Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

A Persistence Service for Java

Scarlet Schwiderski

Abstract

A Persistence Service has been implemented as part of the Reflective Java project in the ANSA programme. This presentation gives an overview of the design and implementation of this Persistence Service. The benefits and problems with using Reflective Java are pointed out.

APM.1933.02 **Approved** 24th January 1997

Technical Report

Distribution:
Supersedes:
Superseded by:

A Persistence Service for Java

Scarlet Schwiderski

Key Issue

- maintain a persistent store for Java objects in order to
  - make Java objects reusable by the same or by different applications
  - provide failure resilience to applications

Objectives

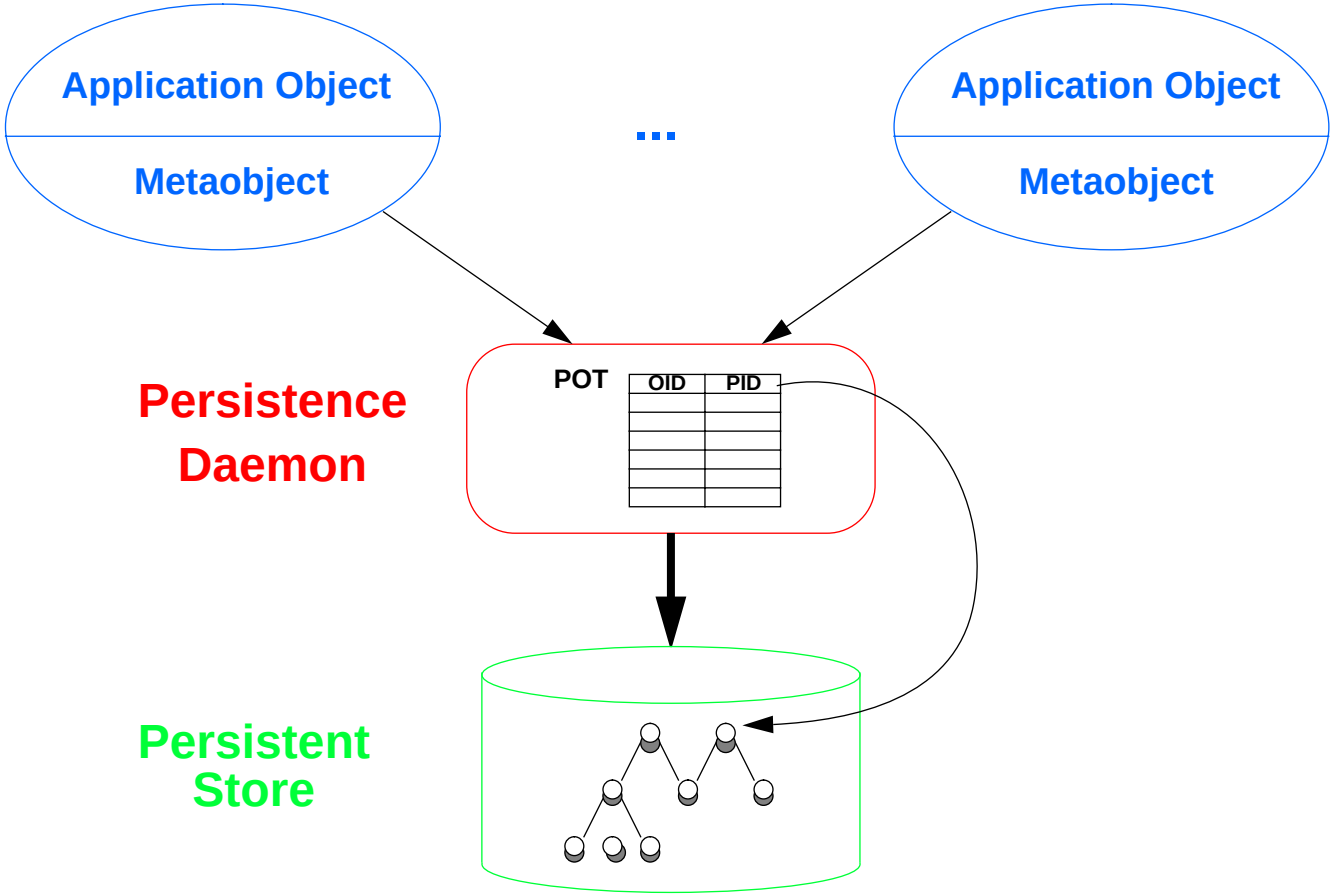
- write Java objects to the persistent store
- read Java objects from the persistent store
- support concurrent access to the persistent store
- provide security measures
- ensure consistency of stored Java objects
- make no changes to the Java language
- exploit existing Java technology

Starting Point

- **Use reflection technology**
  - to evaluate Reflective Java
  - to provide flexibility (for example, using a persistence metaobject in conjunction with a transaction metaobject)

- **Exploit existing Java technology**
  - Object Serialization API (Sun)
  - Cryptography API (Systemics)

The Persistence Model



Persistence Metaobject

- contacts the local *Persistence Daemon* in order to
  - activate a corresponding Java object
  - deactivate a corresponding Java object

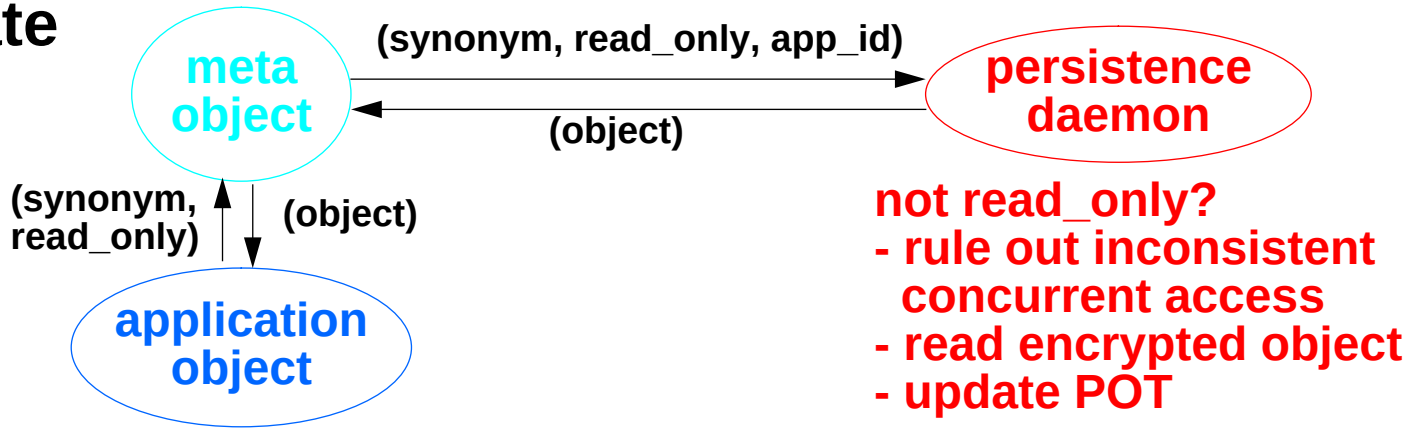
- may support other services, such as a transaction service

Persistence Daemon

- well-known central server at a site
- coordinates access to the persistent store
  - allocates disk space for objects in the persistent store
  - locates objects in the persistent store
  - manages multiple applications concurrently
  - encrypts/decrypts objects
  - writes shadow copies
- maintains the *Persistent Object Table (POT)*
  - in order to keep track of activated and deactivated Java objects
- provides garbage collection

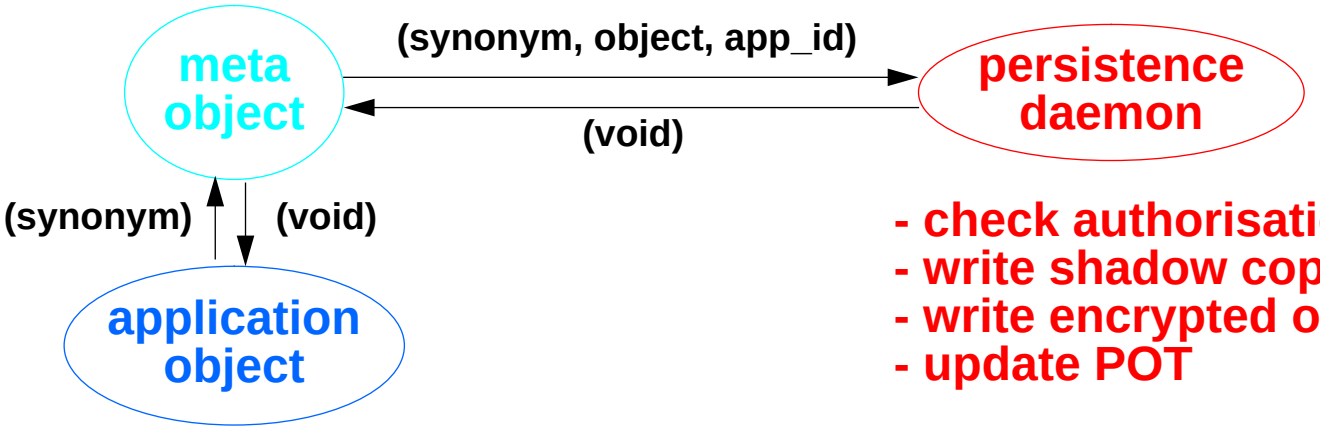
The Activation and Deactivation Process

Activate



not read_only?
- rule out inconsistent concurrent access
- read encrypted object
- update POT

Deactivate



- check authorisation
- write shadow copy
- write encrypted object
- update POT

Summary

- Use Reflective Java to implement a persistent store for Java objects
- Enhance existing Java technology, i.e. the Object Serialization API
  - concurrency
  - security
  - consistency
- **Benefits:**
  - flexibility
  - little overhead to end users
  - generic for all applications requiring persistence
- **Problems:**
  - limited Object Serialization API
  - Reflective Java supports behavioural and not structural reflection
  - not completely transparent to end user