

ORACLE®



ORACLE®

POF Art

Harvey Raja

Oracle Coherence | Principal Member Technical Staff

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Portable Object Format

- Description
- Who is not using POF?
- Features / Time:
 - Annotations
 - Identity and Reference Support
 - Nested Writers
 - POFExtractor / POFUpdater
 - Evolvable

Deconstruct the Binary

DECO	TypeId (custom)	Attribute Index	TypeId 0x41	23	Attribute Index	TypeId 0x79	Attribute Index	TypeId 0x4E	foo	TypeId (custom)	...
------	--------------------	--------------------	----------------	----	--------------------	----------------	--------------------	----------------	-----	--------------------	-----

- Decorations used for internal purposes
- Well Defined structure allows traversal (POFExtractor)
- Packed Integers
 - Integer != 4 bytes
 - Sign and continuation bits
- Tiny Strings
- POF Type Id minimum is 1 byte (63 types)
 - int 16 == 0x79 type-id

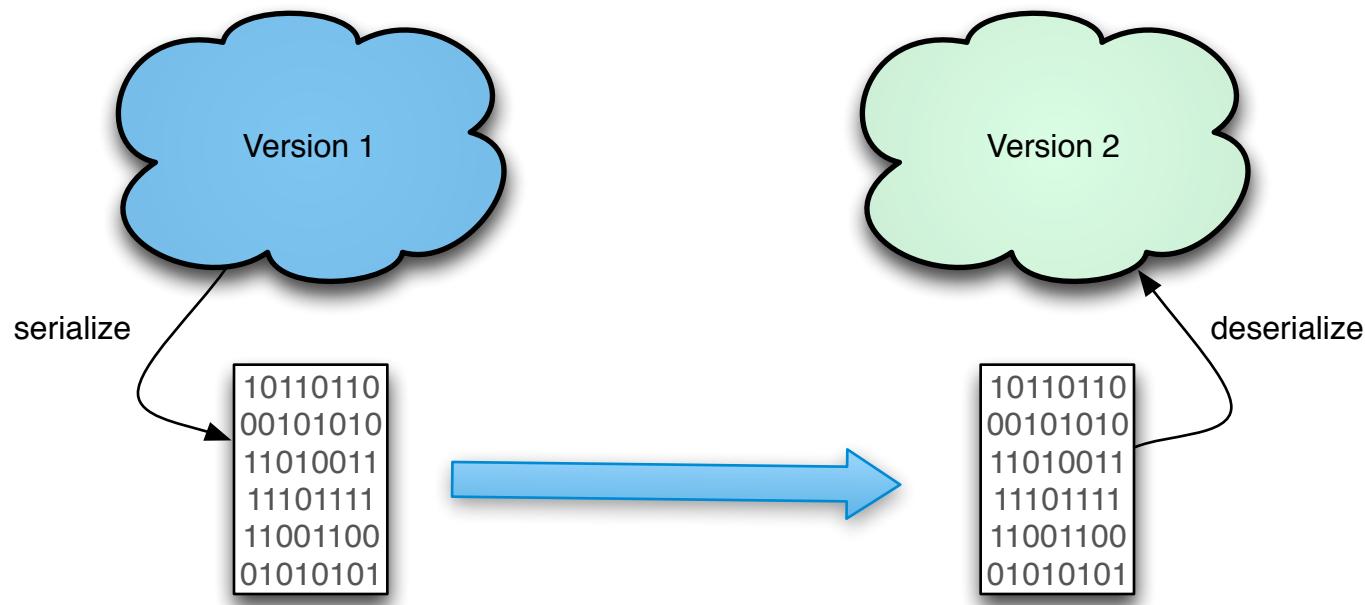
Tips

- String use smallest character set possible
- Nested buffers avoid attribute index collision
- Take advantage:
 - Use integers in the range of -1 to 22

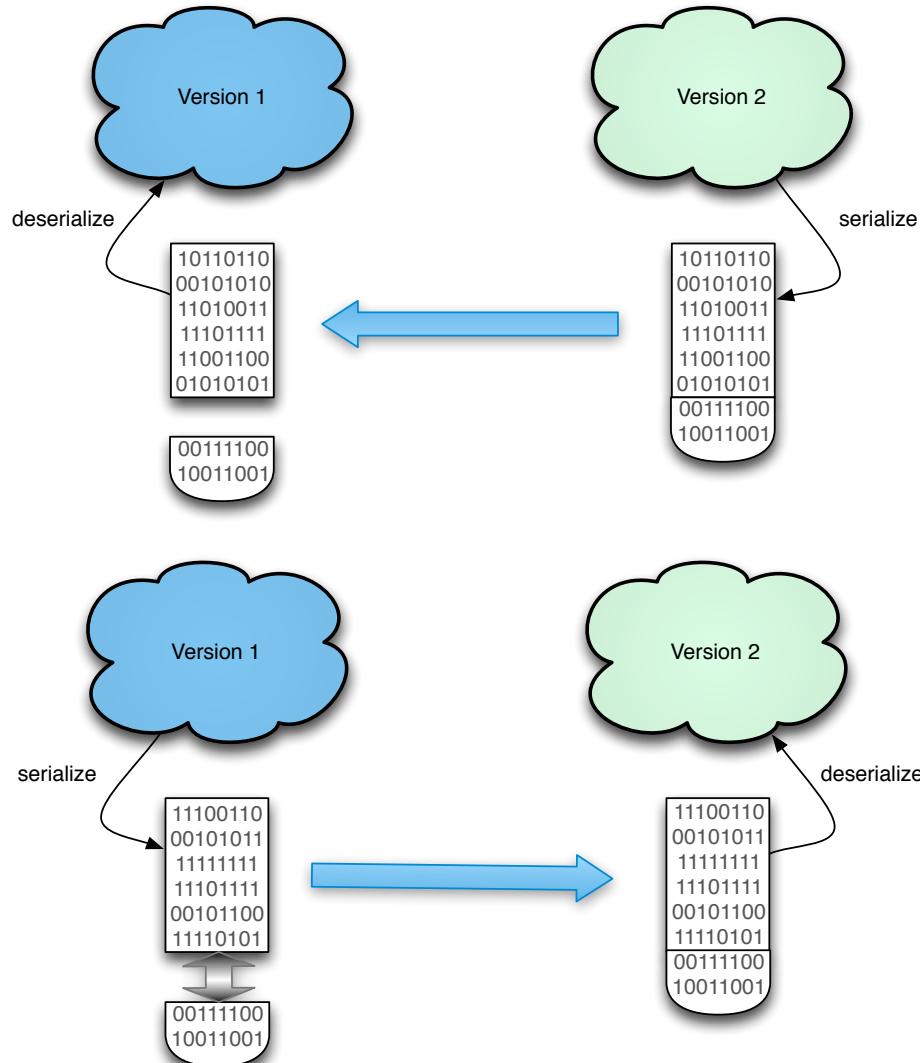
Evolvable

- The ability to maintain heterogeneously versioned objects in a Cluster
 - Lack of control of clients
 - Time window of a rolling restart
 - Just because you can
- Maintain existing structure
- Append new attributes

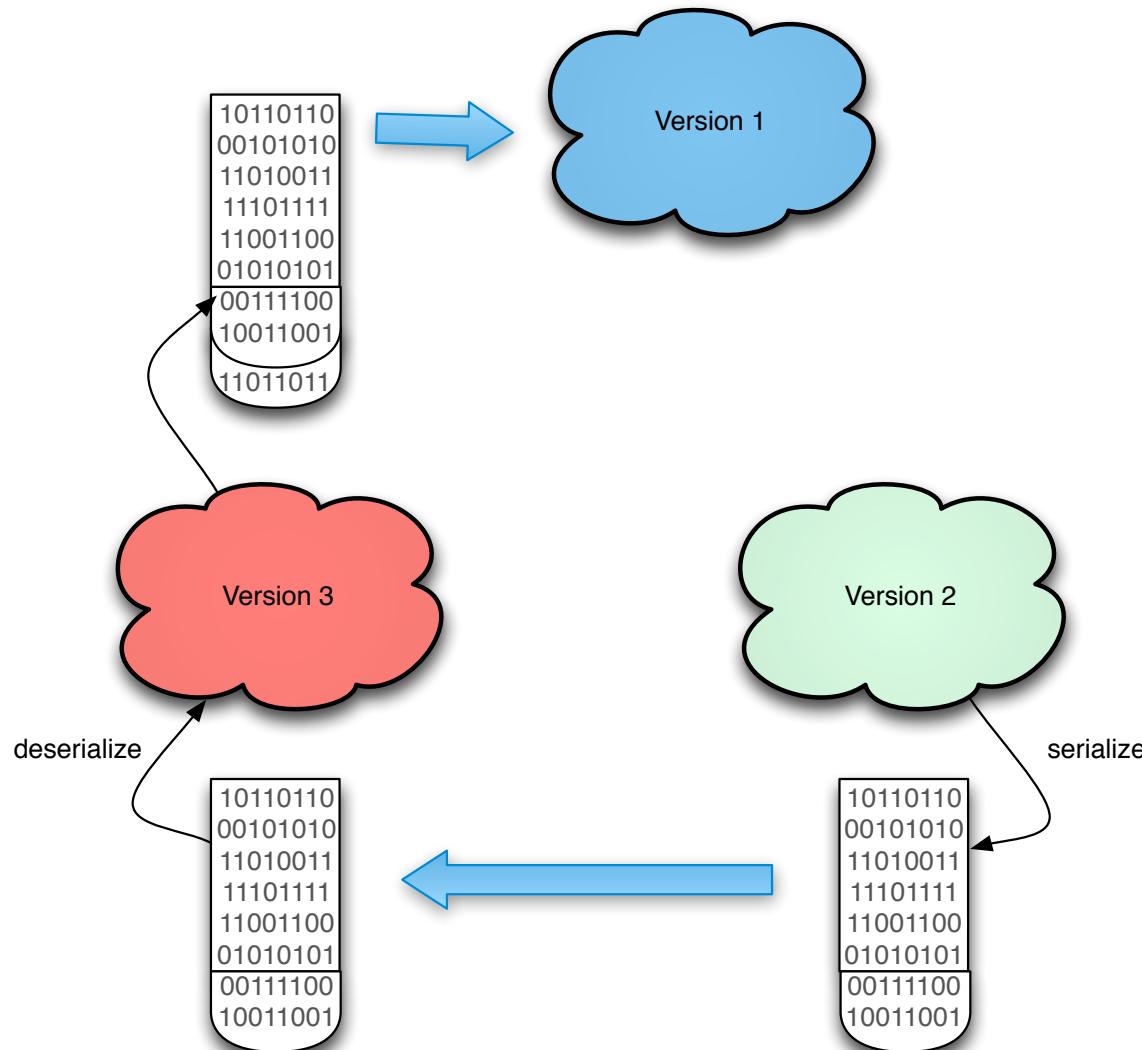
Still Evolving



Evolve already!



Tri-Evolvable



Delta Compressor

- Can be used to create binary diff between two binary objects
- Can be enabled for backups:

```
<distributed-scheme>
  <compressor>standard</compressor>
</distributed-scheme>
```

- May be useful in your applications

BinaryDeltaCompressor#applyDelta

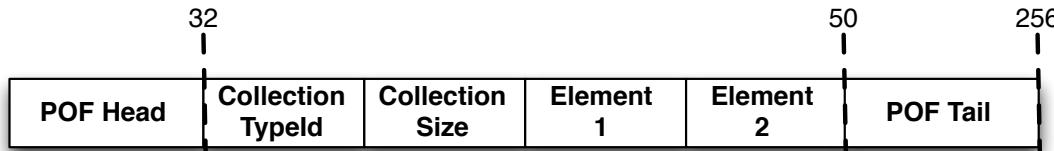
- Receives the original binary and the delta binary
- The delta is a sequence of instructions for applyDelta to execute
- First byte suggests mode of operation:
 - Empty Binary
 - Return an empty byte[]
 - Binary Replace
 - Ignore the old value in favor of the delta in its entirety
 - Binary Diff
 - This delta contains instructions allowing a merge using the old value and the delta

Binary Diff

- Delta is a stack of instructions and operands with the old binary acting as a data register
- Reverse of Java Byte code
 - Instruction followed by operands
- Mnemonics:
 - OP_EXTRACT 0x01 offset, number of bytes
 - Extract from old binary
 - OP_APPEND 0x02 number of bytes
 - Append from delta
 - OP_TERM 0x03
 - Terminate

Collection Manipulation

- Ever wanted to modify a collection without deserializing all elements?
- Binary Old Value:



- Instruction Set & Operands (Binary Delta):

FMT_BINDIFF	OP_EXTRACT	0	32	OP_APPEND	34	Collection Typeld	New Size	Element 1	Element 2	Element 3	Element 4	OP_EXTRACT	50	256
-------------	------------	---	----	-----------	----	-------------------	----------	-----------	-----------	-----------	-----------	------------	----	-----



Demo

ORACLE®



ORACLE IS THE **INFORMATION COMPANY**