

The corbaloc and corbaname URLs

Ciaran.McHale@iona.com
Principal Consultant
IONA Technologies

August 25, 2003

© Copyright 2003 IONA Technologies PLC. All rights reserved.

Contents

1	Introduction	1
2	The corbaloc URL	2
3	The corbaname URL	2
4	Architectural Support for corbaloc	3
4.1	Client-side Support for corbaloc	3
4.2	Server-side Support for corbaloc	3
5	Solving Bootstrapping Interoperability Problems	4

1 Introduction

URLs used on the world wide web (WWW) begin with the name of a protocol, followed by ":", for example, "http:", "ftp:" or "file:". A stringified object reference begins with "IOR:" so this also looks similar to a URL.

In early versions of CORBA, the only kind of string parameter that could be passed to `string_to_object()` was a stringified object reference. CORBA has now matured to allow *other* URL-like strings to be passed as parameters to `string_to_object()`. A CORBA product may optionally support the "http:", "ftp:" and "file:" formats. The semantics of these is that they provide details of how to download a stringified IOR (or, recursively, download another URL that will eventually provide a stringified IOR).

Although support for "http:", "ftp:" and "file:" is optional, all CORBA products must support two OMG-defined URLs: "corbaloc:" and "corbaname:". The purpose of

these is to provide a human readable/editable way to specify a location where an IOR can be obtained.

2 The corbaloc URL

Some examples of `corbaloc` URLs are shown below:

```
corbaloc:iiop:1.2@host1:3075/NameService
corbaloc:iiop:1.2@host1:3075, :iiop:1.2@host2:3075/NameService
```

The first URL specifies that an IOR can be obtained by using version 1.2 of the IIOP protocol to send a lookup-style message with parameter "NameService" to port 3075 on host `host1`.

The second URL is similar except that it specifies two `<host>:<port>` addresses rather than one. In general, any number of `<host>:<port>` addresses can be specified, separated by commas. This second form is used to provide fault tolerance: the lookup-style message will be sent to one of the addresses in the list; if that `<host>:<port>` cannot be contacted then another address in the list will be tried, and so on.

Many parts of the `corbaloc` URL have default values:

- The default protocol is `iiop`.
- If the protocol is `iiop` then the default *version* of IIOP that is used is 1.0. It is advisable to specify the most recent version of IIOP that is understood by both the client and server application. This is because more modern versions of IIOP tend to have better capabilities that might make client-server interaction more efficient.
- The default port number is 2809.

The CORBA specification currently allows two protocols to be used in `corbaloc` URLs. One protocol is `iiop`, which has already been discussed. The other protocol, `rir`, specifies that an object reference should be obtained by calling the `resolve_initial_references()` operation, passing the specified name as a parameter. For example, the `corbaloc` URL below specifies that an IOR should be obtained by passing "NameService" as a parameter to `resolve_initial_references()`.

```
corbaloc:rir:/NameService
```

3 The corbaname URL

A `corbaname` URL is a `corbaloc` that specifies how to contact the Naming Service, followed by "#" and then a name within the Naming Service. Some examples are shown below:

```
corbaname:foo.bar.com:2809/NameService#x/y/z
corbaname:host1:3075,host2:3075,host3:3075/NameService#x/y/z
corbaname:rir:/NameService#x/y/z
```

4 Architectural Support for corbaloc

4.1 Client-side Support for corbaloc

Client-side support for corbaloc and corbaname is built into the `string_to_object()` operation:

- If the parameter starts with "IOR:" then `string_to_object()` treats it as a stringified object reference and builds a corresponding proxy.
- If the parameter to `string_to_object()` starts with "corbaloc:rir" then the operation calls `resolve_initial_references()` and passes the specified name as a parameter.
- If the parameter is a corbaloc URL that uses the `iiop` protocol then the operation opens a socket connection to the specified host and port, and sends a *LocateRequest* message, using the specified name as the `object` key in the header of the message. The IOR embedded in the returned *LocateReply* message is used as the return value of `string_to_object()`. An important point to note is that corbaloc is built on top of *existing* low-level GIOP messages so the OMG did *not* have to define a new version of GIOP to support corbaloc URLs.
- If the parameter to `string_to_object()` is a corbaname URL then the embedded corbaloc details are used to locate a Naming Service. The `string_to_object()` operation then invokes `resolve_str()` on the Naming Service, passing it the string after the embedded "#" as a parameter. The IOR returned from `resolve_str()` is used as the return value of `string_to_object()`.

4.2 Server-side Support for corbaloc

The CORBA specification does *not* standardize the server-side support for corbaloc URLs, nor even the *terminology* for this server-side support. This means that CORBA products provide proprietary mechanisms, often with proprietary terminology. For example:

- The Orbix implementation repository has built-in, server-side support for corbaloc URLs, and this is referred to as *named keys*. A named key is a mapping from the *name* component in a corbaloc URL to a stringified IOR. The `named_key` sub-commands of the `itadmin` administration utility are used to create, show, list and delete named keys. By default, the Orbix implementation repository listens on port 3075 so corbaloc URLs should be formatted as shown below:

```
corbaloc:<host-of-implementation-repository>:3075/<name>
```

When the `itconfigure` utility is used to set up an Orbix domain, named keys are automatically created for whatever CORBA Services are added to the domain. For example, if the domain has a Naming Service then a named key called `NameService` is created.

Orbix does *not* expose APIs that would allow developers to embed server-side `corbaloc` support in their own server applications.

- ORBacus provides some proprietary APIs—in the `BootManager` interface—that can be used by developers to embed server-side `corbaloc` support in their own server applications. These APIs are used by the ORBacus implementation repository, which looks up *name*→*stringified-IOR* mappings in a configuration file.
- OmniORB server-side support for `corbaloc` URLs relies upon placing objects into a specific, predefined POA. OmniORB also provides a prewritten server application called `omniMapper` that can listen on a specified port number and looks up *name*→*stringified-IOR* mappings in a configuration file.

As can be seen, each CORBA product has its own different “look and feel” for server-side support of `corbaloc` URLs. Because of this, there is *no* portable way for a CORBA server to use a `corbaloc` URL to advertise one of its own objects. Developers concerned with writing portable CORBA applications should use `corbaloc` URLs only for CORBA Services, for example, the Naming Service, Notification Service, Object Transaction Service and so on.

5 Solving Bootstrapping Interoperability Problems

One obvious requirement for interoperability between different CORBA products is that they must be able to speak the same on-the-wire protocol (IIOP). However, that by itself is not sufficient. Another, less obvious requirement for interoperability is for one CORBA product to be able to *find*, say, the Naming Service or the Notification Service of another CORBA product. For example, how can an Orbix client *find* (connect to) the Naming Service of, say, an ORBacus installation. This is often called a bootstrapping problem. The `corbaloc` and `corbaname` URLs were invented to address such bootstrapping issues, as we now discuss.

A CORBA application connects to a CORBA Service—the Naming Service, Transaction Service, Notification Service, and so on—by using the name of the desired service as a parameter when calling `resolve_initial_references()`. The CORBA specification does *not* specify how `resolve_initial_references()` works (that is an implementation detail), but in most CORBA products this operation looks in a configuration file to find a *name-of-CORBA-service*→*stringified-IOR* mapping¹ and then passes the stringified IOR as a parameter to `string_to_object()`. These mappings are normally set up during the installation and configuration of a CORBA product. To configure, say, Orbix to use an ORBacus Naming Service is a matter of obtaining a stringified IOR of the ORBacus Naming Service (typically from the ORBacus configuration file) and copying this into the Orbix configuration file. Then the next time an Orbix client calls `resolve_initial_references("NameService")`, it will be directed towards the ORBacus Naming Service. This technique works fine, but it is a bit cumbersome because stringified IORs are not human readable. However, with the introduction of

¹ For example, the Orbix configuration entry is `initial_references:<service>:reference`. In ORBacus, the corresponding configuration entry is `ooc.orb.service.<service>`.

corbaloc URLs, the technique becomes much easier. Now, instead of copying a stringified IOR of the ORBacus Naming Service into the Orbix configuration file, it is sufficient to copy a corbaloc URL into the Orbix configuration file. The fact that corbaloc URLs are easy to read (and edit) by humans makes it more feasible for an organization to use several different CORBA products.²

Sometimes, practical or organizational issues may make it awkward to update a configuration file with a stringified IOR or corbaloc URL for, say, the Naming Service of another CORBA product. To work around this, the OMG defined two standard command-line options that all CORBA products must support.

The first command-line option takes the form `-ORBInitRef <name>=<value>`. An example is shown below:

```
-ORBInitRef NameService=corbaloc:iiop:1.2@host1:3075/NameService
```

The specified `<name>=<value>` pair provides a stringified IOR or URL that is be used if `<name>` is passed as a parameter to `resolve_initial_references()`. This command-line argument takes precedence over any corresponding information in the CORBA product's configuration file.

The second command-line option takes the form:

```
-ORBDefaultInitRef <URL-up-to-but-not-including-the-final-"/">
```

Some examples are shown below:

```
-ORBDefaultInitRef corbaloc:iiop:1.2@host1:3075
-ORBDefaultInitRef corbaname:iiop:1.2@host1/NameService#foo/bar
```

A call to `resolve_initial_references("<name>")`, results in the string `"/<name>"` being appended to the string provided by `-ORBDefaultInitRef`, and the result is then passed to `object_to_string()`.

If both `-ORBInitRef` and `-ORBDefaultInitRef` command-line arguments are used then the `-ORBInitRef` arguments take precedence.

² It is rare for an organization to *deliberately decide* to use several CORBA products. However, several CORBA products may make their way into an organization if different departments or development teams make independent choices about which middleware technology they will use, or if the development of some CORBA applications is outsourced to other organizations.