Per-Proxy Request Contexts

Instead of passing a context explicitly with an invocation, you can also use a *per-proxy context.* Per-proxy contexts allow you to set a context on a particular proxy once and, thereafter, whenever you use that proxy to invoke an operation, the previously-set context is sent with each invocation. The proxy methods ice_context and ice_getContext set and retrieve the context, respectively. The Slice definitions of these methods would look as follows:

Slice

```
Object* ice_context(Context ctx);
Context ice_getContext();
```

ice_context creates a new proxy that stores the given context. Calling ice_getContext returns the stored context, or an empty dictionary if no per-proxy context has been configured for the proxy.

Here is an example in C++:

C++

```
Ice::Context ctx;
ctx["write policy"] = "immediate";

PersonPrx p1 = ...;
PersonPrx p2 = p1->ice_context(ctx);

Address a = ...;

p1->setAddress(a);       // Sends no context

p2->setAddress(a);       // Sends ctx implicitly

Ice::Context ctx2;
ctx2["write policy"] = "delayed";

p2->setAddress(a, ctx2); // Sends ctx2 instead
```

As the example illustrates, once we have created the p2 proxy, any invocation via p2 automatically sends the configured context. The final line of the example shows that it is also possible to explicitly send a context for an invocation even if the proxy is configured with a context — an explicit context always overrides any per-proxy context.

Note that, once you have set a per-proxy context, that context becomes immutable: if you subsequently change the context you have passed to ice_context, such a change does not affect the per-proxy context of any proxies you previously created with that context because each proxy on which you set a per-proxy context stores its own copy of the dictionary.

See Also

- Explicit Request Contexts
- Proxy Methods