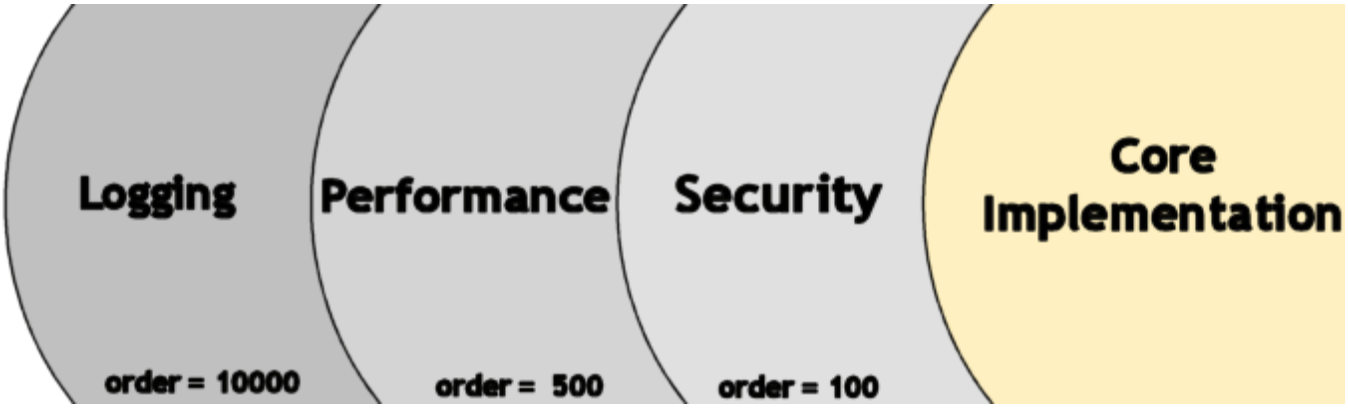
InterceptorOrderingProposal

Problem Description

As of this writing (Apr 12 2004, not quite release 1.0-alpha-4), having multiple interceptors for a single service is somewhat problematic.

Currently, when an interceptor is contributed into a service point, you need to specify its **order**, an integer value. The contributed interceptors are sorted into ascending order and applied in that order.



This is not currently a big issue, since there's only a single interceptor shipped with the framework, the logging interceptor.

However, using explicit ordering *feels* limiting. It's more natural to think in terms of ordering ... order "security" before "performance" and order "logging" after everything else.

In addition, it is likely that developers will frequently want to apply the same *set* of interceptors to many services; typically a logging interceptor, and some kind of performance monitor.

<interceptor-set>

The first potential solution is to provide a set of interceptors:

```
<service-point id="MyService" ...>
  <interceptor-set set-id="StandardSet"/>
</service-point>

<interceptor-set id="StandardSet">
   <interceptor service-id="hivemind.lib.PerformanceInterceptor"/>
   <interceptor service-id="hivemind.LoggingFactory"/>
</interceptor-set>
```

Here, instead of defining an explicit order via an `order` attribute, there's an implicit order, based on simple position.

I'm not completely thrilled with this solution.

- **Plus**: It's succinct, and makes uniformity of many services pretty easy.
- **Neutral**: Do we only allow a single <interceptor> or <interceptor-set> per <service-point>? If not, we're kind of back where we started ... need some ordering.
- **Minus**: The more interesting interceptors (such as a hypothetical security interceptor) will require service parameters, which will be unique for each service.

Ordering Via Dependencies

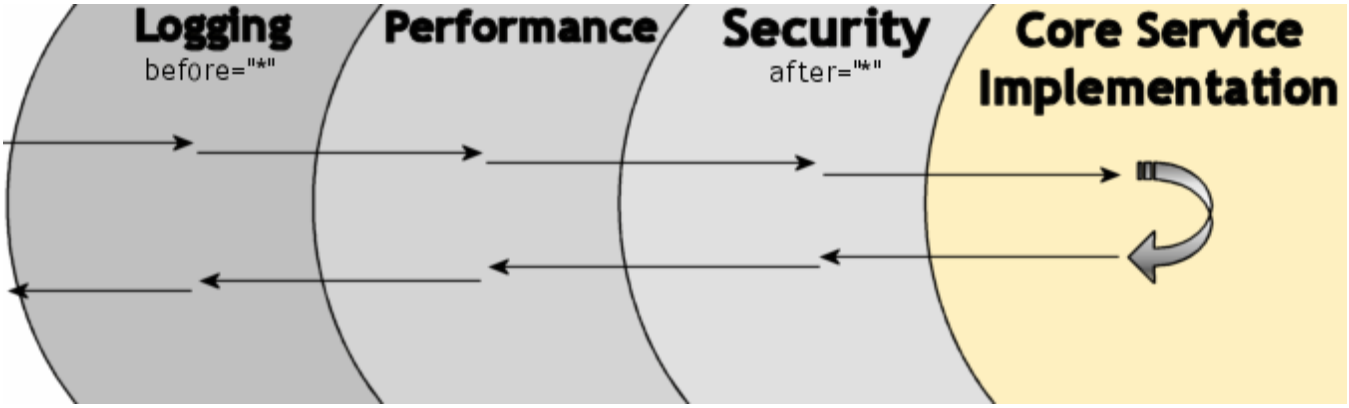
A more complex, but generally more palatable, solution is to explicitly set the dependencies, perhaps as:

```
<service-point id="MyService" ...>
  ...
</service-point>

<implementation service-id="MyService">
  <interceptor name="logging" service-id="hivemind.LoggingInterceptor" after="*"/>
</implementation>

<!-- And in a different module perhaps ... -->

<implementation service-id="MyService">
  <interceptor name="perf" service-id="hivemind.lib.PerforamnceInterceptor"/>
  <interceptor name="security" service-id="hivemind.lib.SecurityInterceptor" before="perf">
    <method name="foo" allow="admin"/>
    ...
  </interceptor>
</implementation>
```

So, each interceptor has a `name` attribute (which is presumably unique), and can specify which interceptors come before them (equivalent to lower order numbers in the current system), and which come after (equivalent to higher order numbers). In some cases, an interceptor may need to specify both `before` and `after`. In some cases, a comma-seperated list is appropriate. As in the example above, the occasional use of a wildcard is appropriate.

Status

As of April 18 2004, Ordering Via Dependencies has been largely implemented. The only piece was to add a `name` attribute to <interceptor>; currently, the ordering is done using service-id. I think this is acceptible for the meantime ... I think it will be rare to have more than two or three interceptors on a given service, and the ordering should generally be straight forward.



Along the way, I changed some of the logging concerned with service creation to use the logger for the service id. This seems like a good idea (enable logging for com.myco.MyService and you will get logging of its creation as well method invocations), and I expect to pursue this further.