Konfiguration und OSGi
- Konfiguration und *Declarative Services*
- Konfiguration und *Spring Dynamic Modules*

Was verstehen wir unter Konfiguration?

- *Setzen von Parametern, die Umfang, Aussehen, Verhalten und Ergebnisse einer Software beeinflussen.*

  Zitat (http://de.wikipedia.org)

- Eingriff in die Wirkungsweise eines Systems
  - Veränderung zur Laufzeit - jederzeit
  - notwendig zur Administration

Beispiele aus dem Leben

- Konfiguration mittels JVM Optionen
- Konfiguration mittels Properties-Datei
  - wird mittels Classloading oder Zugriff aus Filesystem bereitgestellt
  - pro Komponente eine Properties-Datei
- Konfiguration mittels META-INF/Services
  - vgl. JAXP (SAXFactory, Dom Factory, ...)

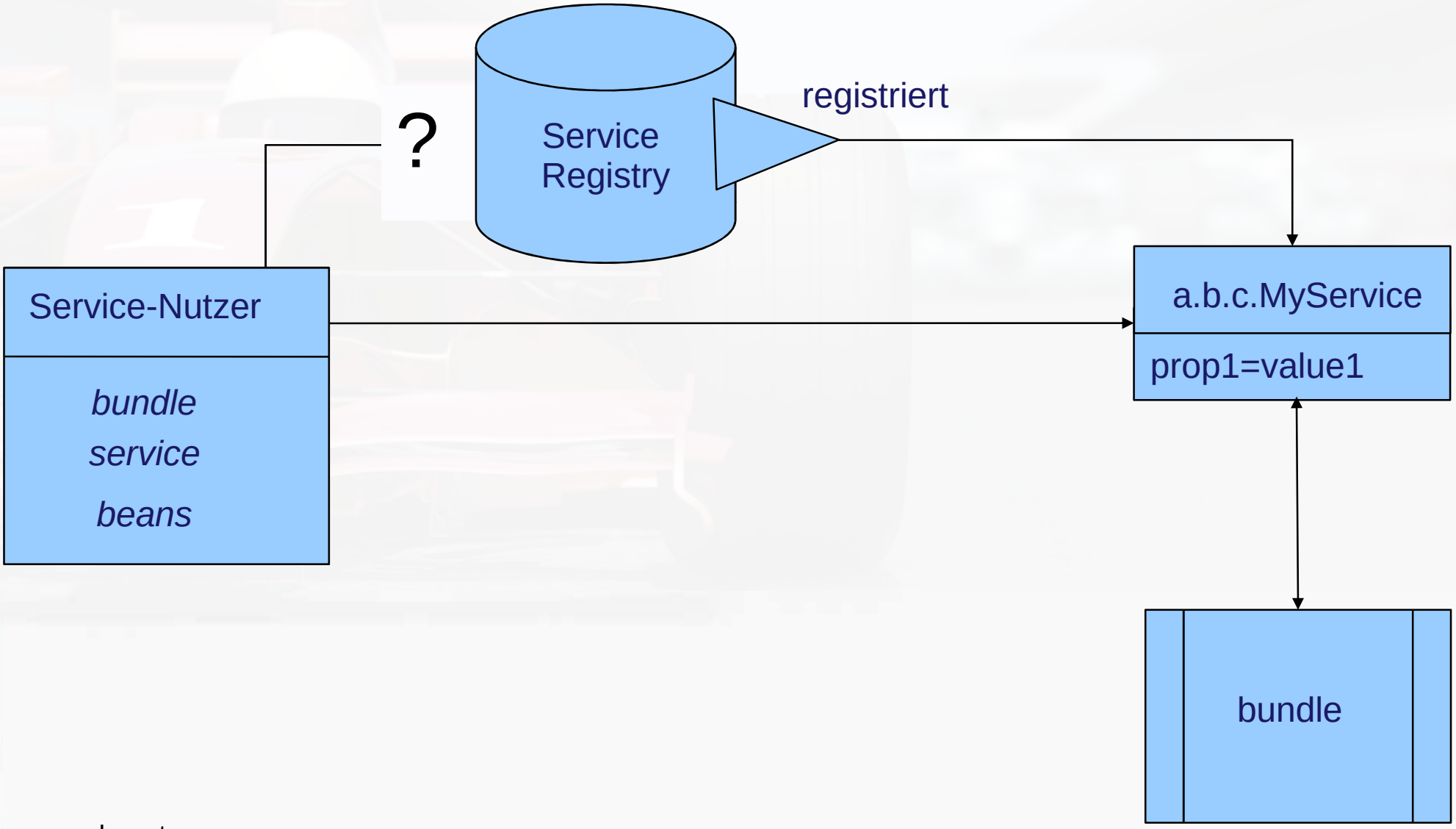
statisch

- keine Administration möglich

Wo sind wir ?

- Konfiguration
- Konfiguration und OSGi
  - OSGi Keywords
  - *Config Admin Service*
  - *Managed Service Factories*
  - *Security*
  - *Deklarative Konfiguration - Metatype Service*
- Konfiguration und *Declarative Services*
- Konfiguration und *Spring Dynamic Modules*

OSGi Keywords



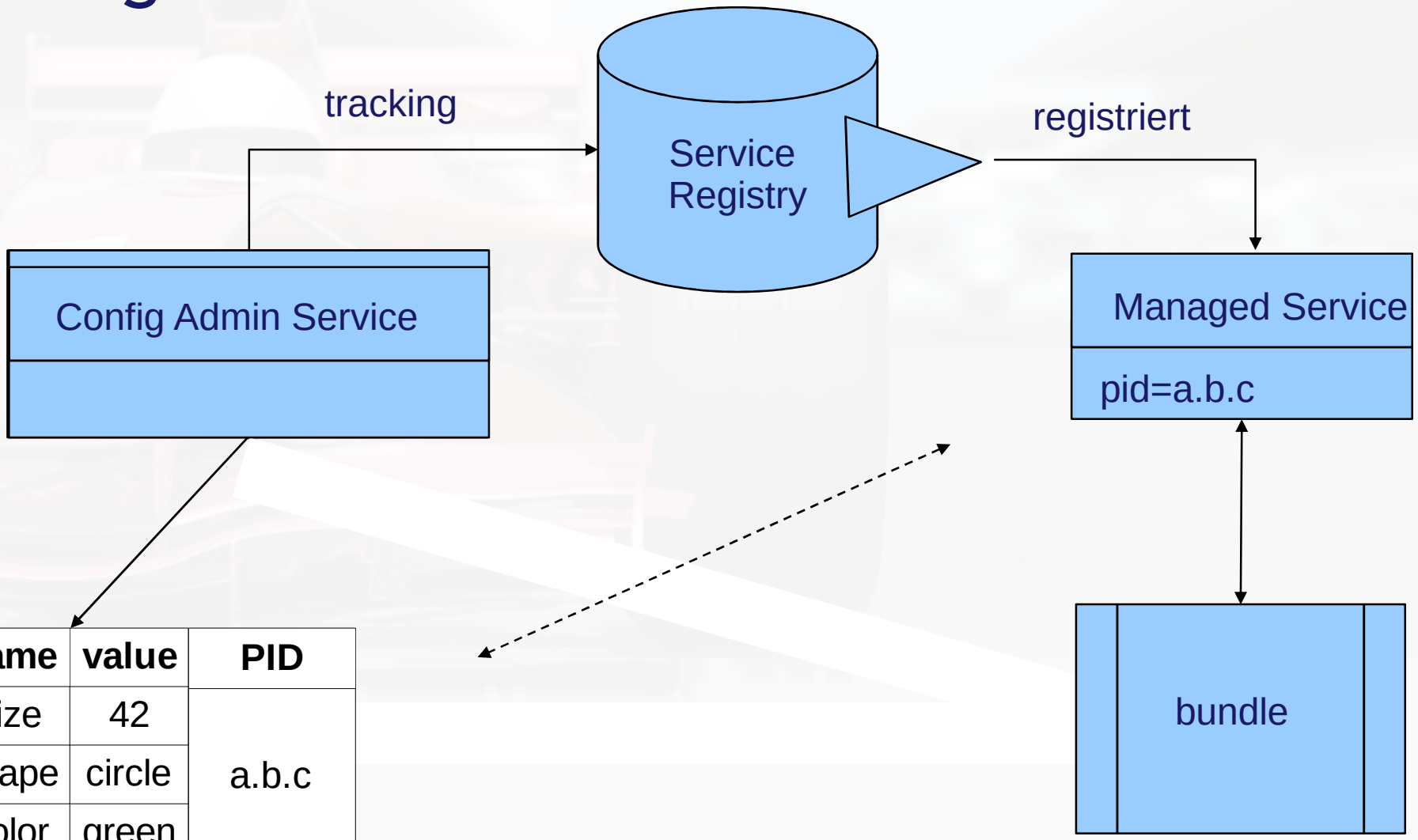
Konfiguration in OSGi

- Wie sind Konfigurationen und Services gekoppelt?
  - grundlegende Mechanismen
  - Wie wird Lifecycle von Services unterstützt?
  - Wie gelangen Konfigurationen zu Services?
- Wie werden Konfigurationen beschrieben?
  - programmatisch und deklarativ
- Auswirkung von Konfigurationsänderungen zur Laufzeit

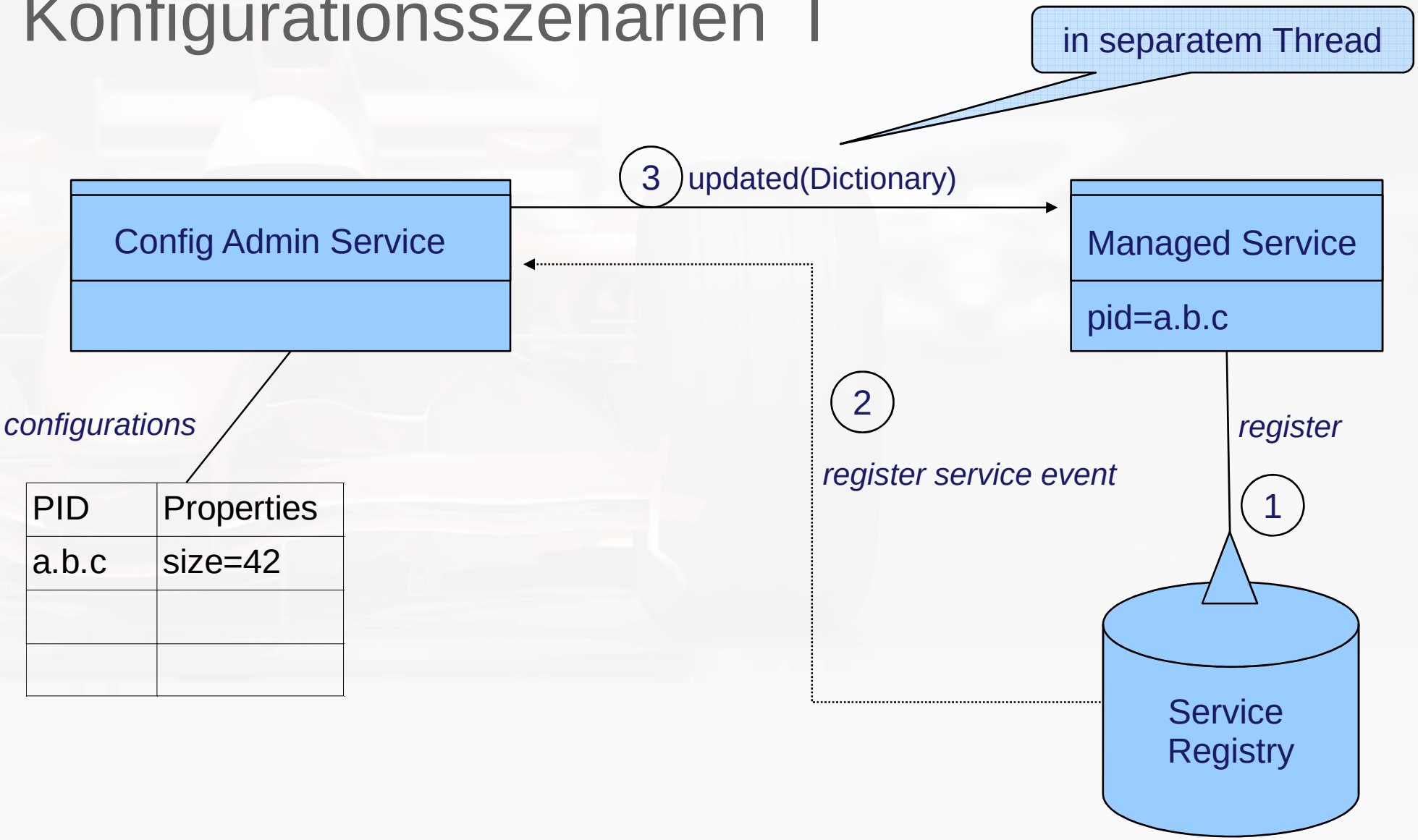
Wo sind wir ?

- Konfiguration
- Konfiguration und OSGi
  - OSGi Keywords
  - *Config Admin Service*
  - *Managed Service Factories*
  - *Security*
  - *Deklarative Konfiguration - Metatype Service*
- Konfiguration und *Declarative Services*
- Konfiguration und Spring *Dynamic Modules*

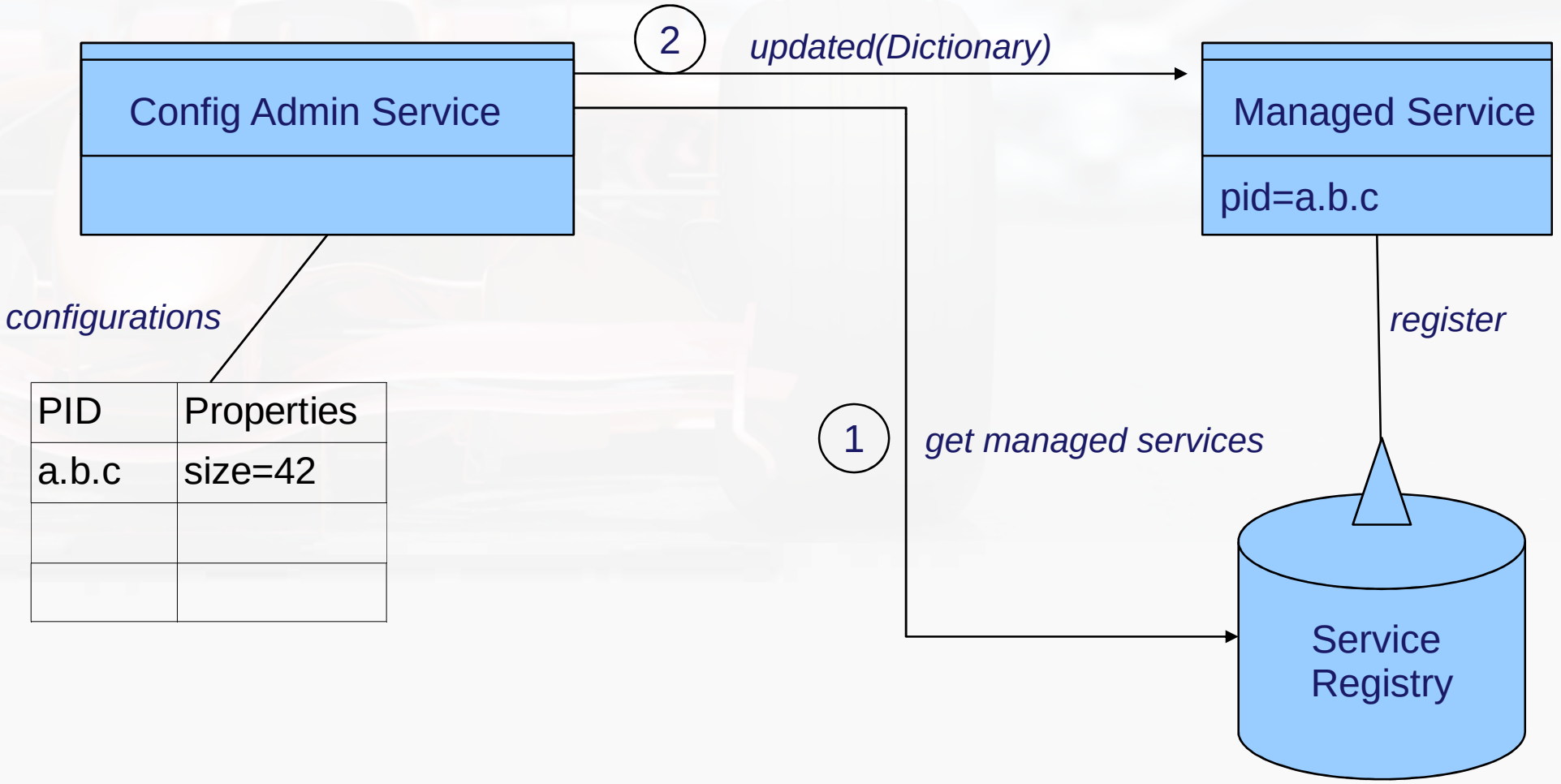
Konfiguration in OSGi



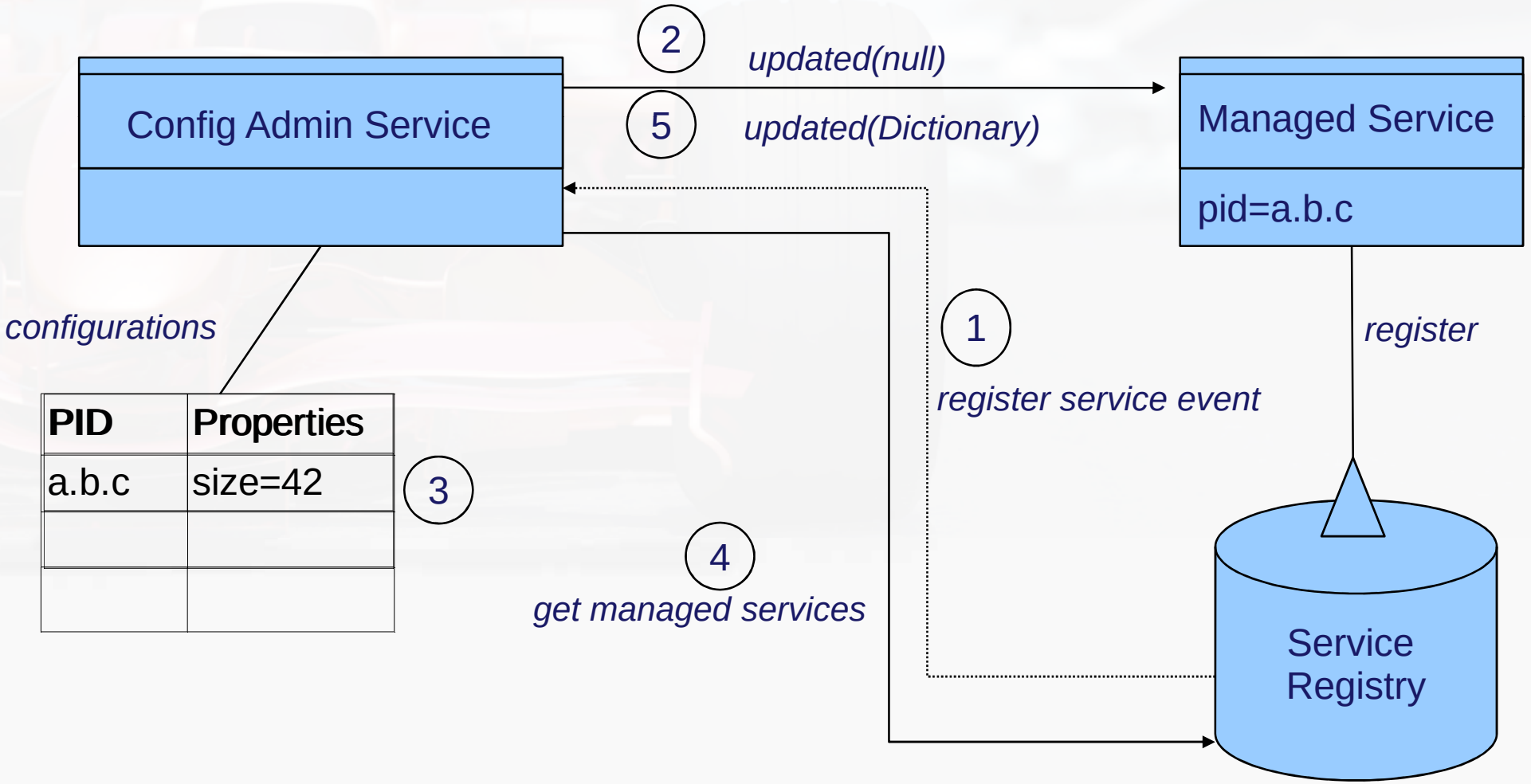
Konfigurationsszenarien I



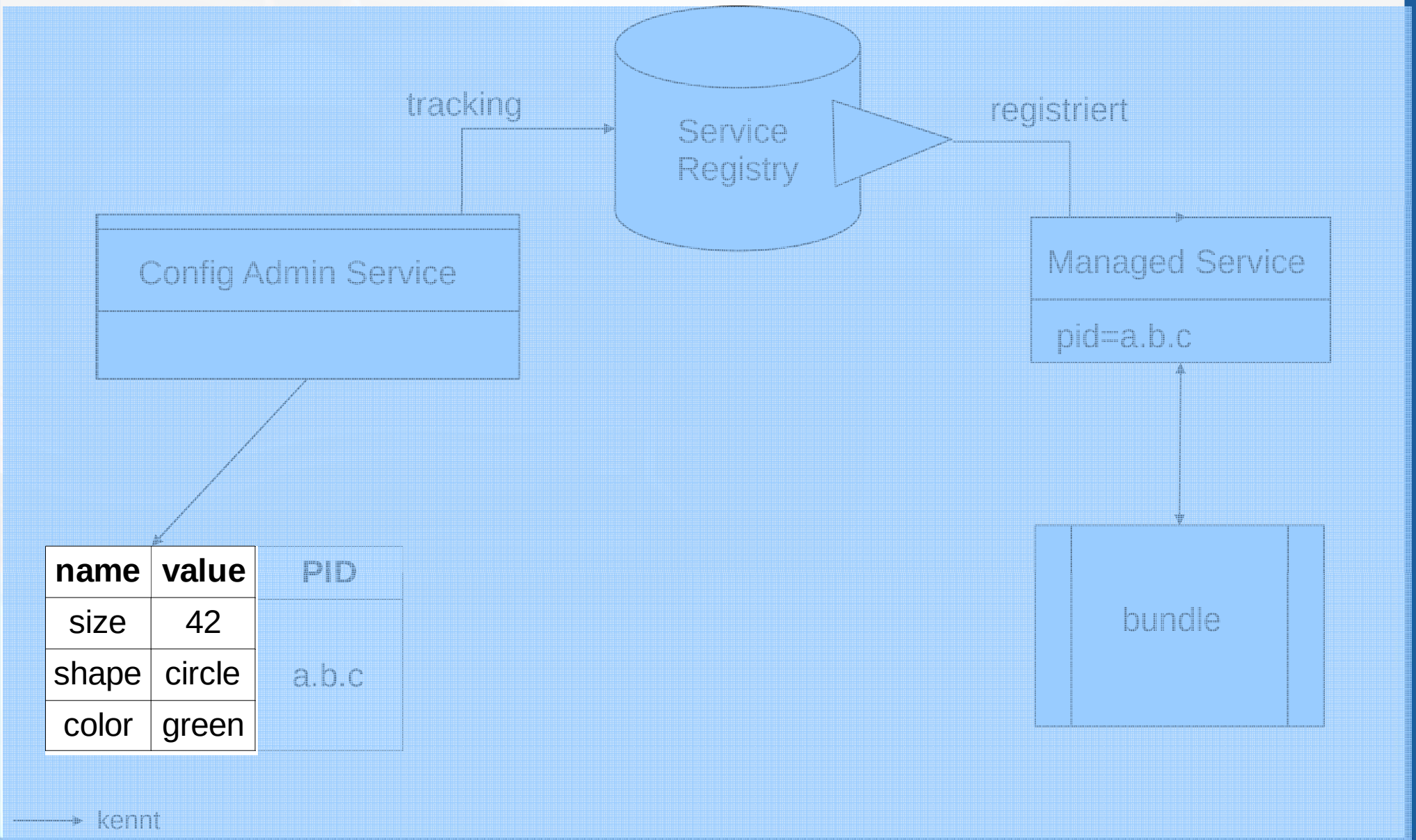
Konfigurationsszenarien II



Konfigurationsszenarien III



Wo sind wir ?


tracking
Service Registry
registriert
Config Admin Service
Managed Service
pid=a.b.c
bundle
name
value
size
42
shape
circle
color
green
PID
a.b.c
kennt

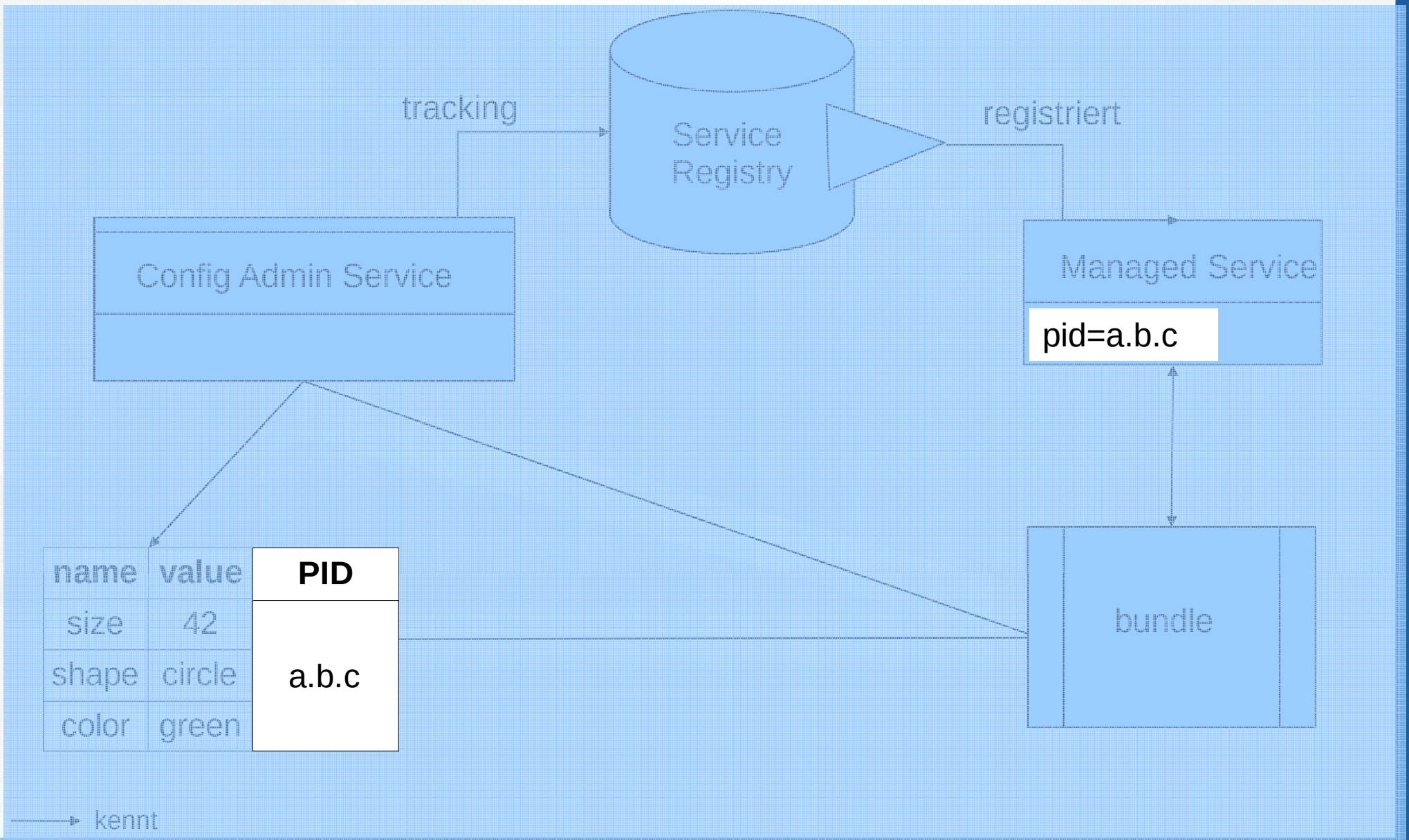
Configuration Object (I)

- Konfigurationsinformationen
  - heißen *Configuration Object*
  - werden als *java.util.Dictionary* repräsentiert
  - bestehen aus Primitiven- und Java-Simple Typen
  - .... oder Arrays oder Vektoren davon

Configuration Object (II)

- verwaltet einen Satz Konfigurationsinformationen
- wird programmatisch erzeugt
  - siehe *Config Admin Service*
- wird deklarativ erzeugt
  - siehe *Metatype Service*

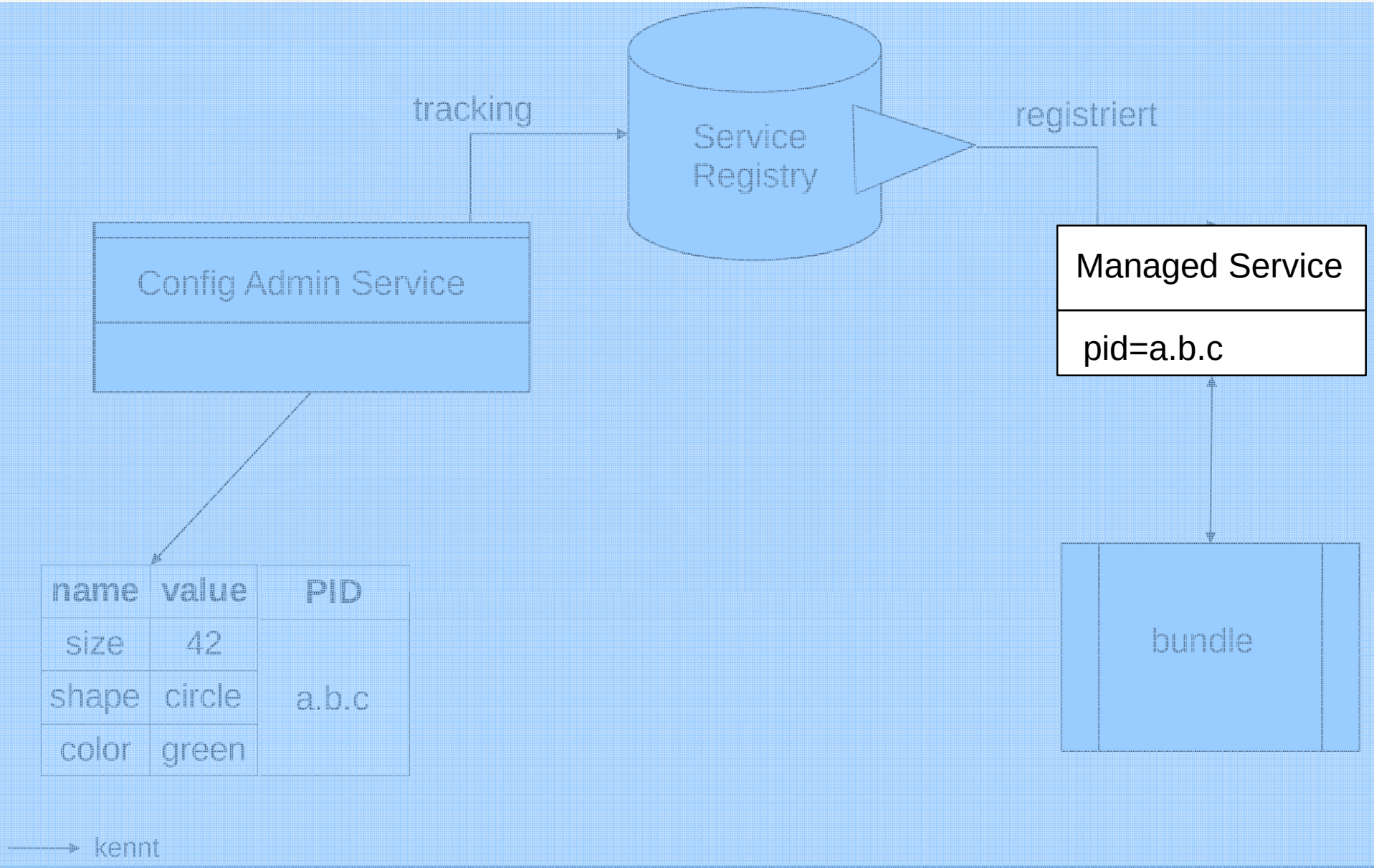
Wo sind wir ?


tracking
Service Registry
registriert
Config Admin Service
Managed Service
pid=a.b.c
name
value
size
42
shape
circle
color
green
PID
a.b.c
bundle
kennt

PID (Persistent Identity)

- identifiziert ein *Configuration Object*
- identifiziert *Managed Services*
- verbindet alle Beteiligten in einem Konfigurationsszenario
- ist eindeutig für alle *Managed Services*
  - maximal ein *Configuration Object* pro *PID*
  - maximal ein *Managed Service* pro *PID*
  - durch Spezifikation eingefordert

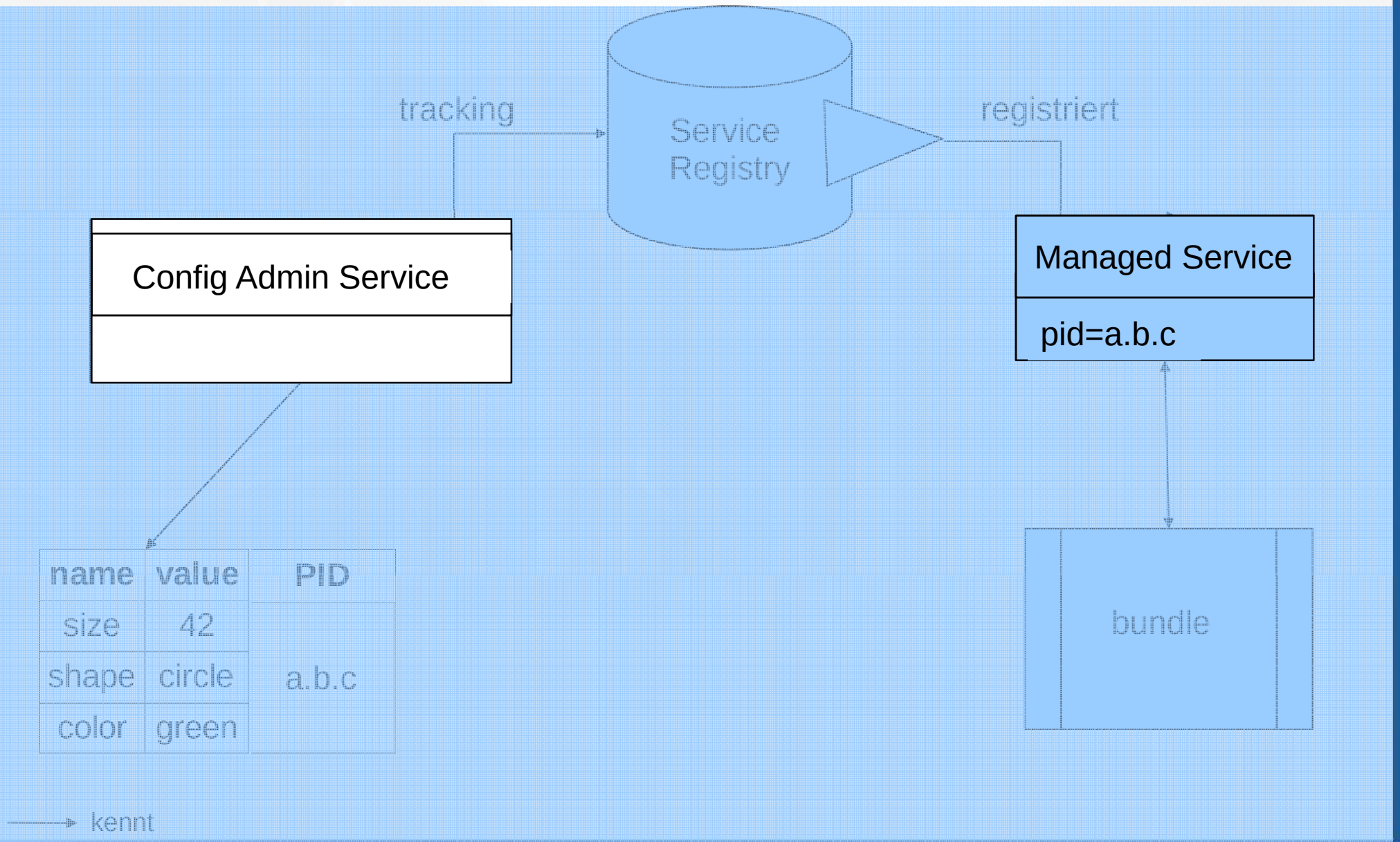
Wo sind wir ?


tracking
Service Registry
registriert
Config Admin Service
Managed Service
pid=a.b.c
bundle
name
value
PID
size
42
shape
circle
a.b.c
color
green
kennt

Managed Services

- sind OSGi Services, die sich auf Konfigurationen beziehen
- konsumieren Konfigurationen des *Config Admin Service*
- implementieren das IF *Managed Service*
  - Methoden für *Configuration Lifecycle*
- referenzieren die Konfiguration mittels *PID*
  - *PID* ist *Service Property*

Wo sind wir ?


tracking
Service Registry
registriert
Config Admin Service
Managed Service
pid=a.b.c
name
value
PID
size
42
shape
circle
a.b.c
color
green
bundle
kennt

Config Admin Service

- ist ein Service der OSGi Spec
- verwaltet *Configuration Objects*
  - *anlegen, löschen, verändern, finden*
  - *bietet Programmierschnittstelle für diese Aufgaben*
- verwaltet Bezug zwischen *Configuration Objects* und *Managed Services*
  - stellt initialen Bezug her
  - kontrolliert die Auswirkung des Lebenszyklus von *Configuration Objects*
- persistiert *Configuration Objects*

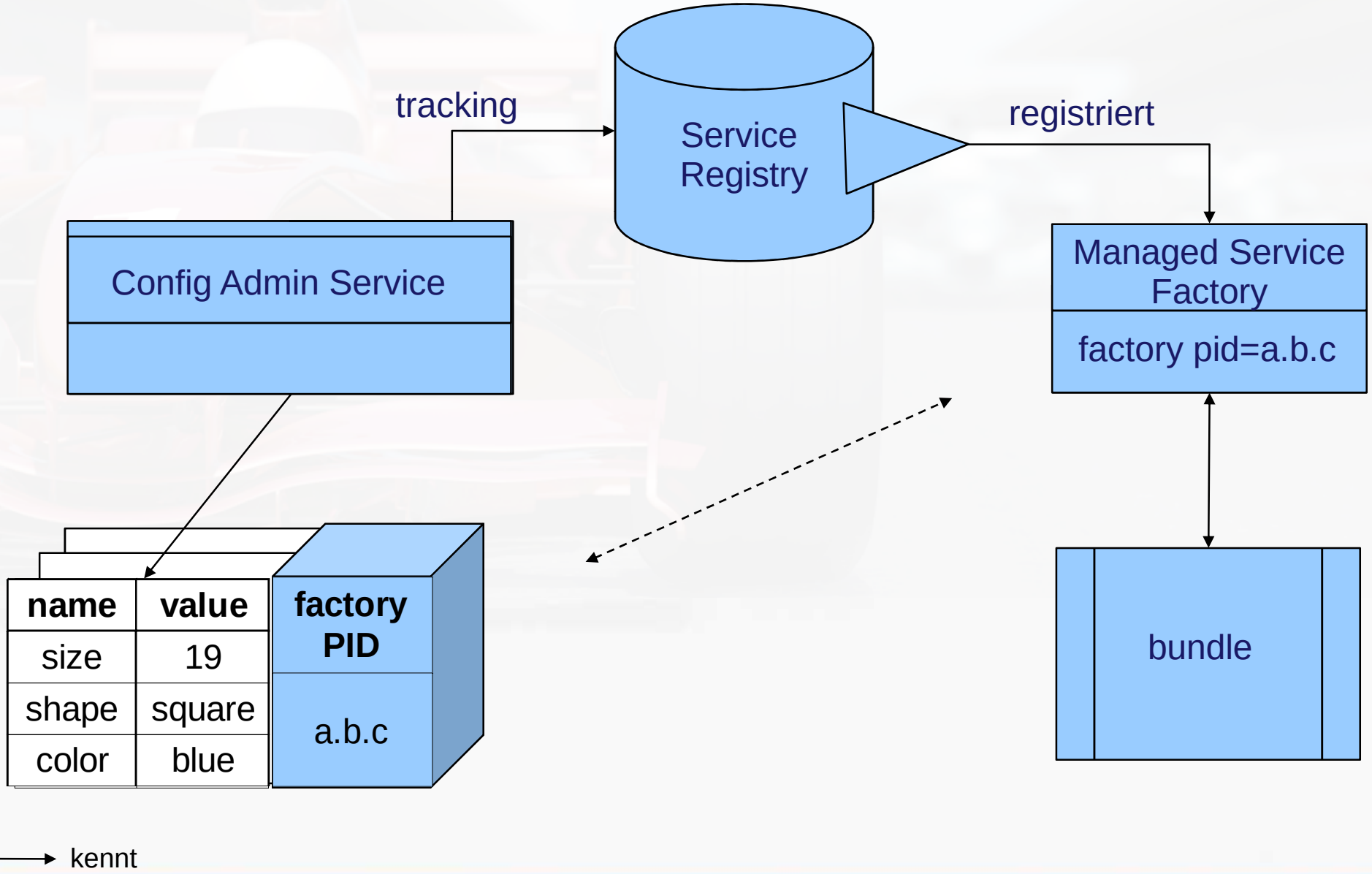
Wo sind wir ?

- Konfiguration
- Konfiguration und OSGi
  - OSGi Keywords
  - *Config Admin Service*
  - *Managed Service Factories*
  - *Security*
  - *Deklarative Konfiguration - Metatype Service*
- Konfiguration und *Declarative Services*
- Konfiguration und *Spring Dynamic Modules*

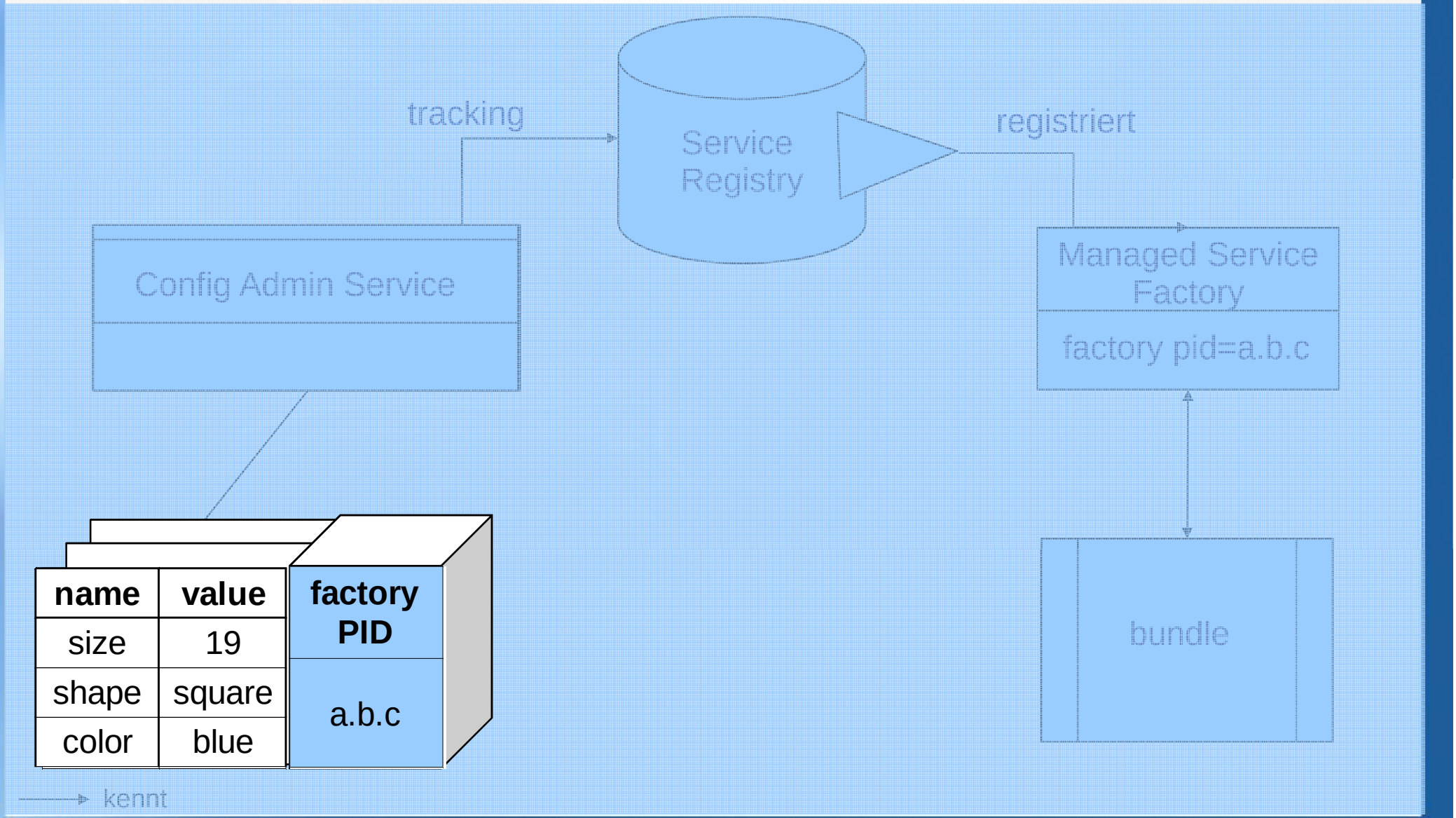
Managed Service Factory - Ausgangsszenario

- Ein Service wird mehrfach mit jeweils eigener Konfiguration instanziert
- Alle Konfigurationen sind gleichartig
  - werden gleichartig verarbeitet

Managed Service Factories



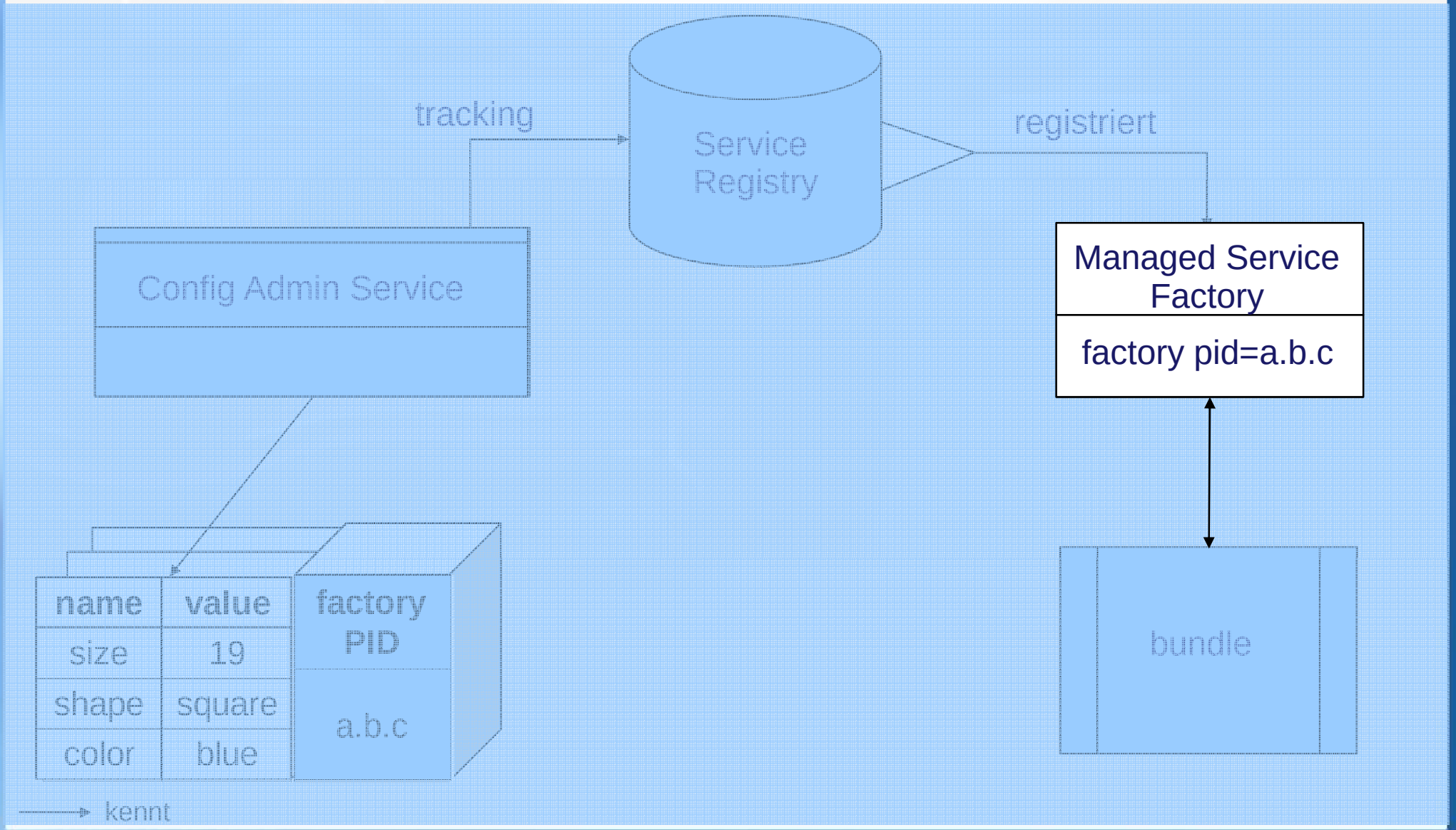
Wo sind wir ?


tracking
Service Registry
registriert
Config Admin Service
Managed Service Factory
factory pid=a.b.c
bundle
name
value
size
19
shape
square
color
blue
factory PID
a.b.c
kennt

Factory Configuration

- *Factory Configuration*
  - enthalten mehrere gleichartige *Configuration Objects*
  - werden durch eine *factory PID* identifiziert
- *Config Admin Service unterstützt*
  - *Factory Configurations*
  - Generierung einer *PID* pro *Configuration*

Wo sind wir ?


tracking
Service Registry
registriert
Config Admin Service
Managed Service Factory
factory pid=a.b.c
name
value
factory PID
size
19
shape
square
a.b.c
color
blue
bundle
kennt

Managed Service Factory

- *Managed Service Factory*
  - referenziert eine *Factory Configuration*
    - *factory PID*
  - ist Adressat aller *Configuration Objects* einer *Factory Configuration*
  - erwartet eigene *PID* pro *Configuration Object*
  - *updated(String(PID), Dictionary)*

Wo sind wir ?

- Konfiguration
- Konfiguration und OSGi
  - OSGi Keywords
  - *Config Admin Service*
  - *Managed Service Factories*
  - *Security*
  - *Deklarative Konfiguration - Metatype Service*
- Konfiguration und *Declarative Services*
- Konfiguration und Spring *Dynamic Modules*

Security

- *Configuration Objects* sind
  - an ein Bundle gebunden
    - i.d.R. an das erzeugende Bundle
- *Bundles* haben keine Möglichkeit, die Konfiguration anderer Bundles zu manipulieren

Config Admin Service – Essentials

- *Configuration Objects*
  - *Singleton Configuration*
    - durch *PID* identifiziert
  - *Factory Configuration*
    - durch f*actory PID* identifiziert
- Adressat ist *Managed Service [Factory]*
- können programmatisch erzeugt werden
  - IF des *Config Admin Service*
- können deklarativ erzeugt werden
  - mit Hilfe des *Metatype Service*

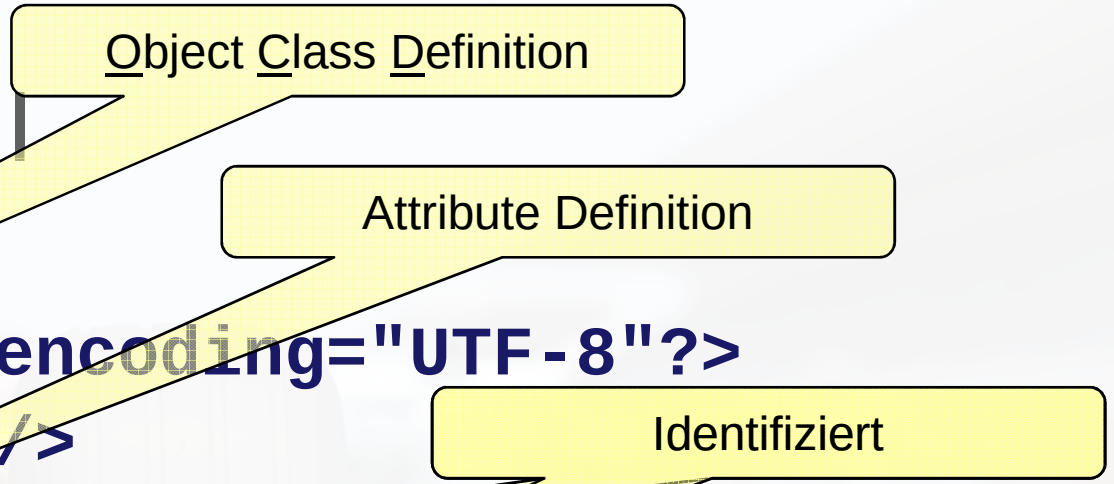
Wo sind wir ?

- Konfiguration
- Konfiguration und OSGi
  - OSGi Keywords
  - *Config Admin Service*
  - *Managed Service Factories*
  - *Security*
  - *Deklarative Konfiguration - Metatype Service*
- Konfiguration und *Declarative Services*
- Konfiguration und Spring *Dynamic Modules*

Deklarative Konfiguration - Erwartungshaltung

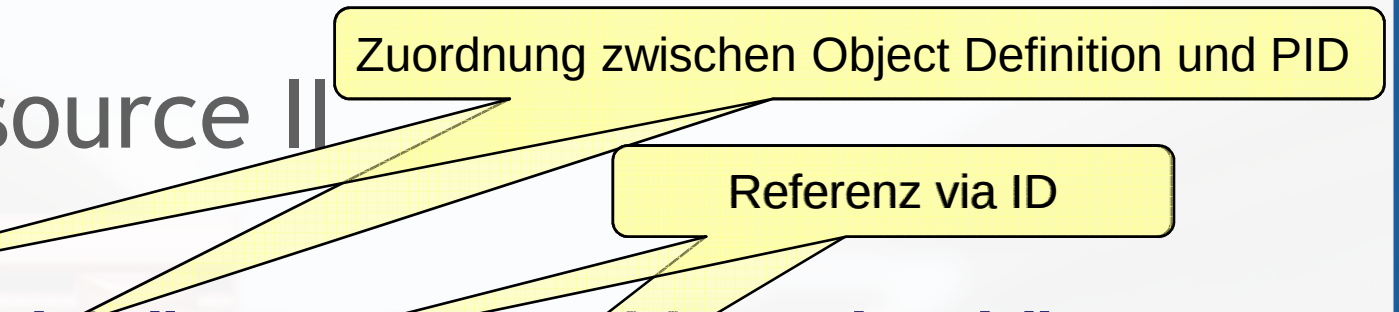
- Definition der Konfigurationsdaten
  - (key,value) – Pairs
  - einfache Typen für Werte
  - multiple einfache Typen als Array oder Vector
- Konfigurationsdatei
  - XML
- Verbindung zu *PID / factory PID*
- Beschreibung der Meta-Daten
  - Name, Typ, Kardinalitäten usw. zu Attributen
  - unterstützt Erfassung von Konfigurationen

Metatype Resource I



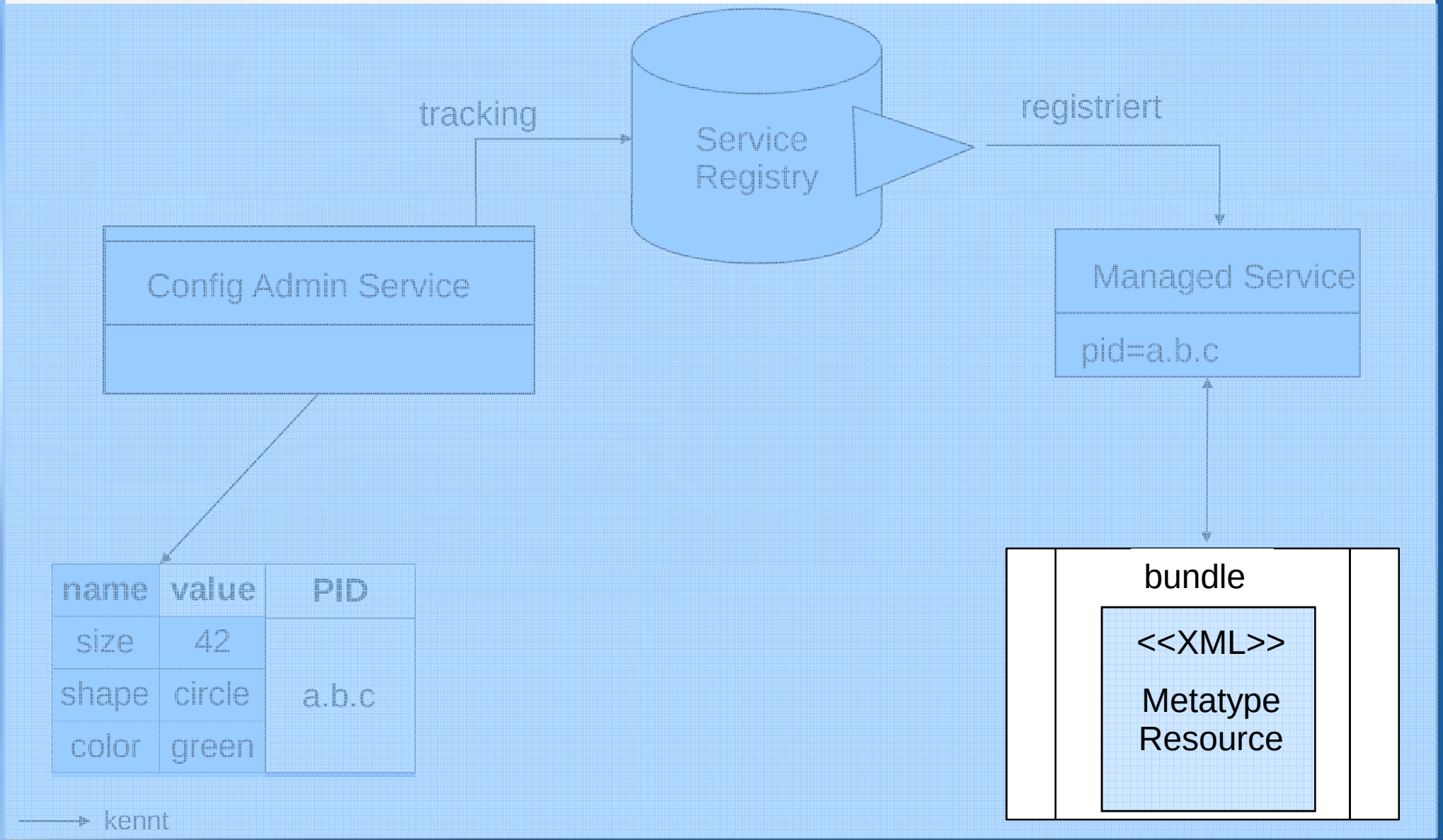

```
<?xml version="1.0" encoding="UTF-8"?>
<MetaData xmlns=... />
 <OCD name="person" ⏎
          id="2.5.6.6" description="...">
   <AD name="sn" id="2.5.4.4" type="String"/>
   <AD name="cn" id="2.5.4.3" type="String"/>
   <AD name="seeAlso" id="2.5.4.34" ⏎
                      type="String"⏎
                      cardinality="3" ⏎

 </OCD>
```

Metatype Resource II




```
<Designate pid="com.acme.addressbook">
  <Object ocdref="2.5.6.6"/>
    <Attribute adref="2.5.4.4" content="X"/>
    <Attribute adref="2.5.4.34">
            <Value>a@gmx.de</Value>
            <Value>x@gmx.de</Value>
            <Value>y@freenet.de</Value>
    </Attribute>
  </Designate>
</MetaData>
```

Konfiguration in OSGi



Metatype Service

- deklarative Beschreibung
  - in XML Ressource (sogen. *Meta Type Resource*)
  - liegen in OSGI-INF/Metatype eines Bundles
  - werden durch *Metatype Service* extrahiert
    - auch *fragments* werden untersucht
- programmatische Beschreibung
  - durch Implementierung der IF *MetaTypeProvider*
  - optional für *Managed Service [Factories]*
- werden durch Config Admin Service ausgewertet

Essentials - Metatype Service

- Metatype Resource
  - erlauben Deklaration von Konfigurationen
- Metadaten sind Basis für Adminstrations-konsolen
  - WebConsole, Knopflerfish Desktop

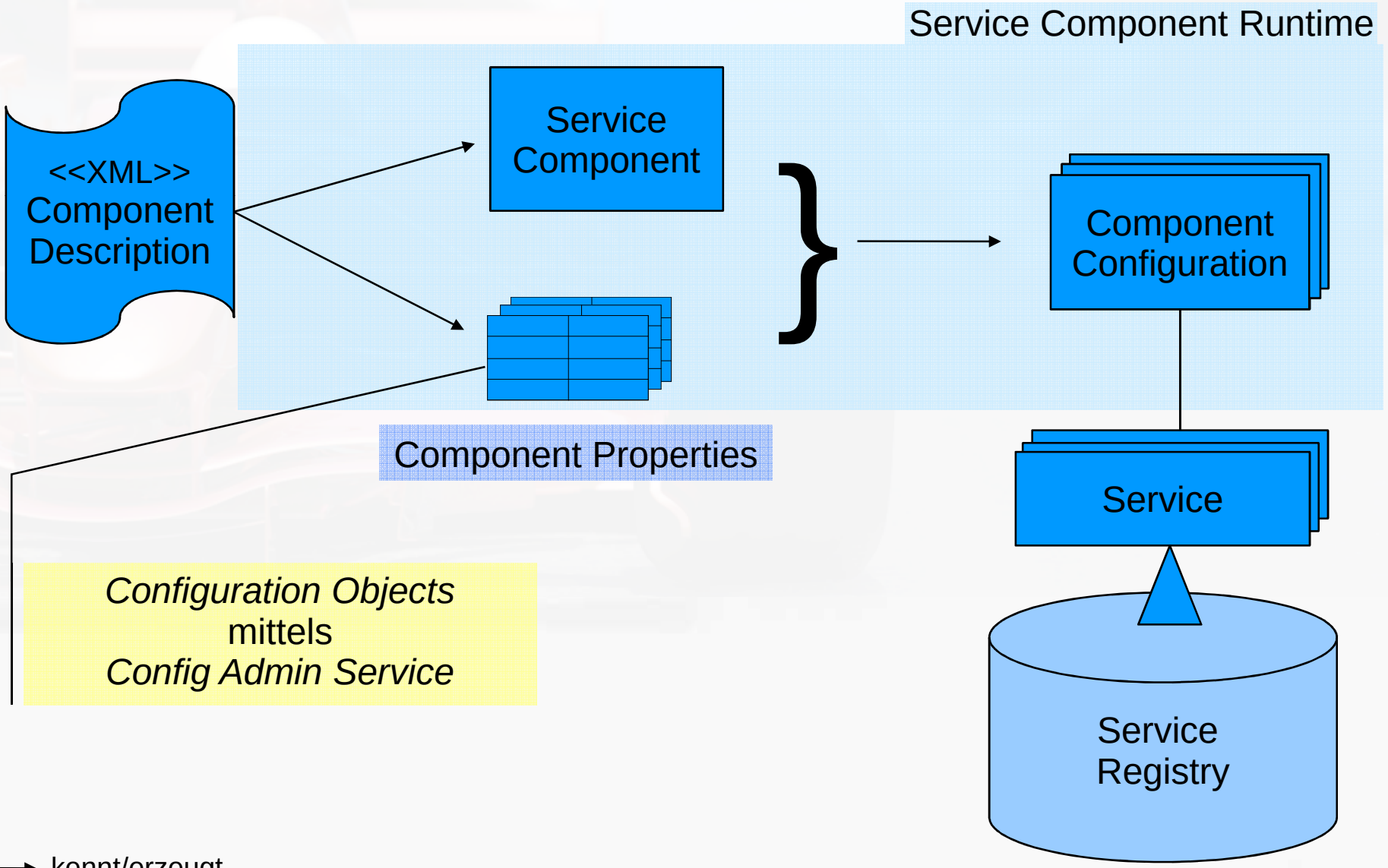
Agenda

- Konfiguration
- Konfiguration und OSGi
- Konfiguration und Declarative Services
  - *Declarative Services*
  - *Configuration Objects* und *und Component Configurations*
- *Konfiguration* und *Spring Dynamic Modules*
  - *Managed Properties*
  - *Managed Service Factories*

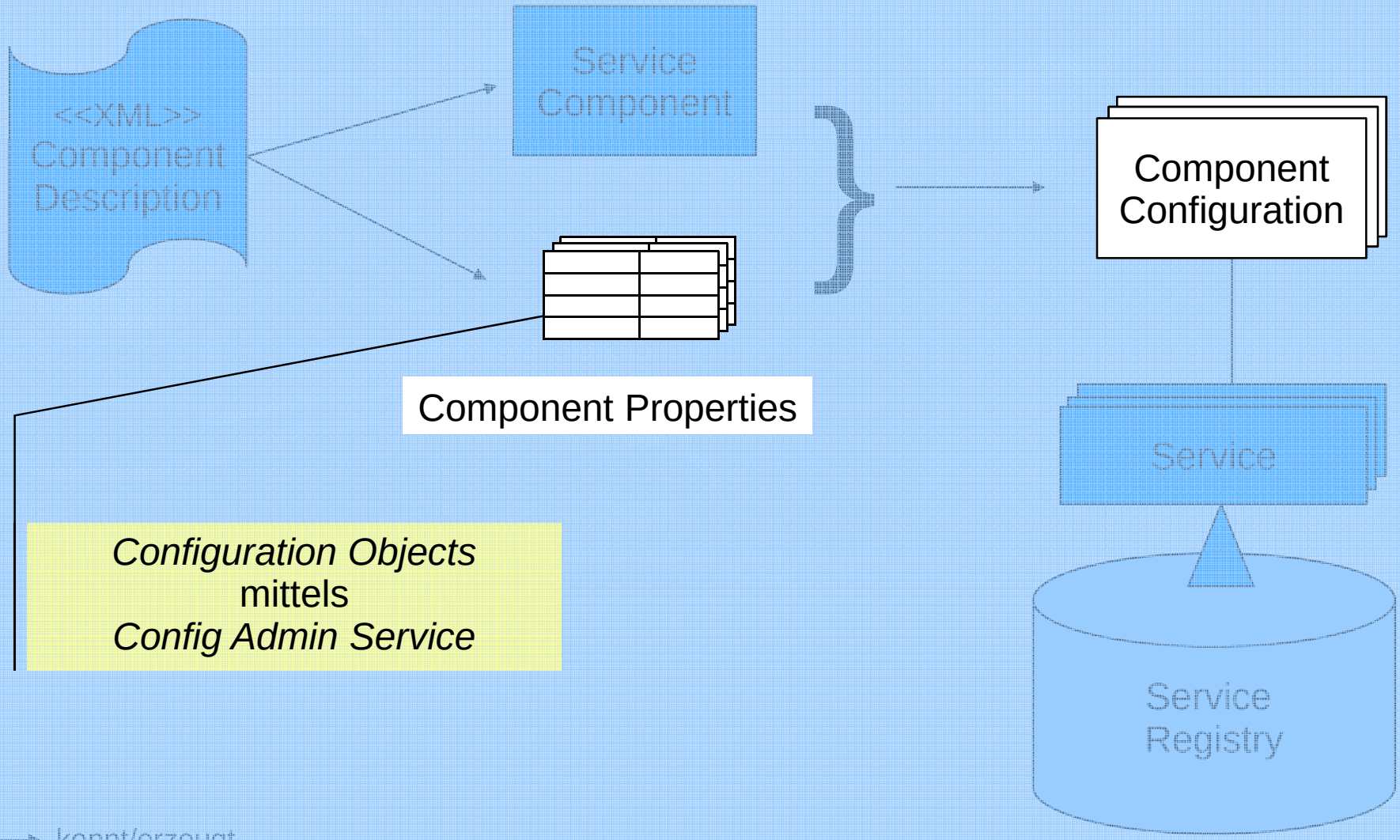
Component Configuration

- beschreibt Komponentenmodell für *OSGi*
- definiert Vertrag zwischen Services
- definiert Vertrag zwischen Service und Framework
- ist Bestandteil des *OSGi Service Compendiums*
- wird durch *Component Description* deklariert
  - XML

Declarative Services



Wo sind wir ?


<<XML>>
Component
Description
Service
Component
Component
Configuration
Component Properties
Service
Service
Registry
Configuration Objects
mittels
Config Admin Service
kennt/erzeugt

Component Configuration

- bezieht sich auf *Configuration Object*
  - mit *PID* gleich *component.name*
  - führt zu neuen *Component Properties*
  - ergänzt die Konfiguration der Komponente
    - Filterkriterien zur Referenz von Services
- Bei Aktualisierung des *Configuration Object*
  - wird die entsprechende *Component Configuration* deaktiviert
  - anschließend reaktiviert
    - Reaktivierung berücksichtigt neue Konfigurationssituation

Essentials - Declarative Services

- *Lifecycle* der *Component Configurations* sind an den der C*onfiguration Objects* gekoppelt
- ganzer Mechanismus der OSGi Konfiguration steht zur Verfügung
  - Eigenschaften von Komponenten
  - Abhängigkeiten zwischen Komponenten/Services

Wo sind wir ?

- Konfiguration
- Konfiguration und OSGi
- Konfiguration und *Declarative Services*
- Konfiguration und *Spring Dynamic Modules*
  - *Managed Properties*
  - *Managed Service Factories*

Spring DM (Dynamic Modules)

- Spring DM wird OSGi Standard in 4.2
  - RFC 124
- erweitert *Spring*
  - *Application Context* umfasst OSGi Kontext
  - dynamischer Service-Support
  - Integrationstests in OSGi
- ist (weiteres) Komponentenmodell für *OSGi*
  - definiert Vertrag zwischen Services
  - definiert Vertrag zwischen Service und Framework
  - definiert Vertrag zwischen Bean und Service

Wo sind wir ?

- Konfiguration
- Konfiguration und *OSGi*
- Konfiguration und *Declarative Services*
- Konfiguration und *Spring Dynamic Modules*
  - *Managed Properties*
  - *Managed Service Factories*

Spring DM - Managed Properties

Bean wird über Config Admin Service konfiguriert

PID

```
<bean id="managedComponent" class="MessageTank">
    <osgix:managed-properties
        persistent-id="com.xyz.messageservice"/>
    <property name="amount" value="100"/>
</bean>
```

```
public class MessageTank {
    private int amount;
    public int getAmount() {return this.amount; }
    public void setAmount(int amount)
                        { this.amount = amount; }
}
```

Spring DM – Aktualisierung der Konfiguration

- Änderung der *Configuration Objects*
  - werden implizit nach Erzeugung der Beans ignoriert
  - explizite Strategien sind auszuwählen
    - *container-managed*
      - *setter* der zu aktualisierenden Bean-Properties
    - *bean-managed*
      - *update-method*, die bei Aktualisierung aufgerufen wird

Wo sind wir ?

- Konfiguration
- Konfiguration und *OSGi*
- Konfiguration und *Declarative Services*
- Konfiguration und *Spring Dynamic Modules*
  - *Managed Properties*
  - *Managed Service Factories*

Spring DM – Managed Service Factories

```
<osgix:managed-service-factory id="data-msf"
        factory-pid="org.xyz.labX"
        update-strategy="bean-managed"
        update-method="refresh">

    <osgix:interfaces>
      <value>java.util.Queue</value>
    </osgix:interfaces>

    <bean class="com.xyz.ResizableQueue">
      <property name="size" value="100"/>
      <property name="fair" value="false"/>
    </bean>
</osgix:managed-service-factory>
```

factory PID

update strategy

service interface

Service Implementierung

Essentials - Spring DM

- *Spring DM* und *Config Admin Service* integrieren sich harmonisch
  - Integration ist konsistent und erwartungstreu
  - Erweiterungen sind folgerichtig
- *Spring DM* koppelt *Services* an *Configuration Objects*
  - vgl. *Declarative Services*

Essentials - Konfiguration in OSGi

- bietet jederzeitigen Eingriff in Wirkungsweise
- setzt stringent auf *OSGi* Mechanismen
- unterstützt *Singleton-* und *Factory Configurations*
- koppelt *Services* und *Configuration Objects*
  - Erweiterung von Komponentmodellen
    - *Declarative Services* und *Spring DM*

Bibliographie / Links

- OSGi Spec 4.1 / 4.2 Preview
  - http://www.osgi.org/download/osgi-4.2-early-draft.pdf
- OSGi Service Platform (Wütherich)
  - http://www.dpunkt.de/buecher/2635.html
- OSGi in Practice (Neil Barlett) – Free!
  - http://neilbartlett.name/blog/osgibook/
- Modular Java: Creating Flexible Applications with OSGi and Spring (Craig Walls)
  - http://www.pragprog.com/titles/cwosg/modular-java

Weitere Fragen?

Gesellschaft für
Informations- und
Kommunikationssysteme mbH

www.iks-gmbh.com

t.vogel@iks-gmbh.com
c.schmidt-casdorff@iks-gmbh.com