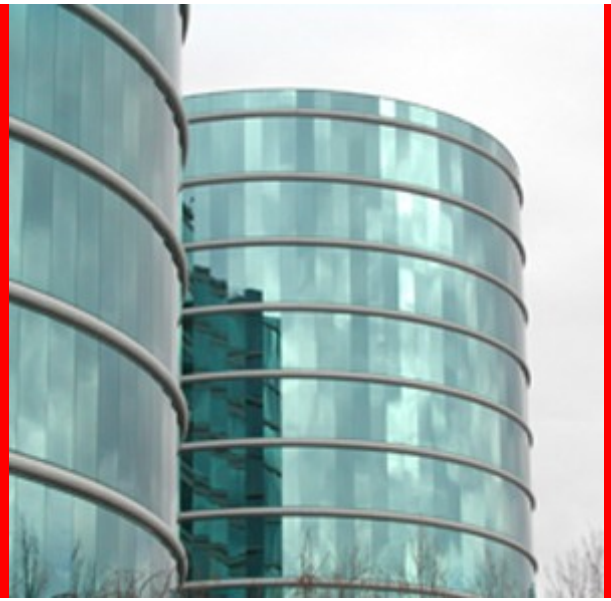
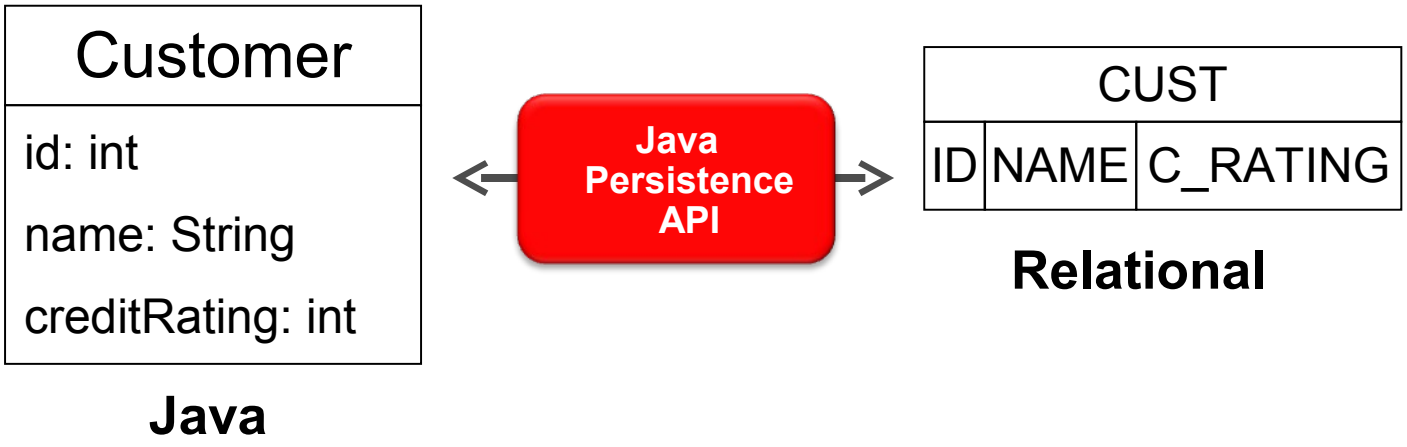


ORACLE®

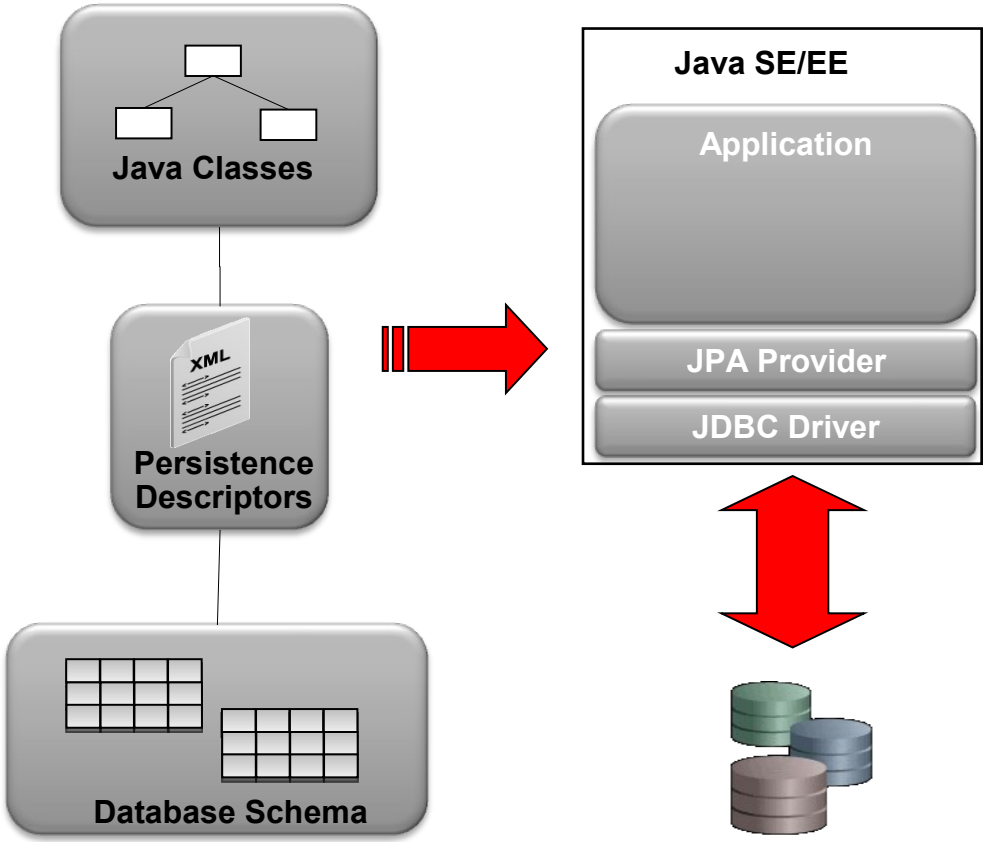
OSGi and Gemini JPA

Java Persistence: The Problem Space

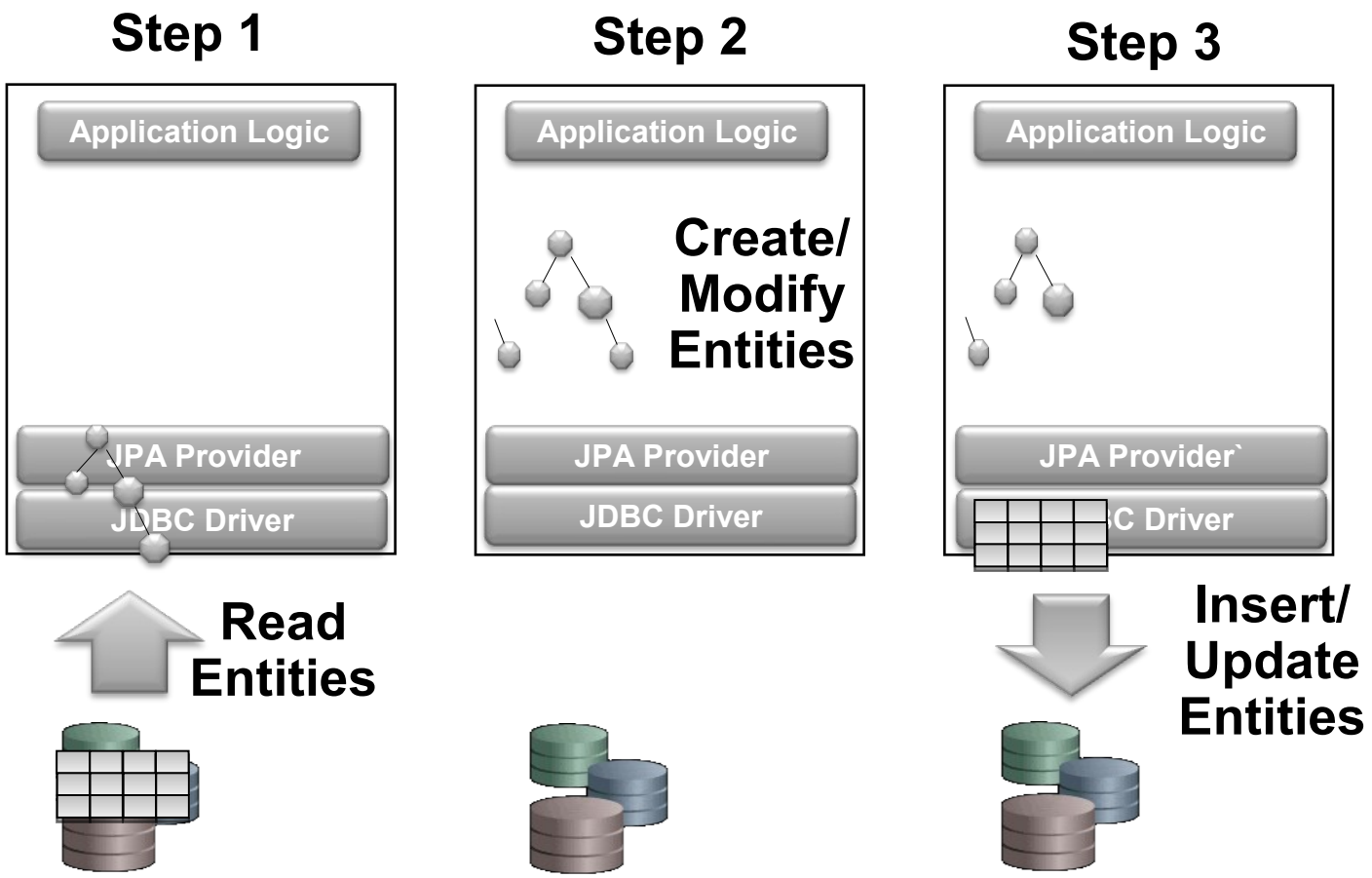


Java Persistence API (JPA)

- Defines:
  - How Java objects are stored in relational
  - A programmer API for reading, writing, and querying persistent Java objects (“Entities”)
  - A full featured query language in JP QL



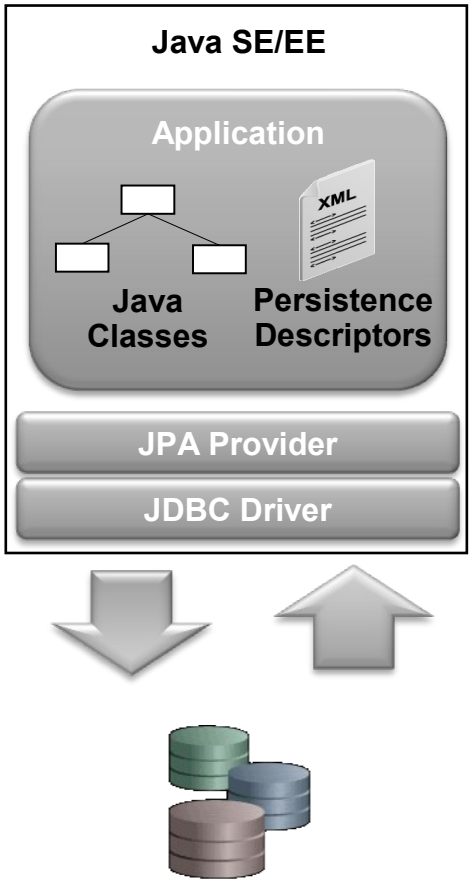
Mechanics of a JPA Application



Example JPA Client Code

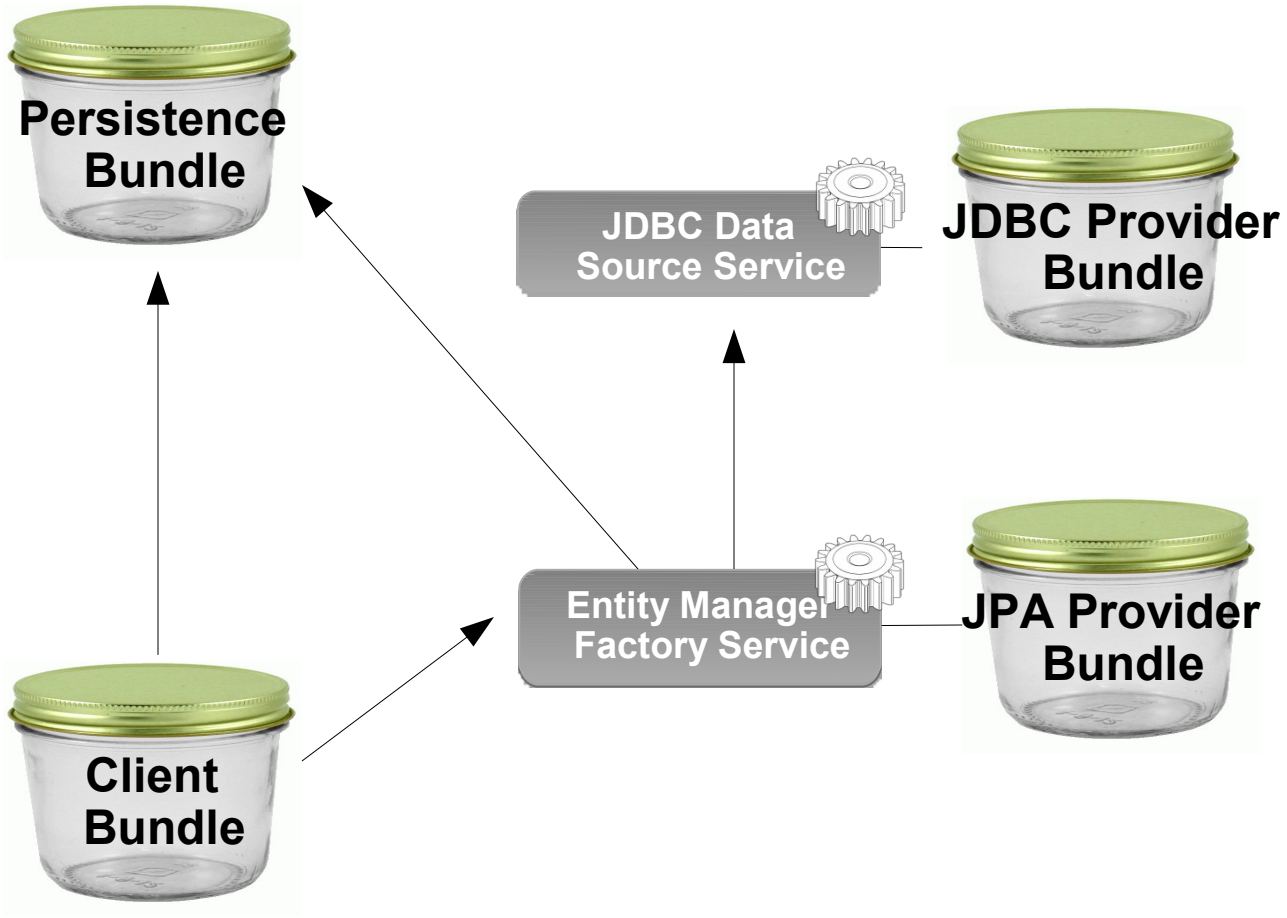
```
EntityManagerFactory emf = Persistence
        .createEntityManagerFactory("Accounting");
EntityManager em = emf.createEntityManager();
List<Employee> employees = em
    .createQuery("select e from Employee e")
    .getResultList();
…
em.close();
emf.close();
```

A non-OSGi Java SE/EE JPA Application



Set of Jars with Flat Classloader Space

An OSGi JPA Application



Set of OSGi Bundles and Services

The OSGi Java Persistence Service Specification

- Part of the OSGi Enterprise Specification
- Defines:
  - How persistence units (Entities and metadata) can be published
  - How client's can find persistence units
  - How database drivers are found using the OSGi JDBC spec
  - How JPA Providers can be made available in OSGi

Eclipse Enterprise Modules Project: 'Gemini'

- An open source project developing of implementations of the services defined in the OSGi Enterprise Specification including:
  - RFC 66 – Web Container
  - RFC 122 – Database Access
  - RFC 124 – Blueprint Service
  - RFC 139 – JMX Integration
  - RFC 142 – JNDI Integration
  - RFC 143 – JPA Integration
- Subproject of Eclipse Runtime (RT) Project

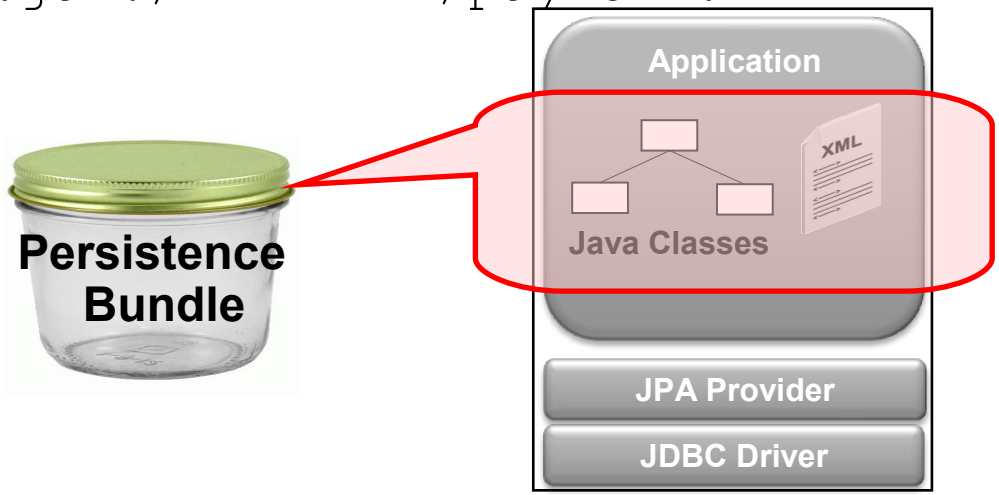
EclipseLink Project

- Provides JPA, JAXB, SDO, DBWS, and EIS persistence services
- Open source Eclipse project
- Founded by Oracle with the contribution of full TopLink source code and tests
- Based upon product with 12+ years of commercial usage
- Basis for Gemini OSGi JPA Service Implementation

Persistence Bundle

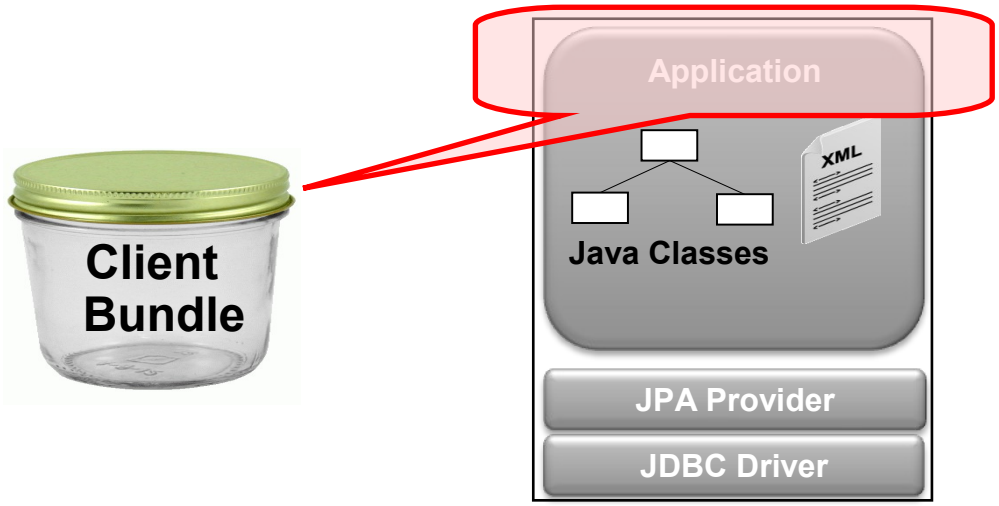
- Entity classes
- Persistence Descriptor declaration in manifest
  - Always processes `META-INF/persistence.xml` if present
  - Supports multiple descriptors and descriptors in jars:

```
Meta-Persistence: META-INF/accounting.xml,
payroll.jar!/META-INF/payroll.xml
```



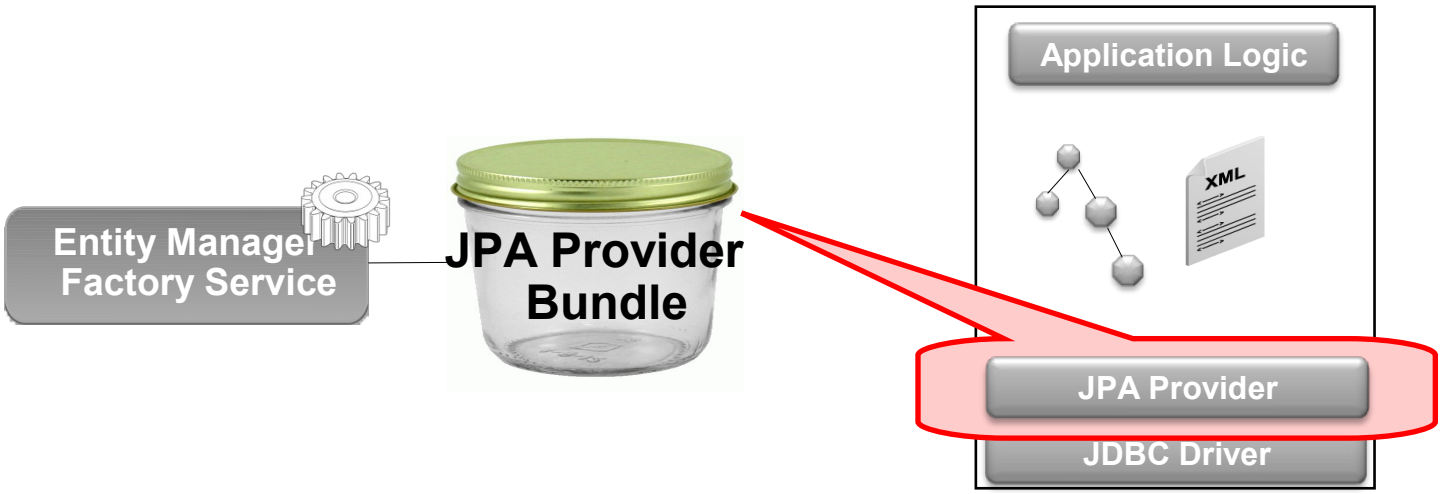
Client Bundle

- Application Logic
  - Must import Entity classes or require Persistence Bundle that exports Entity classes
  - Persistence Bundle may also be the Client Bundle—application logic and Entities in same bundle



JPA Provider Bundle

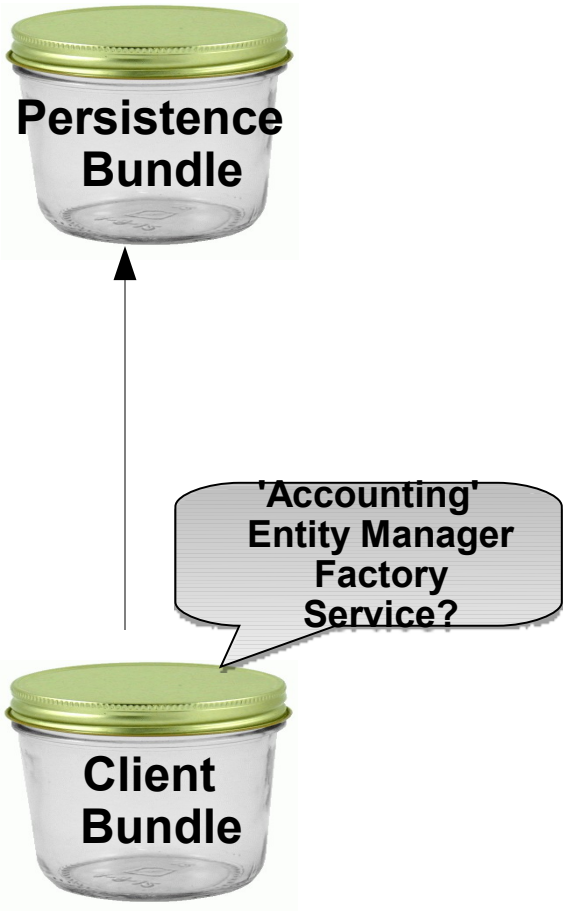
- JPA implementation providing a Persistence Provider and JPA Services



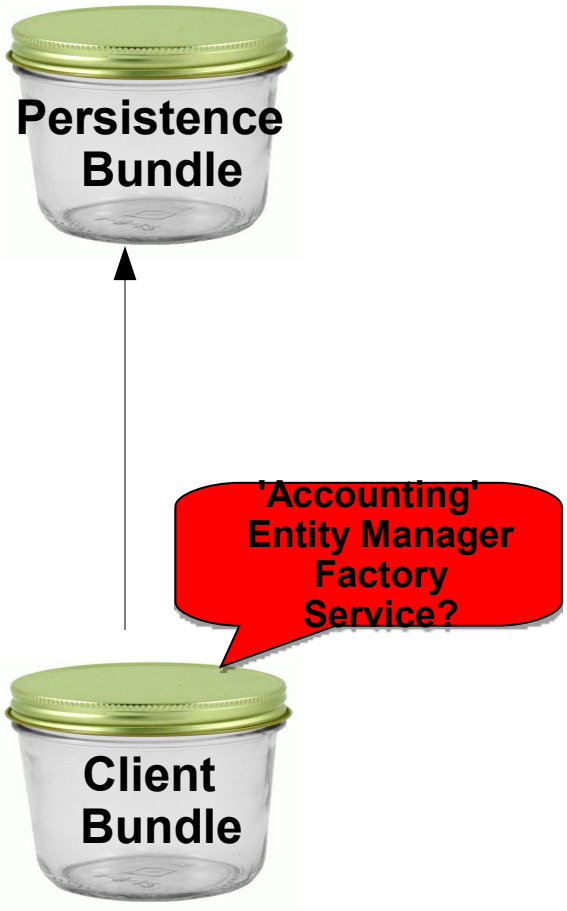
JDBC Provider Bundle

- JDBC driver implementation providing Data Source service.

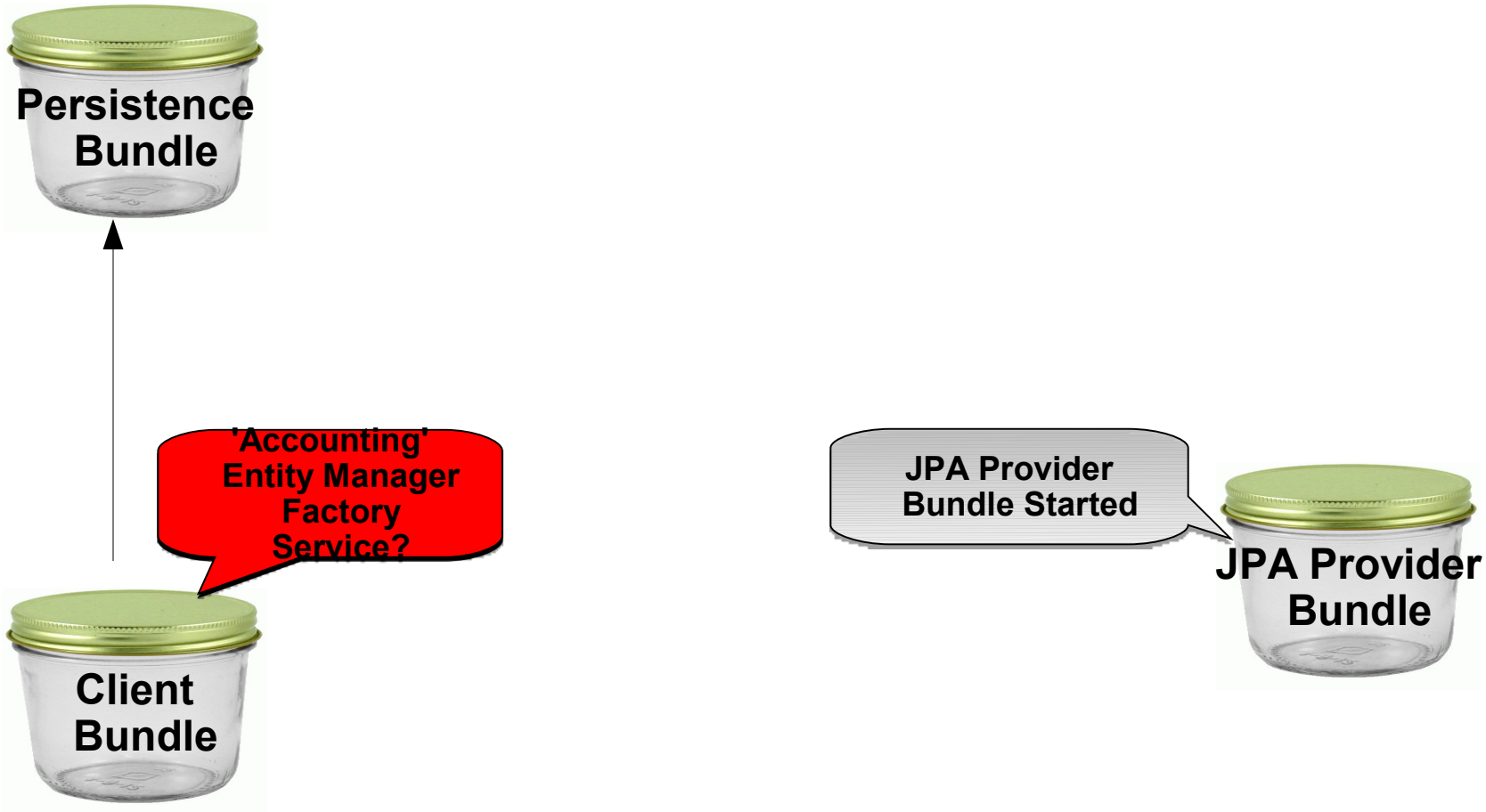
JPA OSGi Example



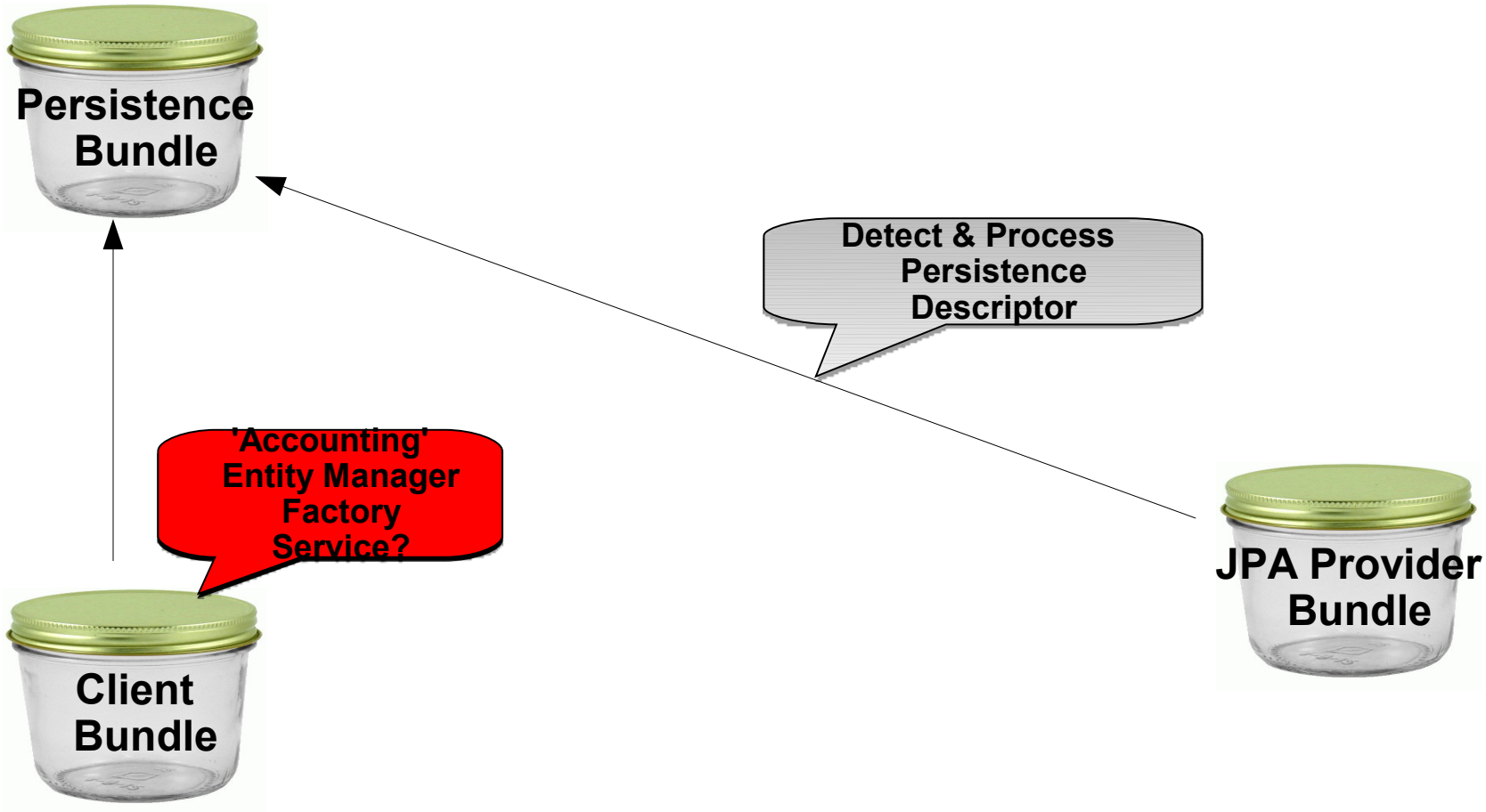
JPA OSGi Example



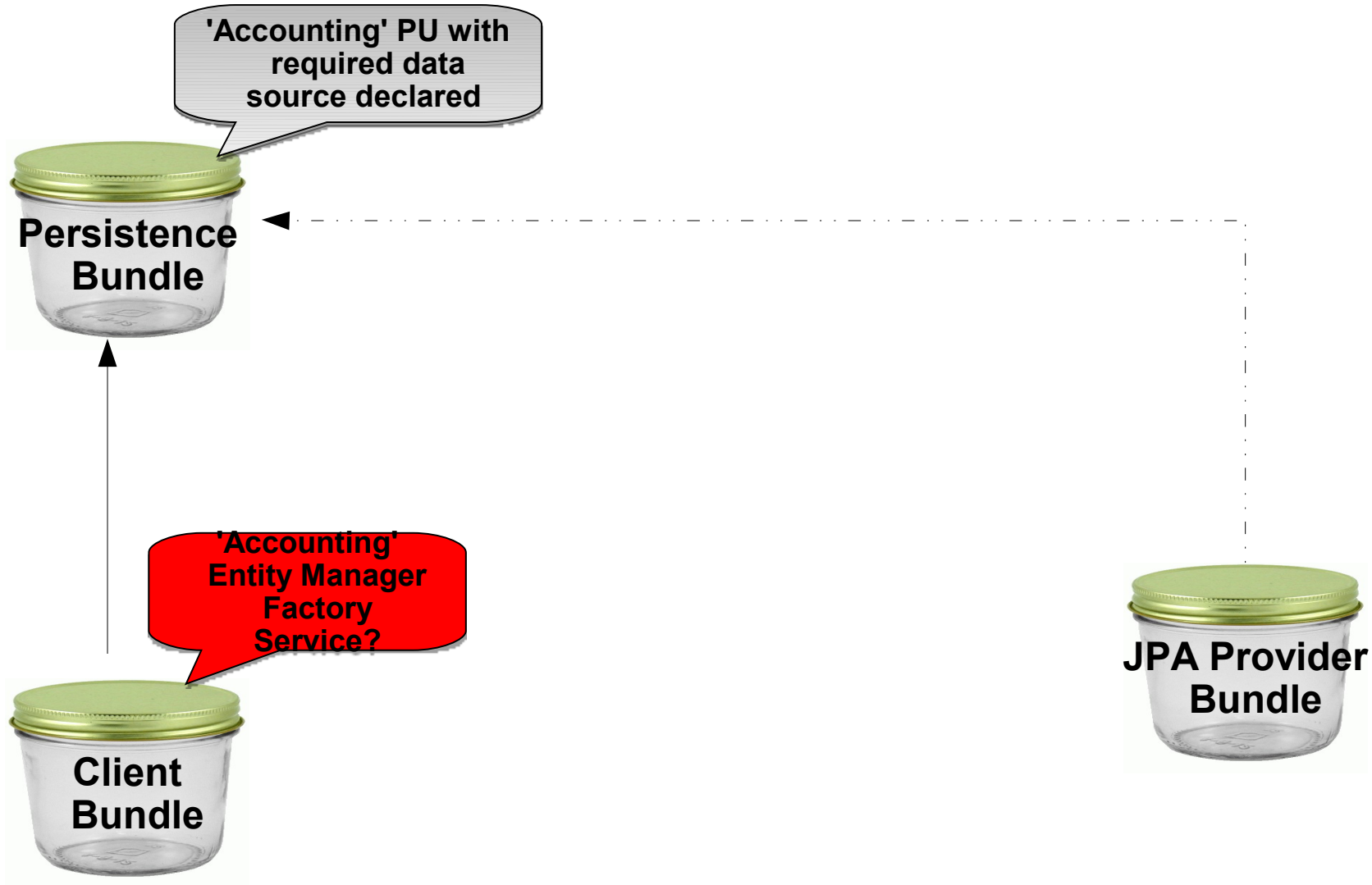
JPA OSGi Example



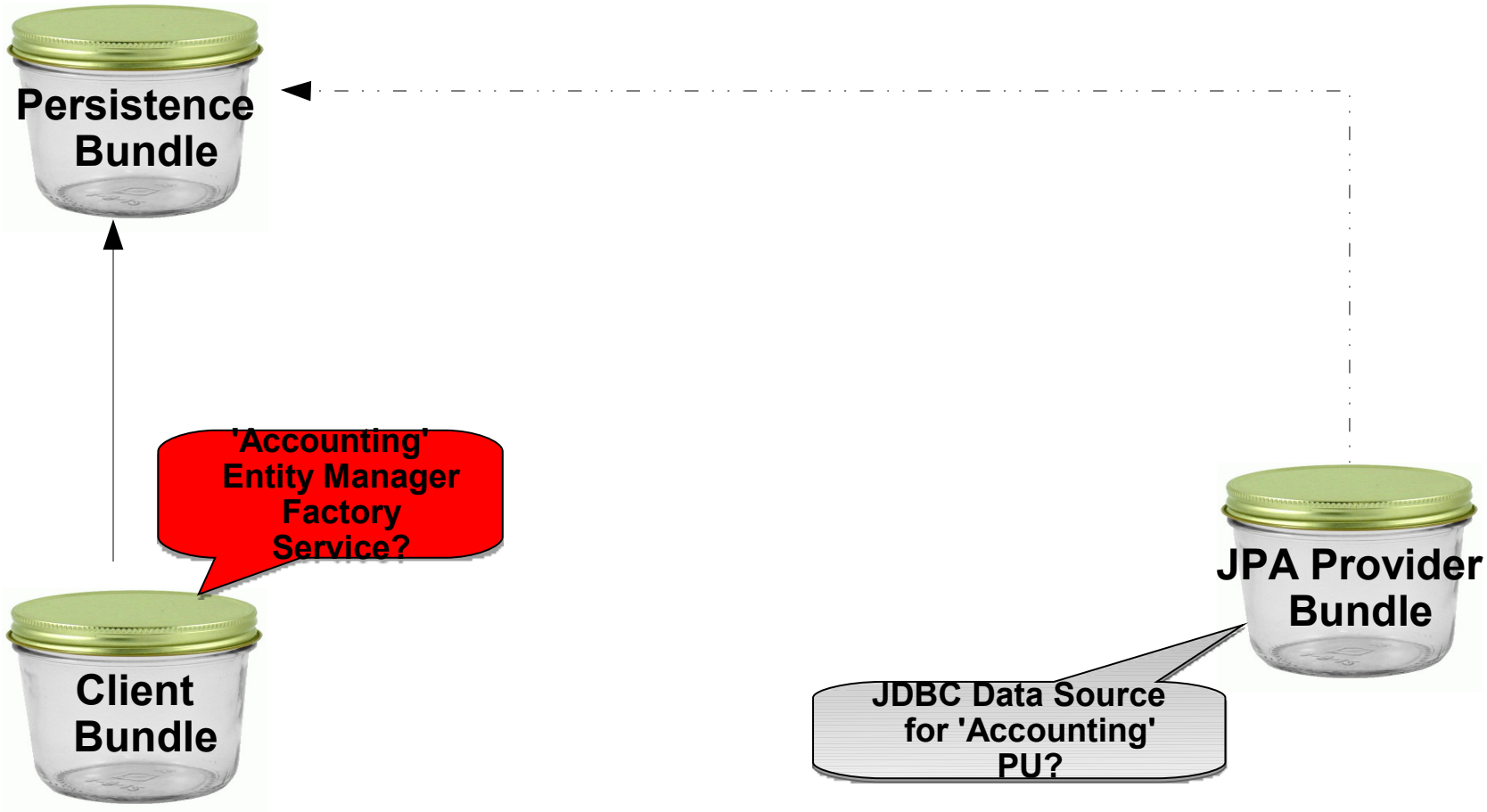
JPA OSGi Example



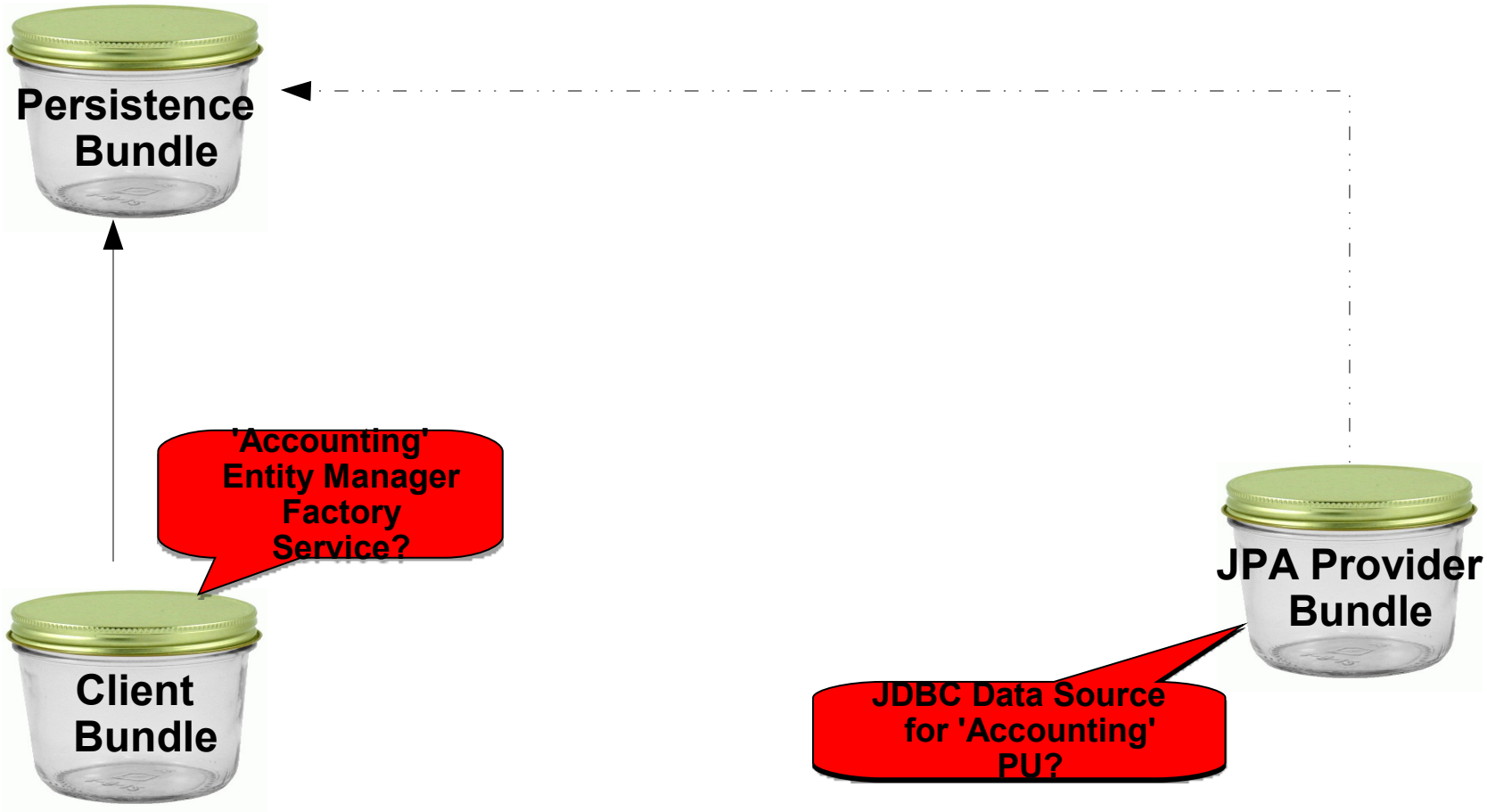
JPA OSGi Example



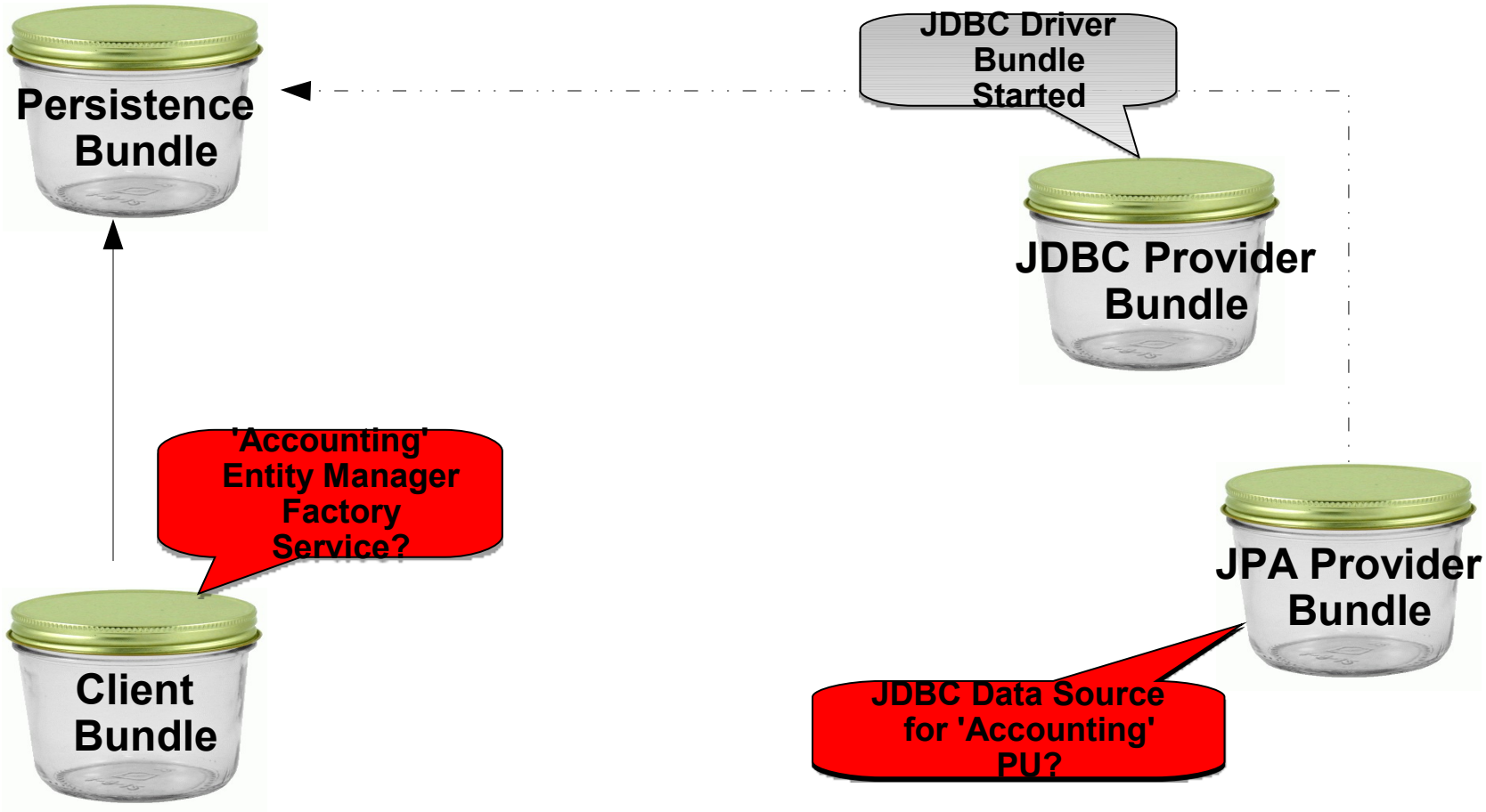
JPA OSGi Example



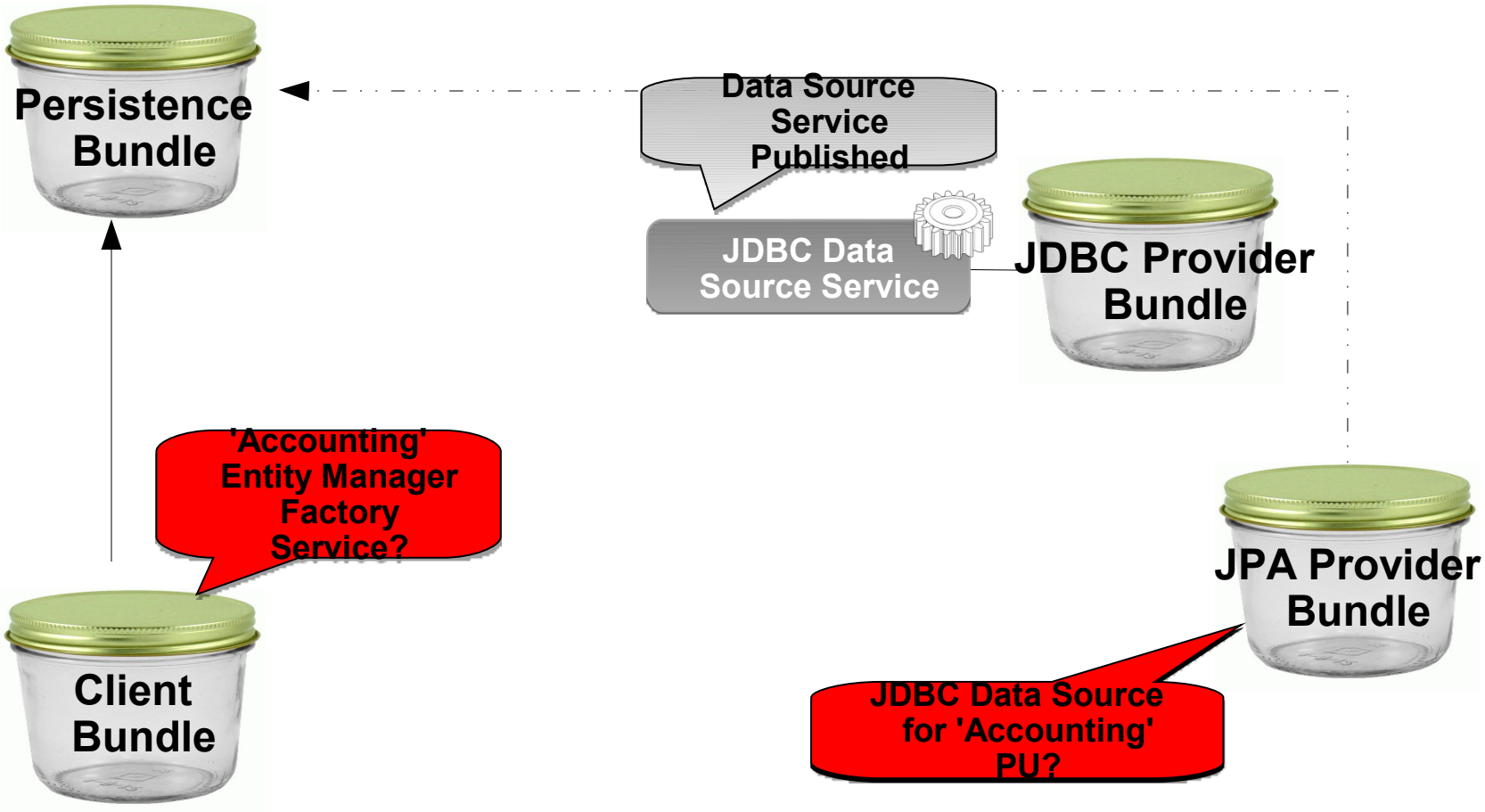
JPA OSGi Example



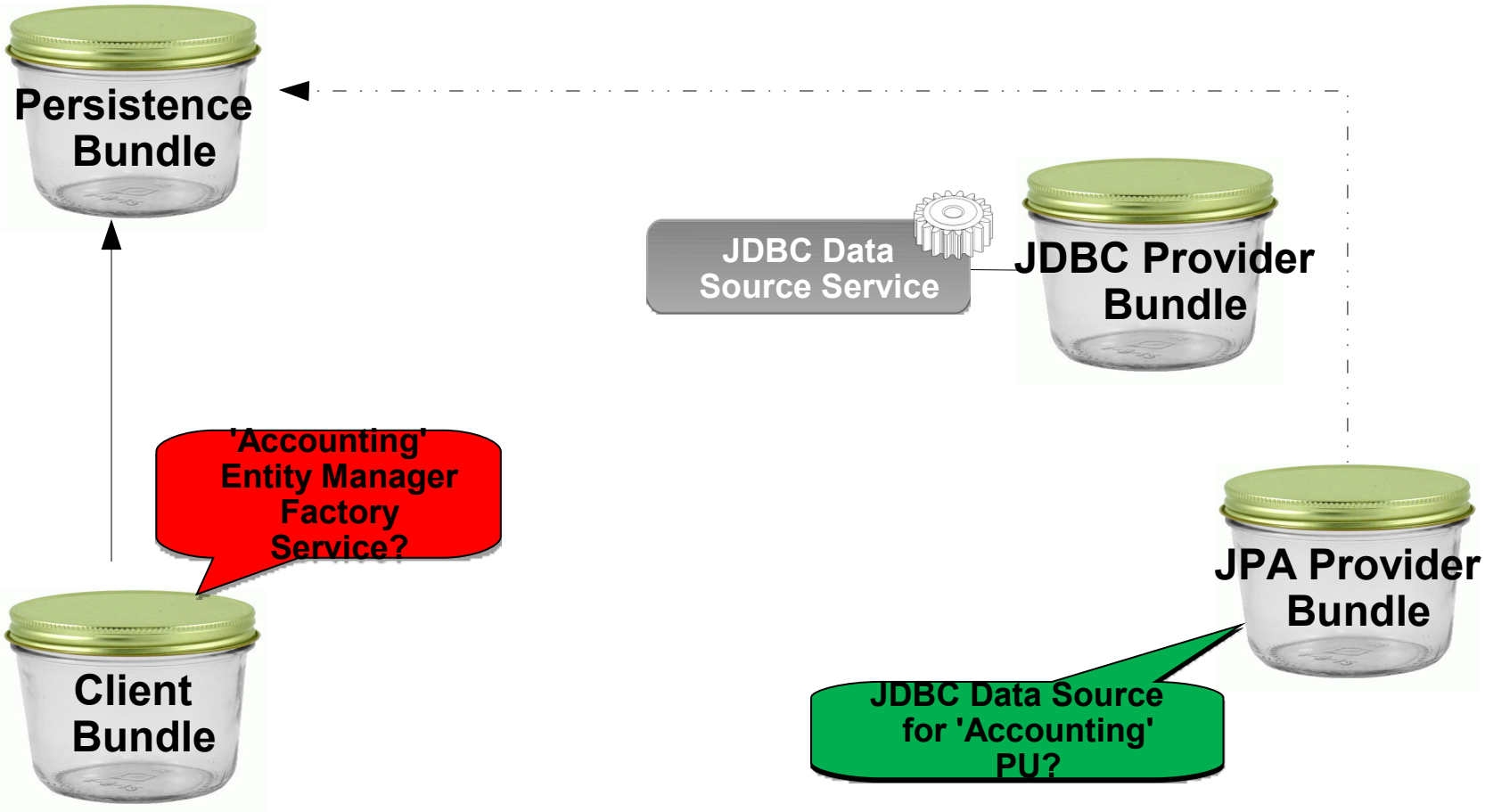
JPA OSGi Example



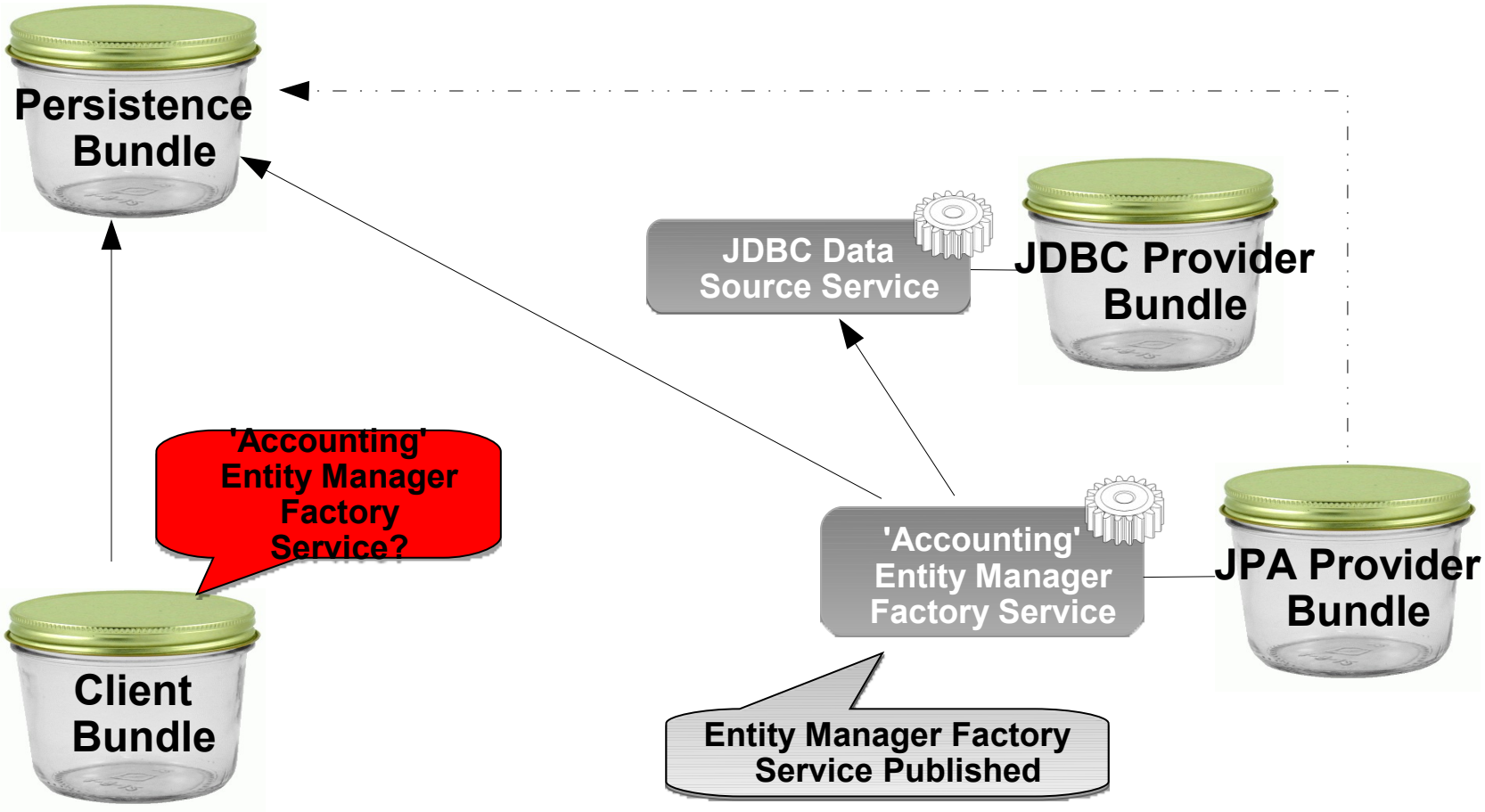
JPA OSGi Example



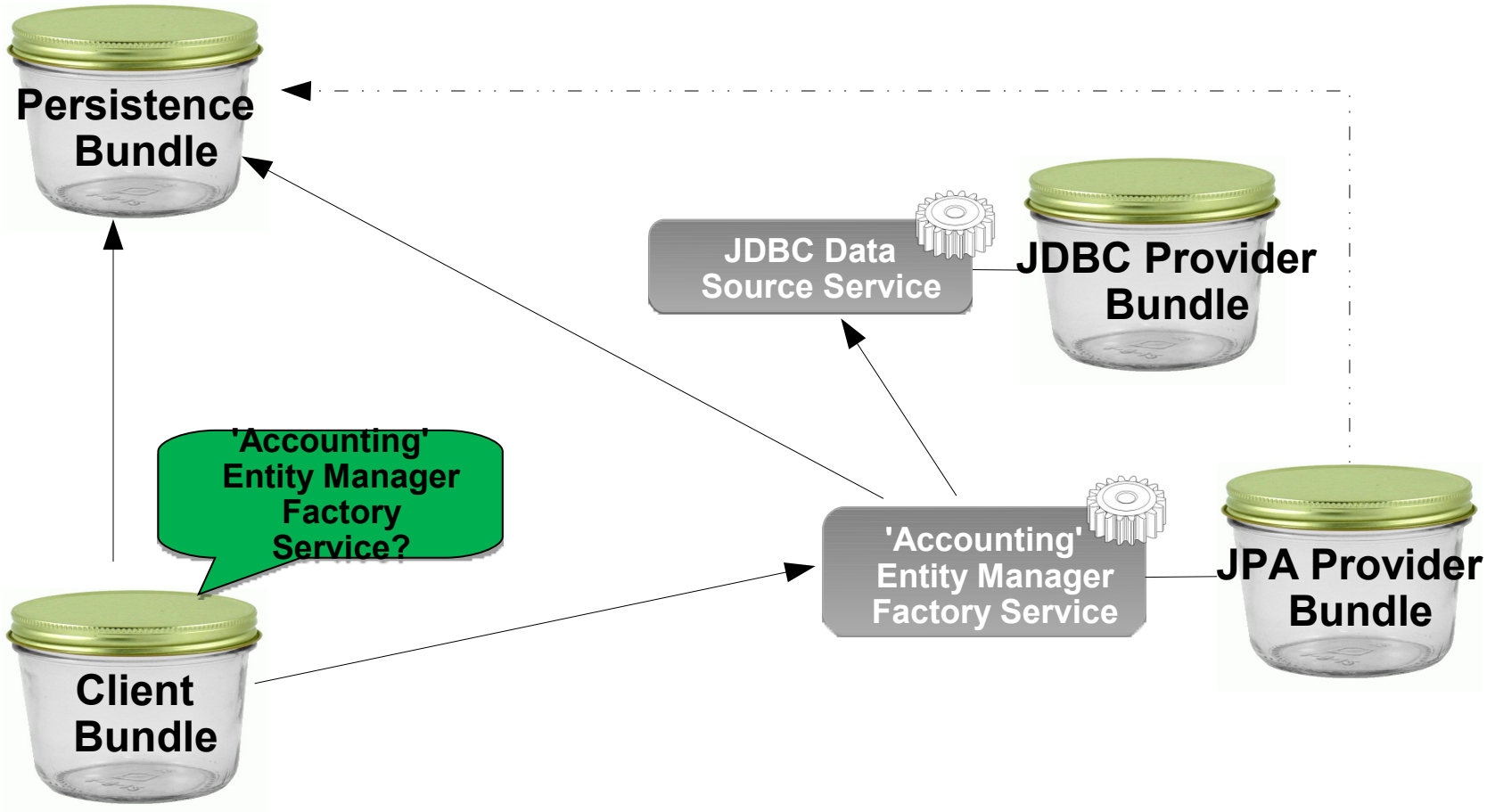
JPA OSGi Example



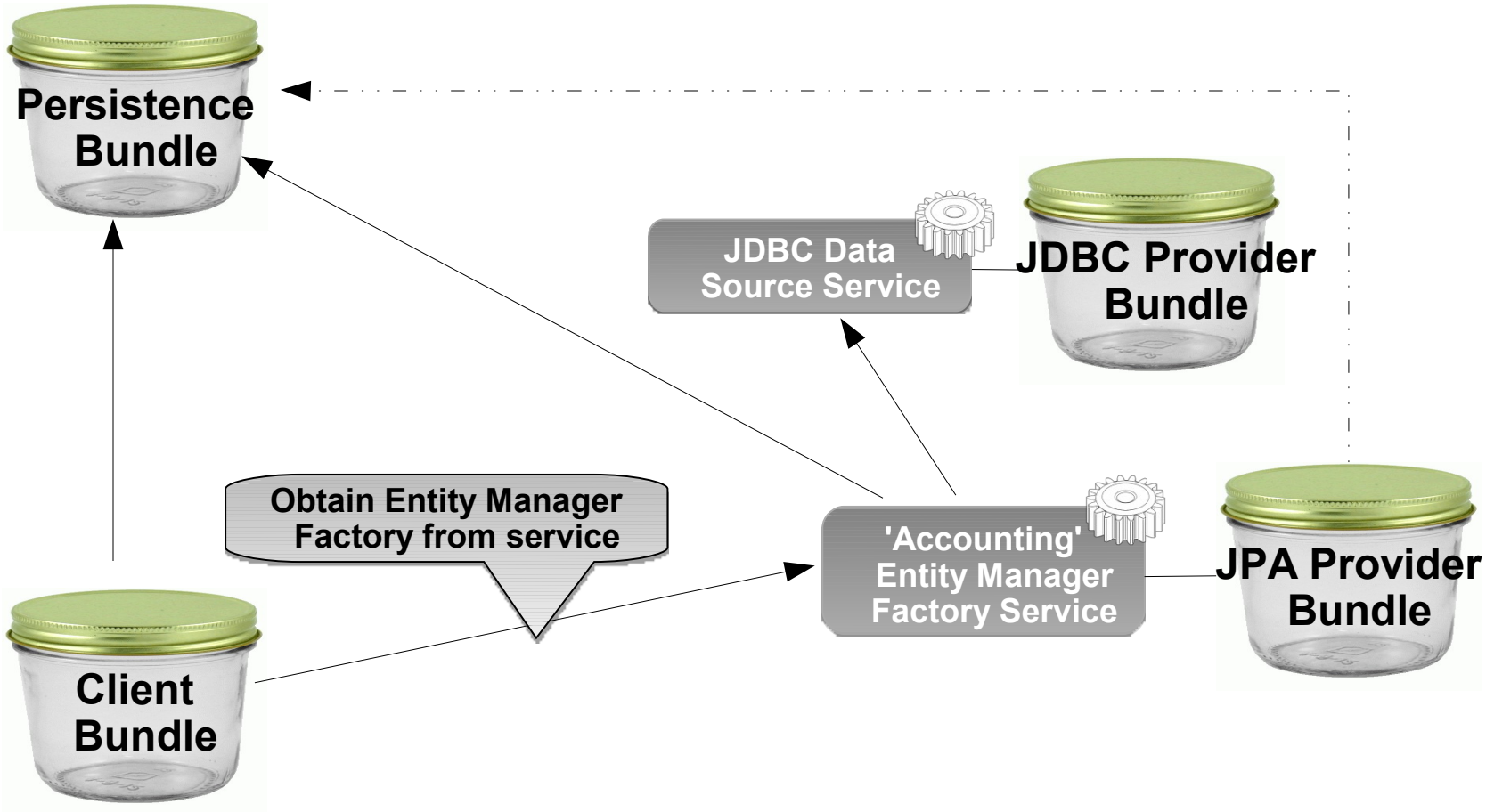
JPA OSGi Example



JPA OSGi Example



JPA OSGi Example



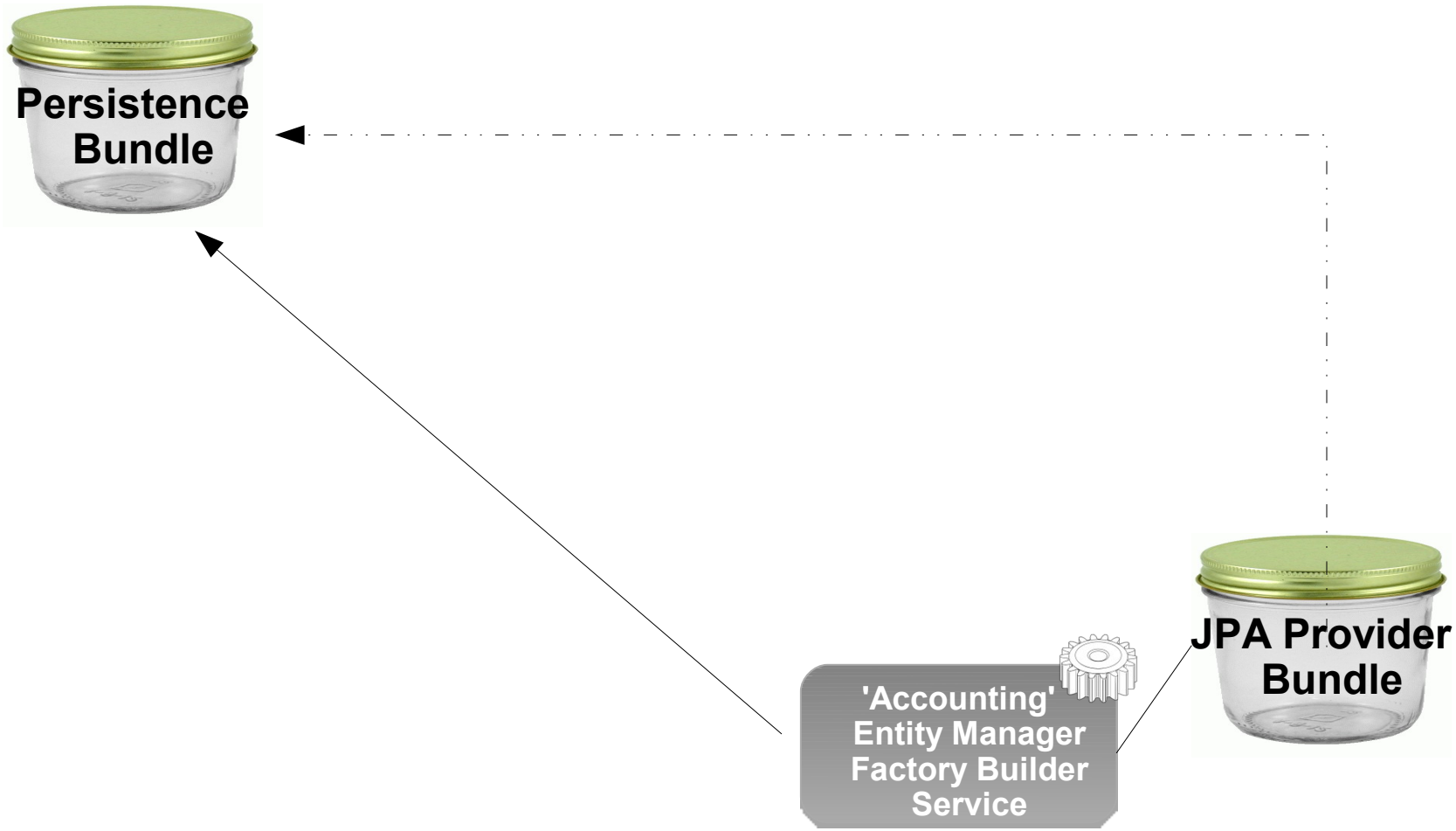
Entity Manager Factory Considerations

- Clients should track Entity Manager Factory service to respond to loss of service
- No need to track Data Source service as this is tracked by Entity Manager Factory service

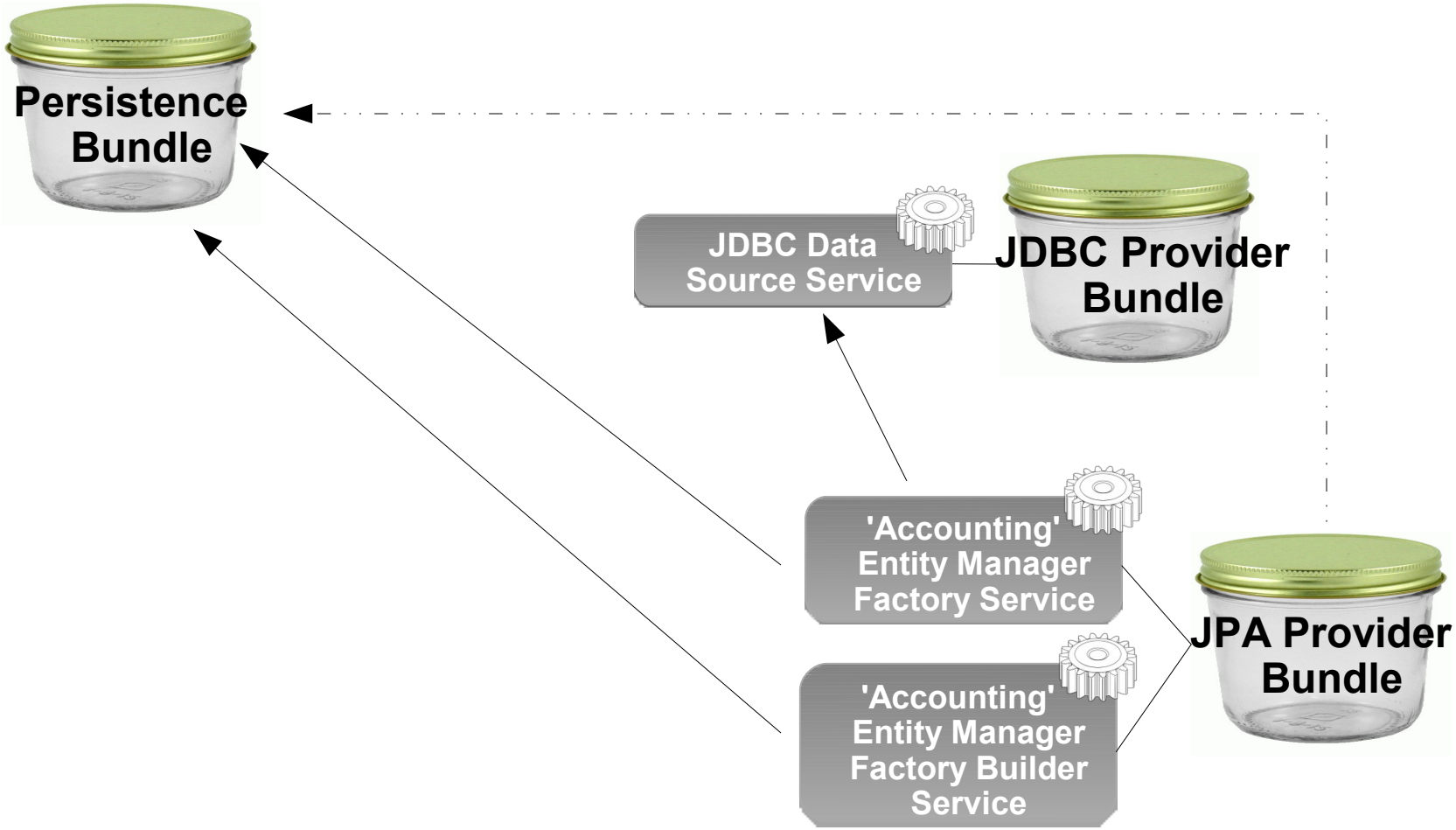
Entity Manager Factory Builder Service

- If a persistence unit does *not* declare a data source:
  - Clients must supply data source connectivity information at Entity Manager Factory creation time.
  - Clients cannot use Entity Manager Factory Service because it will only publish for Persistence Units whose data source service is declared and available.
  - Solution: JPA Providers also publish an Entity Manager Factory Builder service for each Persistence Unit that takes connectivity information
- Builder also for specifying properties at runtime

Entity Manager Factory Builder



Entity Manager Factory Builder



Entity Manager Factory Builder Considerations

- If specified Data Source service is not available createEntityManager(..) call will return null.
- Clients should ensure Data Source service is available before trying to create an Entity Manager
- Clients should track *both* Data Source and Entity Manager Factory Builder services to respond to loss of service

Static Clients

- Previous slides show client using OSGi services to obtain JPA Entity Manager Factory
- Non-OSGi aware clients use static methods on JPA SPI class javax.persistence.Persistence, e.g.:

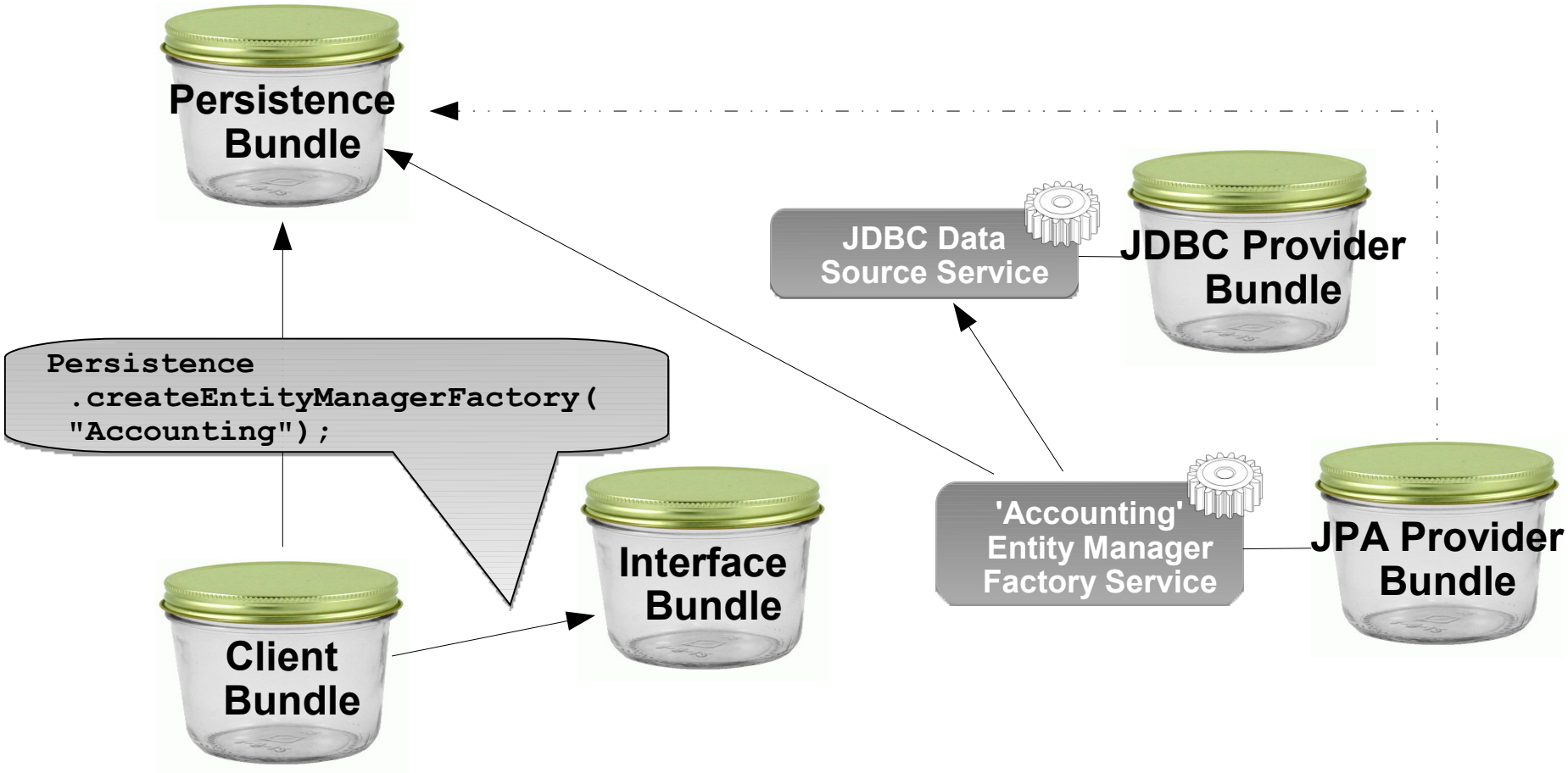
```
EntityManagerFactory emf = Persistence
  .createEntityManager("Accounting");
```

- For compatibility OSGi JPA Service specification provides support for "Static Clients" that use JPA SPI.

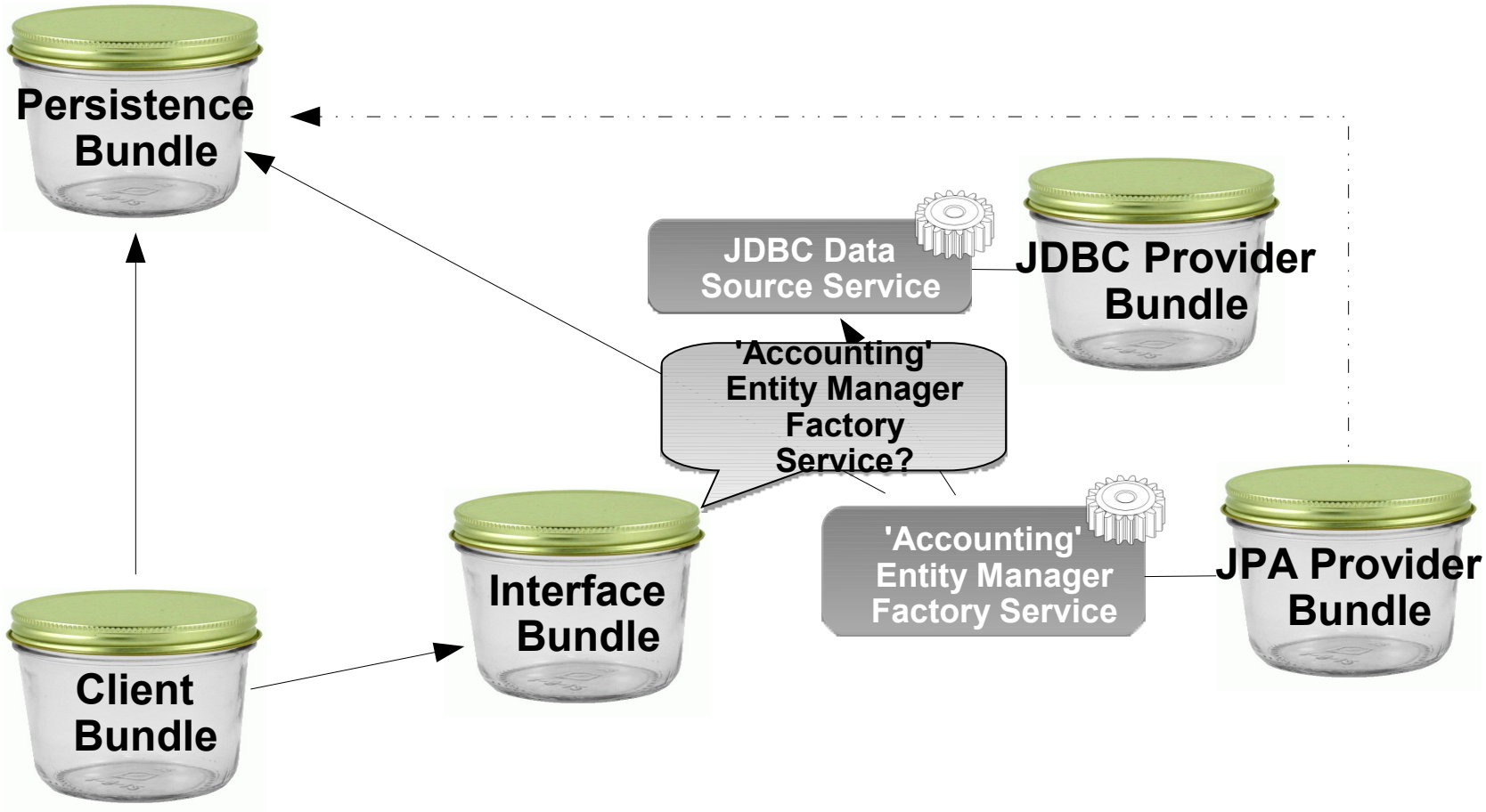
Interface Bundle

- Exports JPA Spec packages javax.persistence.*
  - Packages imported by Persistence Bundles, JPA Provider Bundles, and Client Bundles that use the JPA
- Provides static methods that wrap OSGi JPA services
  - Achieved using OSGi service aware JPA Provider locator strategy

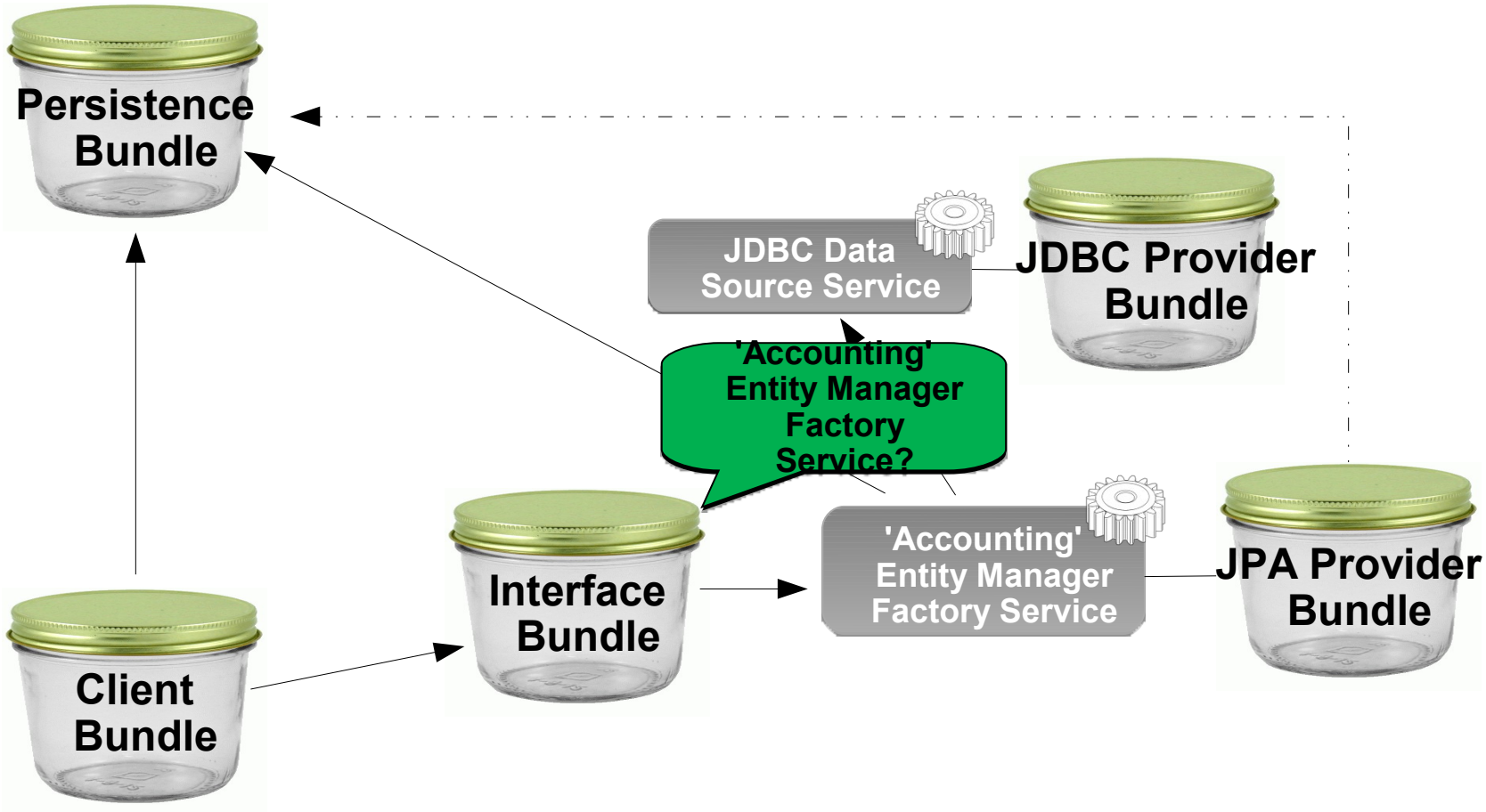
JPA OSGi Static Client Example



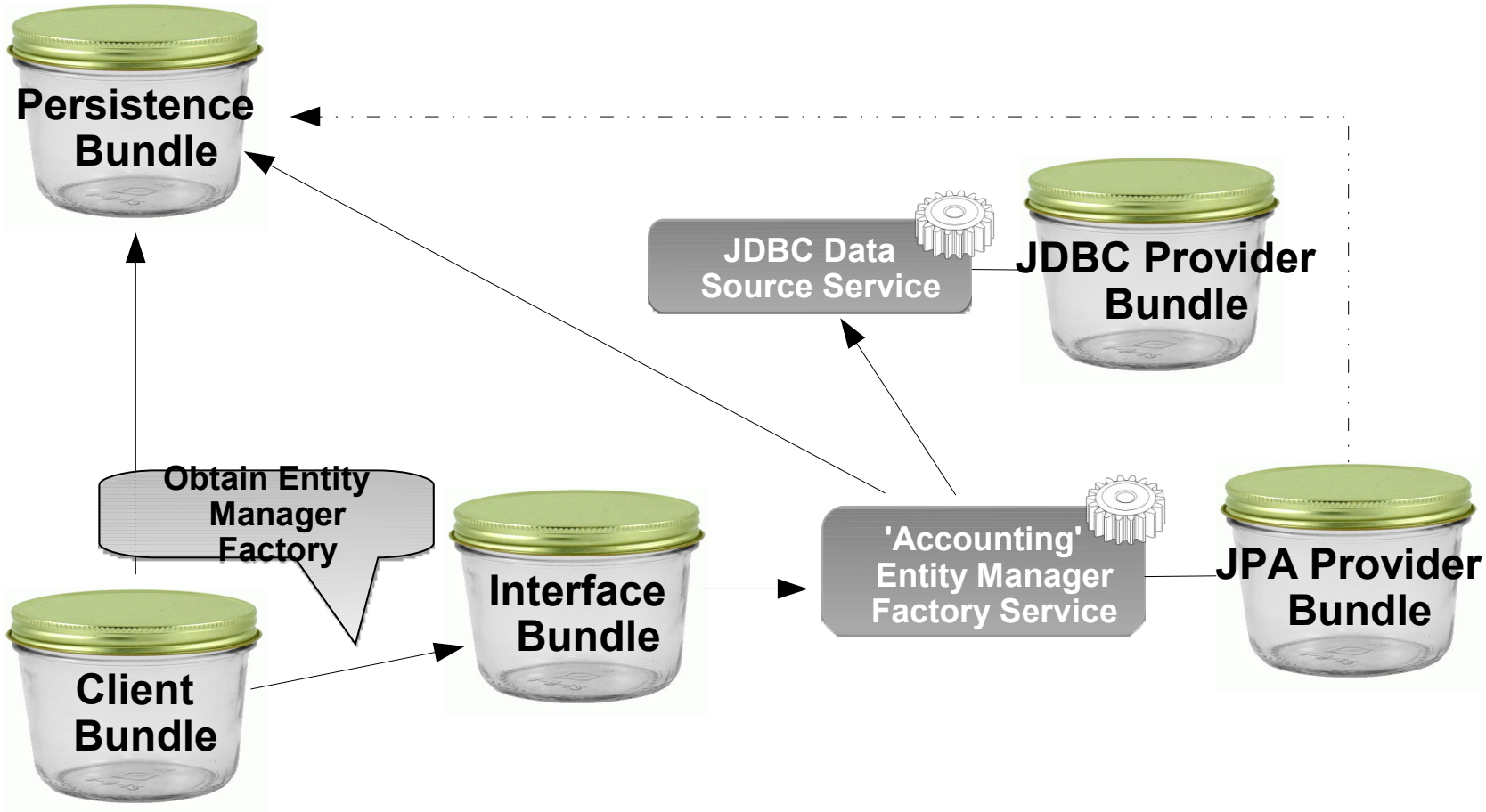
JPA OSGi Static Client Example



JPA OSGi Static Client Example



JPA OSGi Static Client Example



Static Client Limitations

- Fully synchronous so if Entity Manager and Data Source services are not available createEntityManager() call will return null.
  - Dependent on bundle start order for success
- Client not informed when services become unavailable—can lead to unpredictable failures
- Not ideal in dynamic OSGi environment

Versioning

- JPA 1.0 vs. 2.0
  - JPA Provider selected using class consistency
  - Clients importing JPA API classes will be matched with JPA Providers that import the same API classes
- Persistence Unit Versions
  - Specify osgi.unit.version attribute when locating Entity Manager Factory [Builder] service
  - Entity Manager Factory [Builder] service's version is Persistence Bundle's version
- JPA Provider
  - No standard way to specify a provider version, e.g. EclipseLink 1.2 vs. 2.0

Byte Code Weaving

- Byte Code Weaving is used by some JPA Providers to implement lazy loading, change tracking, and other performance features.
  - The OSGi JPA Service specification does not define how weaving may be supported. Implementation details are left to providers.

Summary

- The Java Persistence API is the Java standard for object-relational mapping
- The OSGi JPA Service Specification, a part of the OSGi Enterprise Specification, is the new standard for JPA usage in OSGi
- The Eclipse Gemini project is delivering the OSGi JPA Service specification reference implementation based on EclipseLink, the JPA 2.0 reference implementation
- **A standard and portable way to use JPA is now available for OSGi!**