Subject: Re: Optional serialization techniques
Posted by mirek on Fri, 25 Feb 2011 08:46:26 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 24 February 2011 10:27I have a version 1.0 of my application which serializes a VectorMap of some object into the file with StoreToFile. We of course know that if VectorMap object is being changed, the whole de-serialization is failed.
So here is my problem: I develop 1.1 version with objects which are slightly different. And actually what I want is that 1.1 version reads everything from config file ignoring the fact that objects can't be de-serialized completely (I just add more members since 1.0).
I don't want to make object members 'dynamic' (using VectorMap<String,Value> instead of plain members).
Yes, and I really don't want to use XML as speed is the most important in this case. And there is no problem just adding new members since new version (not removing old or replacing them).

Can you please suggest the most effective way of doing it?
Effective means the most quickly working while not rewriting all of U++ serialization code

I usually do:

```
void Foo::Serialize(Stream& s)
{
   int version = 0;
   s / version;
   s % x % y;
}
```

... and later I add 'z' to Foo:

```
void Foo::Serialize(Stream& s)
{
   int version = 1;
   s / version;
   s % x % y;
   if(version >= 1) {
      s % z;
   }
}
```

That said, it does not solve the problem all the time and generally, I would NOT recommend using binary serialization for permanent storage of important files. It is fine for configs (where if you loose one, it is not that bad) or for transfering data (e.g. over network).

Do not use it for documents

Mirek