Coding Conventions

Formatting

- Run AStyle script
  RunAstyle.pl
  I'm not even slightly happy about the way this looks but I've found no better alternative.

Begin/End versus start/length

STL is reasonably consistent, with most APIs using T* start, T* end, but some APIs use length instead of end. The Stroika convention is to always use T* start, T* end.

Rationale

One, this gives more consistent expectations. That's especially important for APIs that use offsets (like String) – so that it's obvious the meaning of integer parameters.

And it avoids problems with overflow. For example, if you had an API like:

```
basic_string substr(
   size_type _Off = 0,
   size_type _Count = npos
) const
```

To map this to an internal representation you have todo:

```
 char* s = m_bufPtr + _Off;
 char* e = m_bufPtr + _Off + _Count;
```

but if count was numeric_limits<size_t>::max(), then the e pointer computation would overflow. There are ways around this, but mixing the two styles creates a number of problems - but for implementations – and for use.

mk Factories

Stroika doesn't make much use of the factory pattern, but occasionally – it is useful. If the type provided by the factory is exactly the type of a given class, then we generally use

```
struct T {
    static T mk();
};
 Of course in this case, there was little obvious motivation to use a factory instead of regular constructor. However, if the class T is effectively a smart-pointer wrapper on some underlying dynamic 'rep' - this pattern may make sense.
```

But – for shared_ptr types, and typedefs, we generally use

```
struct                      X;
typedef  shared_ptr<X>      XPtr;
XPtr                        mkXPtr ();
```