Object Definition

- Example:

```
class Mytype
{
private:
  int x1;
  DO_CLASS_SETUP (Mytype);
};

class Obj : d0_Object
{
private:
  int a;
  d0_String b;

  d0_Ref<Obj> c;
  Obj* cp;

  d0_Vector<float> d;
  vector<float> ds;

  Mytype f;
  double g;  // ! This field is not persistent.
  DO_OBJECT_SETUP (Obj);
};
```

Persistent Data Members

- Fundamental types: `int`, `short`, `char`, `bool`, `float`, `double`; `d0_Int`, `d0_UInt`, `d0_Short`, `d0_UShort`, `d0_Char`, `d0_Bool`, `d0_Float`, `d0_Double`, `d0_Octet`.
- `string` and `d0_String`.
- Enumeration types.
- Previously declared structures or classs.
- References: `d0_Ref` or C++ pointers.
  - Target type must derive from `d0_Object`.
  - Same object cannot be pointed to by both `d0_Ref` and C++ pointer.
- DØOM Collections: `d0_List`, `d0_Vector`, `d0_Set`, `d0_Multiset`, `d0_Map`, `d0_Multimap`.
- STL Collections: `list`, `vector`, `set`, `multiset`, `map`, `multimap`.

Restrictions

- No C++ references.
- No bitfields.
- No unions.
- No class can appear more than once in an inheritance lattice.
- Virtual base classes cannot contain persistent data.
- Targets of pointers must derive from `d0_Object`.
- Classes deriving from `d0_Object` should include `D0_OBJECT_SETUP`(class).
- Other classes with private or protected persistent data should include `D0_CLASS_SETUP`(class).

Collections

- Available:
  - `d0_List<T>`
  - `d0_Vector<T>`
  - `d0_Set<T>`
  - `d0_Multiset<T>`
  - `d0_Map<K, T>`
  - `d0_Multimap<K, T>`
- All derive from the corresponding STL class.
- All also derive from `d0_Collection<T>`, which provides common ODMG-style operations.
- ODMG-style `d0_Iterator<T>`, created by `d0_begin` and `d0_end` methods. Also `d0_Const_Iterator<T>`.

References

- Reference type: `d0_Ref<T>`.
- `T` must derive from `d0_Object`.
- Provides:
  - Deferred reads.
  - Reference counting of objects.
- Usage:

```
d0_Ref<Obj> r;
r->func (r->memvar);
Obj tmp = *r;
Obj* p = r.ptr ();
```

- References point at objects in the same ‘event’ (or constant objects). ‘Longrefs’ planned, but not yet implemented.

Reference Counting

- Objects allocated with ‘new’ are reference-counted by default.

- Example:

```
d0_Ref<foo> a1 = new foo (1);
// foo(1) now has one reference.
{
  d0_Ref<foo> a2 = a1;
  // now it has two references.
  a1 = new foo (2);
  // now foo(1) has one ref. again.
  a2 = a1;
  // all references to foo(1) are
  //  now gone; foo(1) is deleted.
  //  foo(2) now has two references.
}
// a2 goes away.  foo(2) now has
//  one reference.
a1.clear ();
// now foo(2) is deleted.
```

- Watch out for cycles!

Bare C++ pointers

- No reference counting: must explicitly delete referenced objects.
- No deferred reads. When a structure is read containing C++ pointers, all the referenced objects are also read.
- Cannot have both a C++ pointer and a `d0_Ref` pointing at the same object.
- Target type must still derive from `d0_Object`.

Constants

- Objects which derive from `d0_Const_Object` which are logically written only once per file.

- Constant objects kept in DSPACK section 2. When writing an event, if any constant objects have been changed or added, the entire constant record is written before the event record.

- Use `mark_modified` to force objects to be rewritten.

Output Example

```
d0om_init ("test");
d0StreamFactory* fac =
  d0StreamFactory::locateStreamFactory ();

d0Stream* dsout =
  fac->make_d0Stream (fname, "DSPACK", ios::out);

while (...) {
  d0_Ref<Event> ev = ...;
  dsout->write (*ev);
}

dsout->close ();
delete dsout;
```

Input Example

```
d0om_init ("test");
d0StreamFactory* fac =
  d0StreamFactory::locateStreamFactory ();

d0Stream* dsin =
  fac->make_d0Stream (fname, "DSPACK", ios::in);

d0Key* nullkey = 0;
d0_Ref_Any in;
while (in = dsin->read(nullkey)) {
  d0_Ref<Event> ev = D0_REFCAST((Event)) (in);
  process (ev);
  ev.clear ();
}

dsin->close();
delete dsin;
```

Release Status

- Released in `d0om` CVS module.
- Latest version tagged in CVS: `v00-00-06`.
- Latest version fully released: `v00-00-05`.
- Documentation: `d0om/doc/d0om_user_guide.ps`.
- Small test programs: `d0om/tests`.
- Larger example: `d0sgi0:~greenlee/d0om/test4`.