CONSORTIUM

JEFF™

ISO 20970

Jean-Paul Billon
SchlumbergerSema
chairman@stip.org

What is JEFF?

- JEFF is a storage format for set of classes
  - "alternative" to JAR
- JEFF solves completely one of the most important obstacles to the deployment of Java on small footprint devices
- JEFF Results of 4 years of research and experimentation from the participants in removing this obstacle (maturity)
- JEFF can be used also with important benefits on "bigger" platforms

JEFF History

- 01/2000: Workgroup created to define a ready-for-execution format.
- End 2000: Public review of JEFF specification release 1.0
- Feb 2001: Final release of JEFF 1.0 specification
  - JEFF™ is a trademark of the J Consortium
- Apr 2001: Submitted to ISO "Fast Track" procedure
- September 2001: JEFF becomes ISO/IEC DIS 20970
- Nov 2001: 19 countries vote, 14 approve, 2 disapprove votes
- January 2002: Ballot Resolution Meeting
  - Comments are resolved

Usual Sun's Class File Format

- Two parts:
  - constant pool: viewed as a TLV linear expression of a table with variable length elements
  - bytecode: references = indexes (not offsets!) in constant pool viewed as a table:
    - ⇒ 2 bytes indexes allow for 65536 entries in constant pool
    - ⇒ a class file can be > 64KB (e.g. class java.lang.Character in J2SE)

- Requires a reformatting for efficient execution
  - ⇒ Requires a recopy into runtime memory for transformation
  - ⇒ Runtime RAM = heap + stack + Reformatted Classes > size stored program!!
  - ⇒ Slowness of program start!

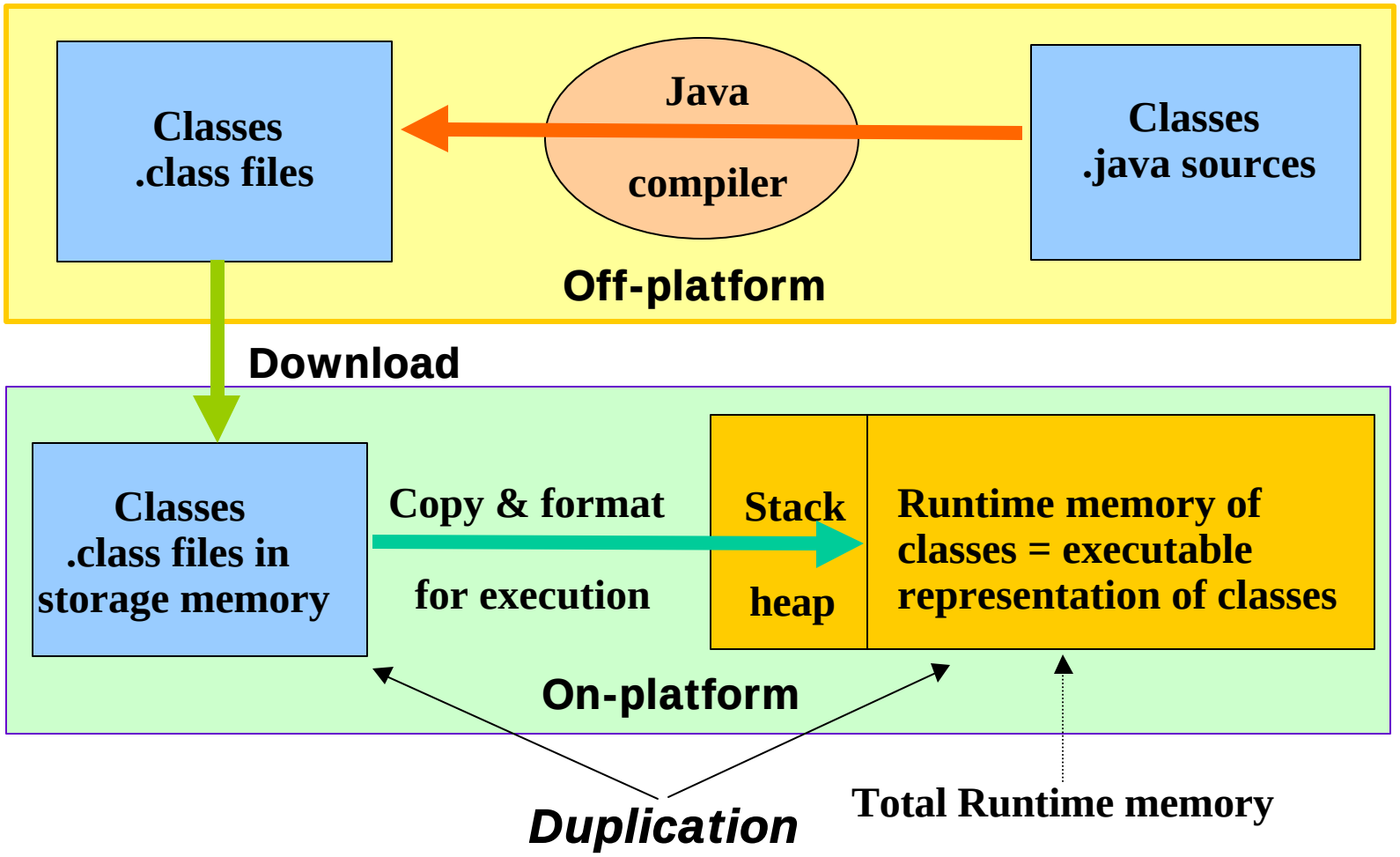
Consequence

- On small devices:
  - To provide small VMs does not solve the real problem of the waste of runtime memory when running non-gadget programs.
  - On very small devices (<128 KB for runtime memory) no “serious” use of Java possible, even with KVM.
- On “bigger”platforms:
  - The need in runtime memory often goes beyond reason (e.g. Forte or Jbuilder4 need a minimum of 256 MB to run efficiently!).
- Real-Time: no “instantaneous” start

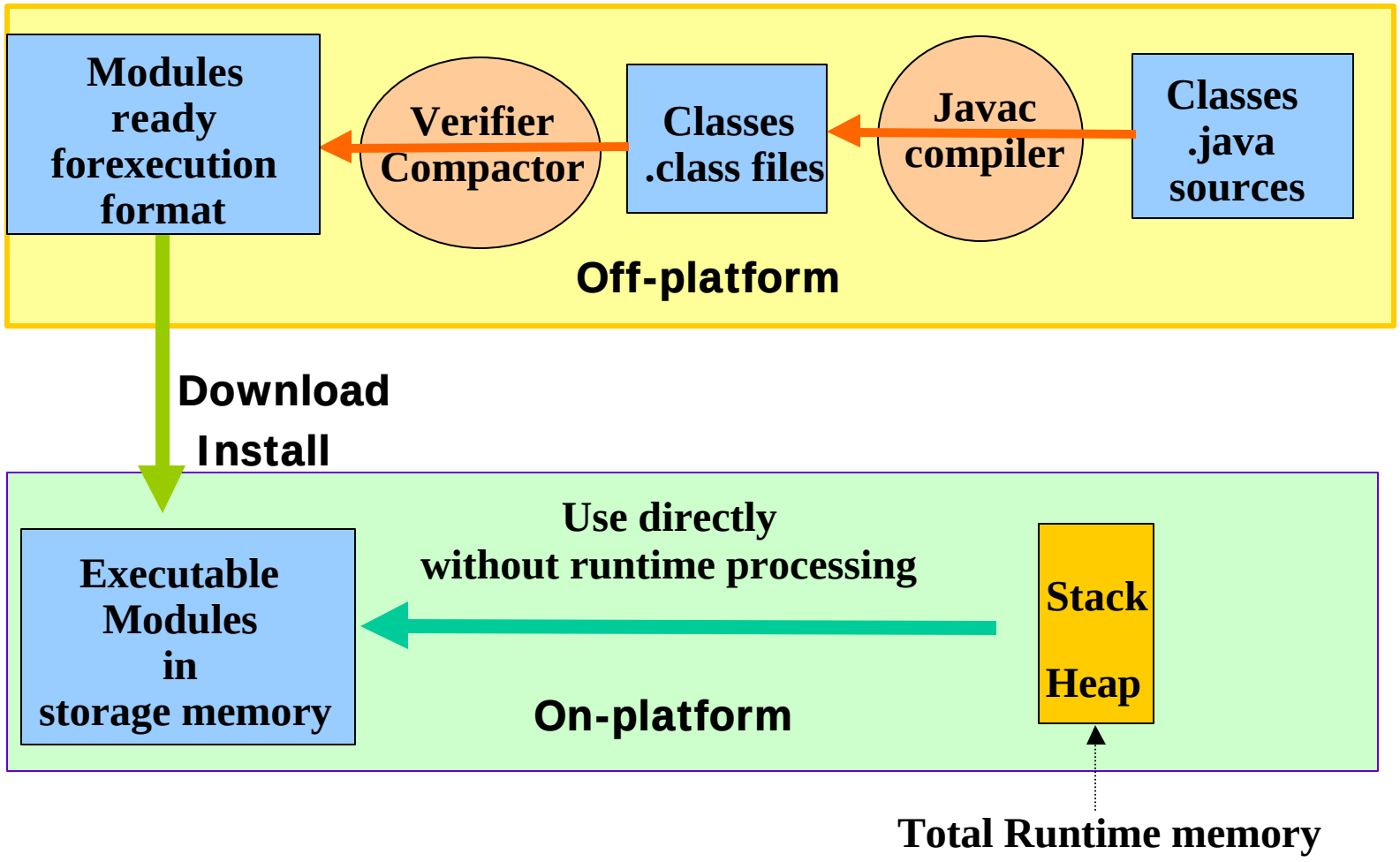
Split-VM Solution

- Uses a separate *converter* translating Class File Format into a Ready-For-Execution format
- Uses a special VM executing in place the ready-for-execution format without recopy into runtime memory
- On small devices the converter can be off-Platform
- VM + Converter = “classical” VM split in two components

Usual Duplication of Classes between Storage and Runtime Memory



Split-VM Approach



Previous Formats for SPLIT-VM

- "Romized" Classes
  - image of the runtime memory of classes in one file
  - completely linked
  - => flexibility of dynamic linking **completely lost**

- Java Card CAP File Format (Java Card)
  - modules internally pre-linked
  - use of 16-bit offsets instead of indexes
    - CAP file limited to 64 K
  - external references by numeric "tokens" instead of symbolic names
    - **compact**
    - flexibility of dynamic linking **partially kept**
    - *but requires rigorous management of token allocation by central authority*

Objective in Designing JEFF

"Dream" format for split-VM that could be both:

- Compact like Cap File format
- Not pre-linked at all
- Preserving all original .class information, even symbolic
- Support of all Java features
- Really ready for execution (could be put in ROM)
- Really efficient (more than usual runtime representation of .class classes)
- Able to store any additional resources (e.g. files) besides classes
- No size limitation

Choices

- A file can contain several classes from several packages. The content can be a complete application, parts of it, or only one class
- To allow the “dynamic linking” of the classes, the references between classes must be kept at the symbolic level
- The binary content of the file is adapted to be efficiently read by most of the processors (byte order, alignment…)
- JEFF is also highly efficient for the dynamic download of classes in dynamic memory (RAM)

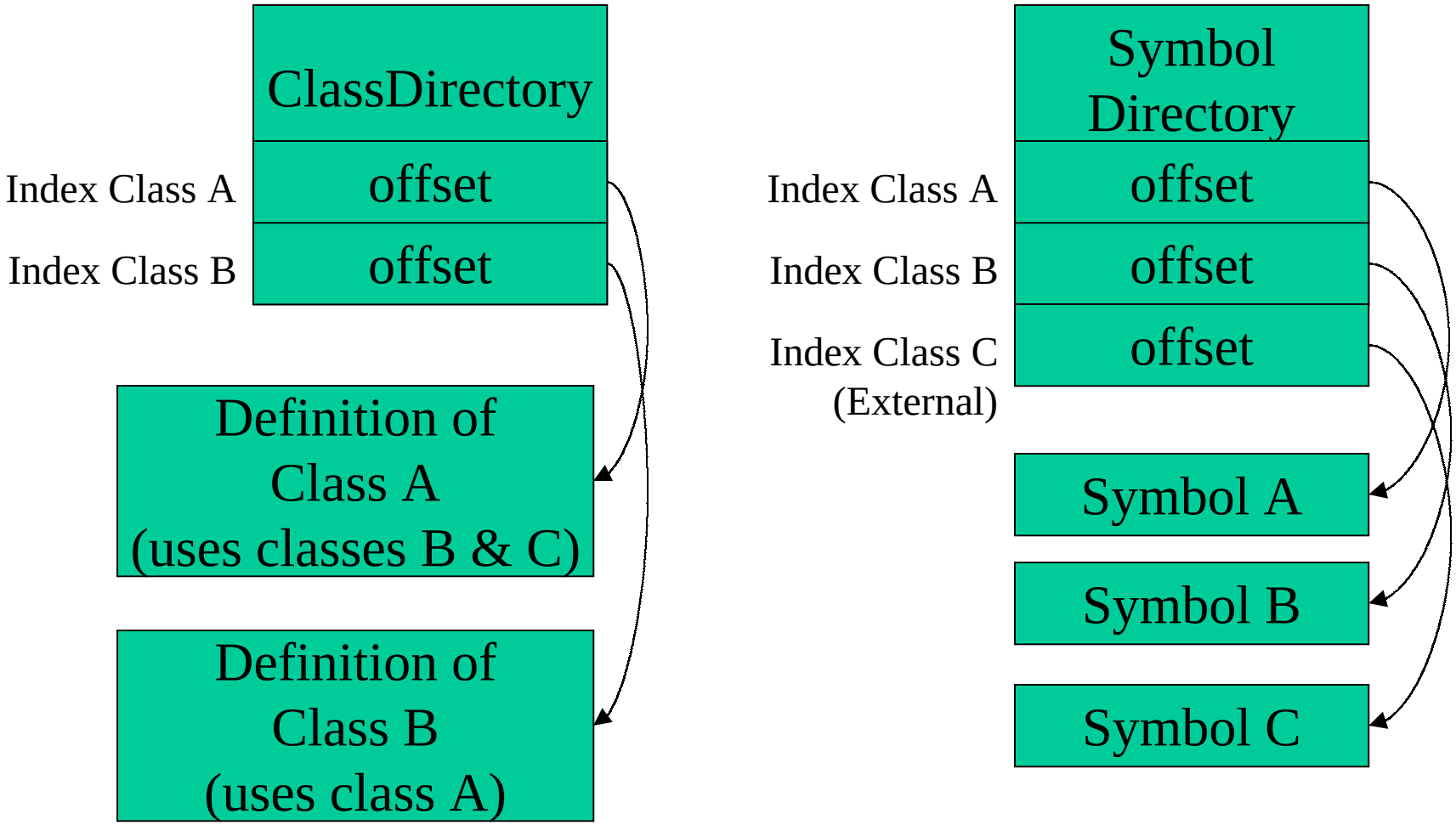
JEFF File Structure

- JEFF file = set of individual classes
  - not pre-linked
- Each class referenced is assigned an index
  - 16 bits indexes: the number of referenced classes < 64K
  - lower indexes for internal classes
  - Indexes local to the JEFF file
- The JEFF file has a common table of the internal classes with a 32 bit offset to reach the individual class headers
- The symbolic names are stored in a common symbol pool

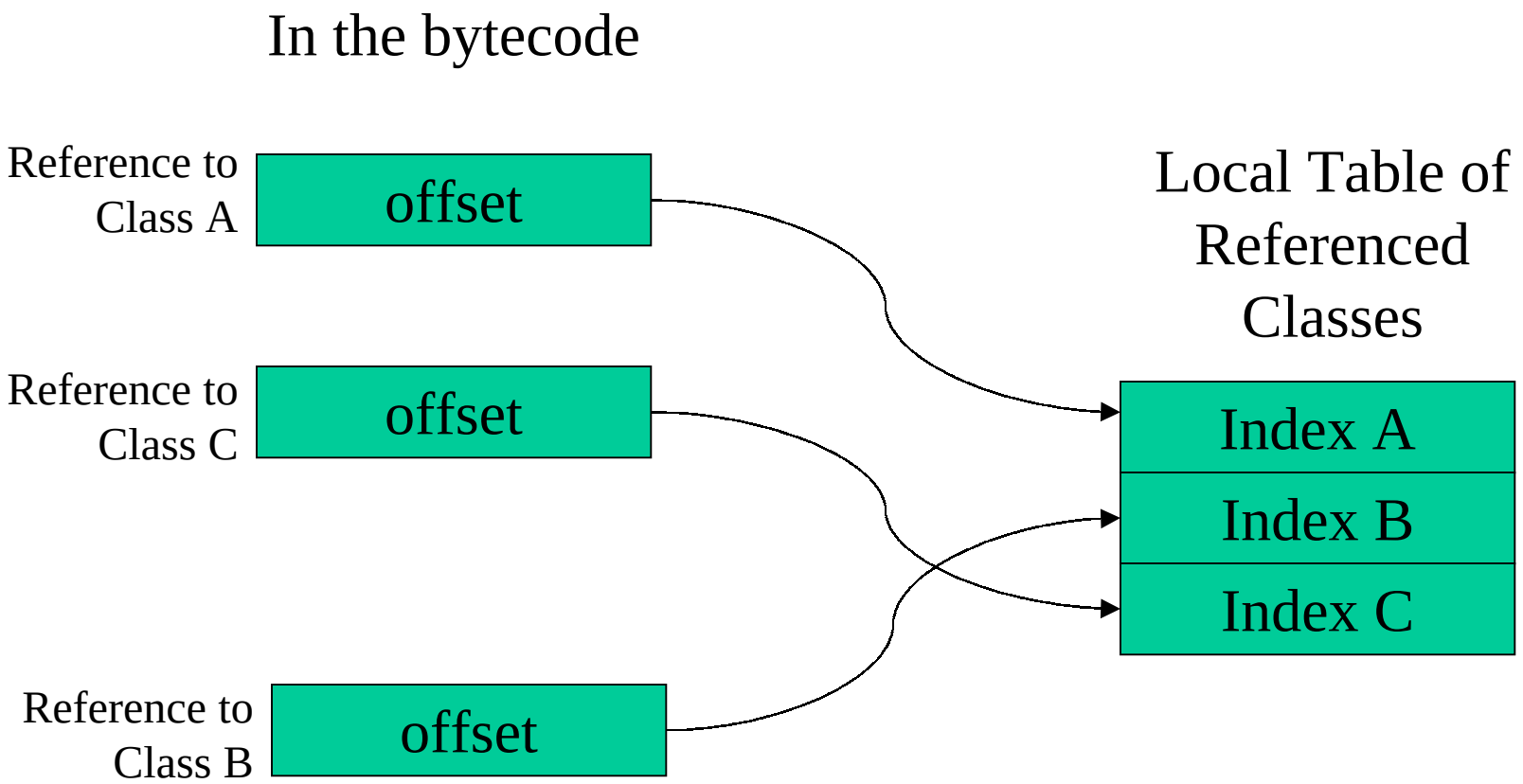
Individual Class

- Internal references inside a class made by 16 bit offsets
  - a class < 64 KB (only limitation)
- Each class has a local table of the classes it references that contains the indexes of the referenced classes
- The references external to a class are made via a 16 bit offset to this local table

Content of a JEFF file



In the Definition of class A



JEFF “Linking”

- Simple and efficient
- On-platform
- Do not modify the content of the files
- Keeps the symbolic information
- Allows the linking of additional JEFF files

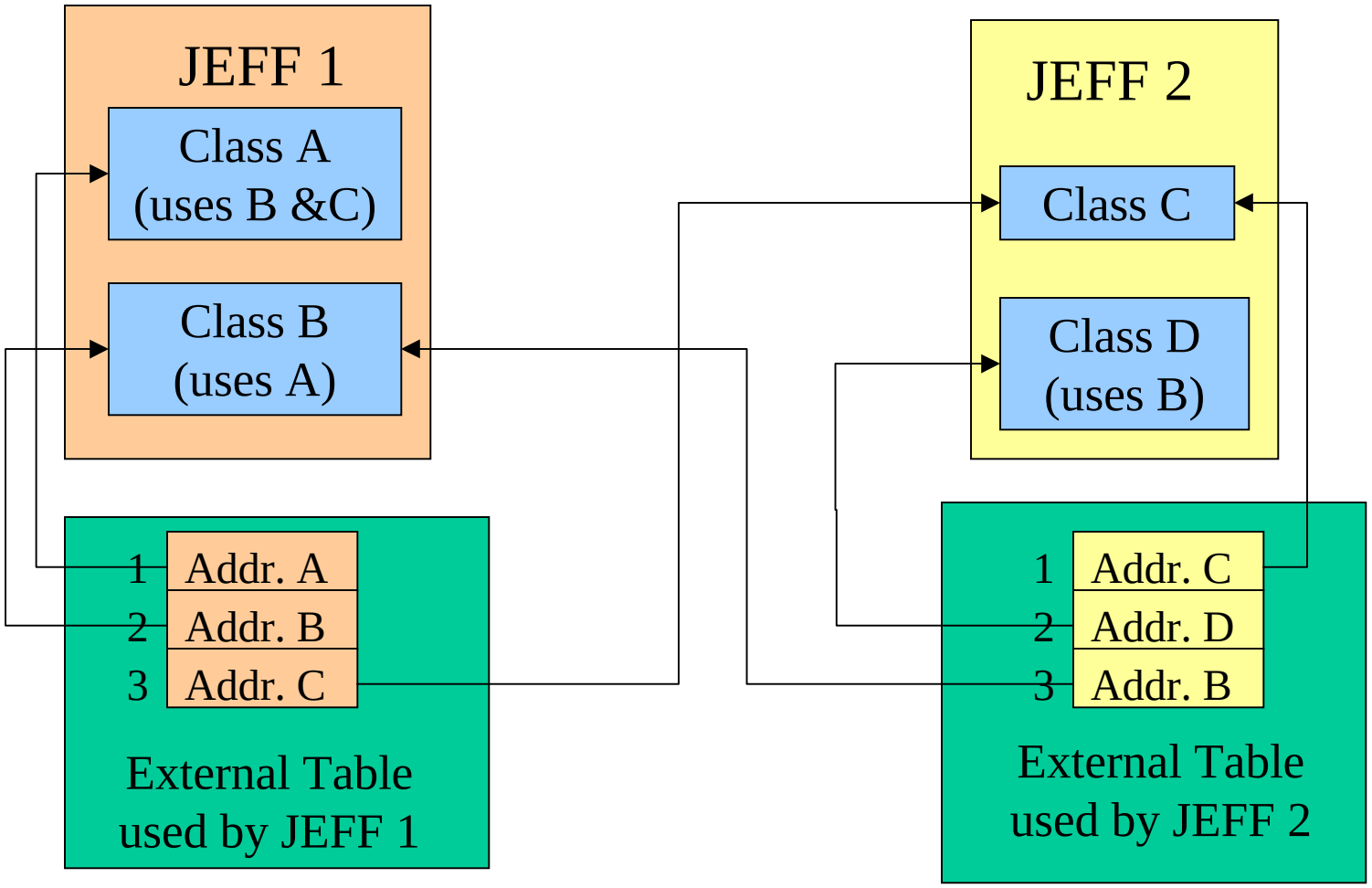
Linking:Use of a single JEFF

- All the classes stored in a single JEFF file
- The class indexes identify unambiguously the class locations
- No linking needed!

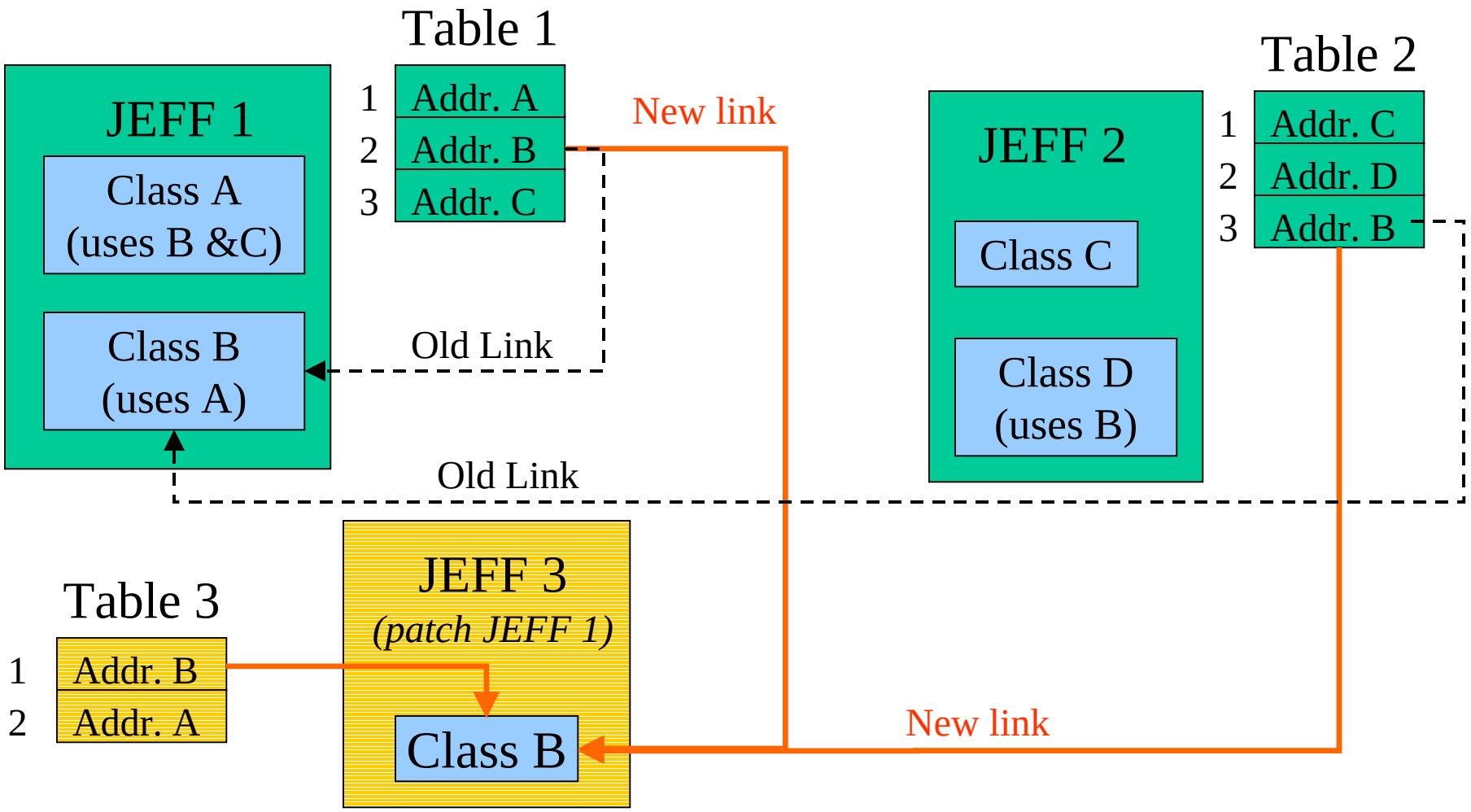
Linking: Multiple JEFF Files (1)

- JEFF indexes cannot be used directly
- An external translation table must be built for each JEFF file:
  - An entry per local class index (internal and external)
  - Each entry in the table contains the global index of the class (in the same JEFF or in another JEFF)

Multiple JEFF Files (2)



Patch of Classes

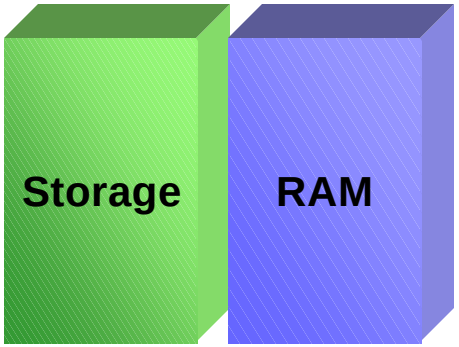


CONSORTIUM

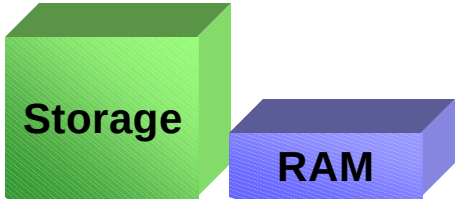
Advantages

- **Compact execution file format**
  - •Without compression, code size is reduced 40-50% compared to Sun's class file.
  - •Well designed for Java processor (Memory alignment, organization,...)
- **Execution from storage memory**
  - •Saves unnecessary copying to RAM
  - •Cost savings for devices – less RAM
  - •Power consumption saving-The static memory consumes less power than the dynamic memory

=> More efficient execution



Regular Class File Format

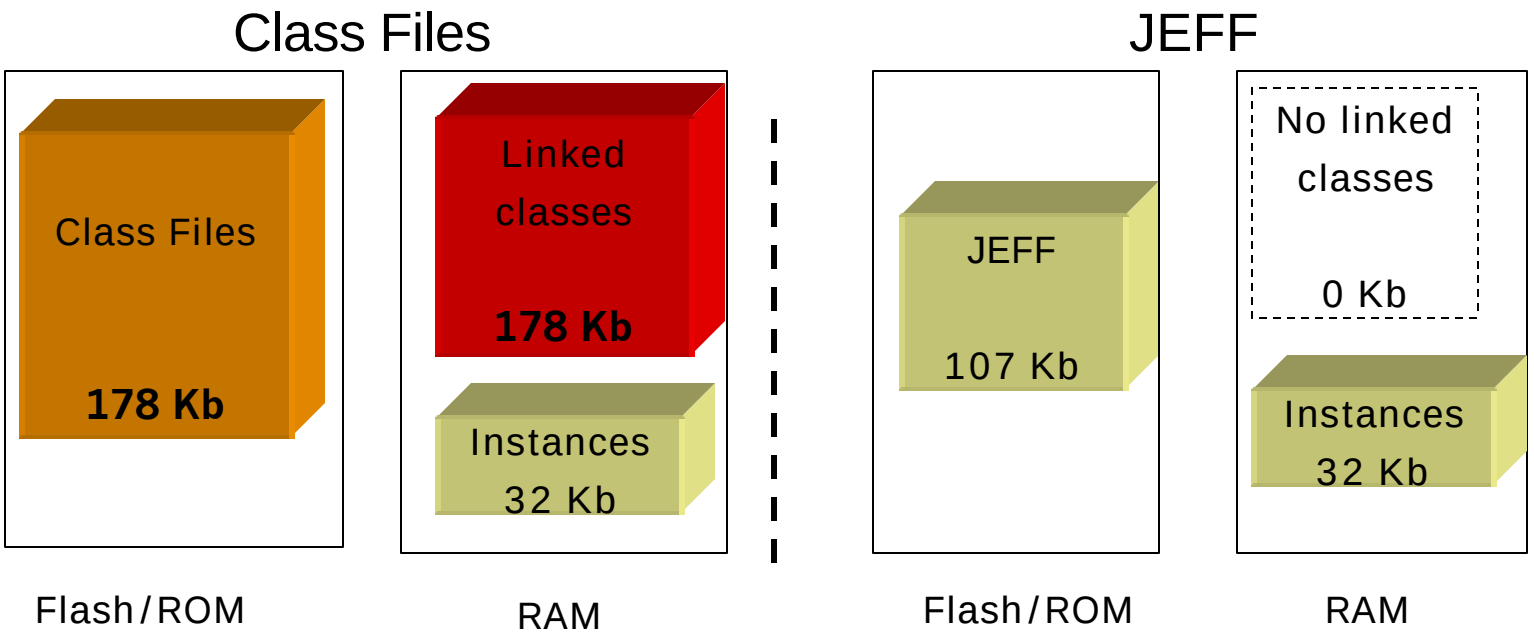


With JEFF

Average Size Reduction 40-50%

J2ME CLDC 1.0.2

	CLDC	+	Application
•(112 classes)			
• Class files	148 KB	+	30 KB
• JEFF	89 KB	+	18 KB



Summary JEFF Decisive Benefits

- JEFF is completely not pre-linked
  - each class of a JEFF file keeps its individuality and can be overriden at link time by a new class downloaded in another JEFF file if needed
- JEFF keeps the original information
  - preserves all original symbolic information and attributes
- JEFF is a strictly ready-for-execution format
  - no transformation of the original file needed for execution, even during linking: can be put in ROM.
  - completely pre-aligned for fast execution
- JEFF is compact:
  - usuall twice smaller than original .class classes, this without any compression
- JEFF can store any kind of data in addition to classes:
  ≈ JAR

CONSORTIUM

JEFF-TOOLS Availability

- Jar to JEFF Converters
- JEFF Disassemblers
- JEFF-Based VMs for STIP/FINREAD Platforms

Providers: Cardsoft, Trusted-Logic, Silicomp, Ingenico, SchlumbergerSema