Release Notes

VisiBroker-RT for C++
Version 6.0

RELEASE CONTENT

The VisiBroker-RT for C++ 6.0 release is a fully functional major release. The distribution consists:

- Development tools and utilities (including the VisiBroker Smart Agent for the Development host)
- C++ header files
- VisiBroker-RT runtime libraries
- Documentation (in Acrobat PDF formats)
- Sample applications

CHANGES SINCE 5.2

Whats new in VisiBroker-RT for C++ 6.0

VisiBroker-RT for C++ 6.0 contains minor changes since the 5.x releases. The following new functionality has been added as part of VisiBroker-RT 6.0.

- OMG compliant Codec API
- Improved VisiBroker Console
- Licensed usage of idl2cpp tool.

For more information on the new features and enhancements provided by this release please see “What’s new” on page 1-1 of the VisiBroker-RT for C++ 6.0 Developer’s guide.

Upgrading from 5.x

For the most part, VisiBroker 6.0 is not binary or source compatible with previous releases of VisiBroker. This means that you will need to regenerate your stubs and recompile your applications.

Please refer to Part VIII "Backward compatibility," of the VisiBroker-RT for C++ 6.0 Developer’s guide for details on migrating VisiBroker-RT for C++ version 4.x or 5.x applications.

When upgrading, it is required that VisiBroker-RT for C++ 6.0 be installed in a separate directory from previous installations.

To use the idl2cpp development tool you must own a VisiBroker-RT for C++ license. There are no license files needed for deploying a VisiBroker-RT based application on your target systems.

CHANGES SINCE 3.2.X

Whats new in VisiBroker-RT for C++ 6.0

VisiBroker-RT for C++ 6.0 contains major changes since the 3.x releases, including major API rewrites.

Some of new features available in this release include:

- POA (Portable Object Adaptor)
- Enhanced VisiBroker Logging Service
- Interoperable Naming Service
- GIOP 1.2 support
- OBV (Objects by Value)
- New property manager
- Bi-Directional GIOP
- Firewall Support
- Quality of Service (QoS)
- OMG compliant Portable Interceptors and Codec API
- Enhanced VisiBroker Console
- Licensed usage of idl2cpp tool

For more information on the new features and enhancements provided by this release please see “What’s new” on page 1-1 of the VisiBroker-RT for C++ 6.0 Developer’s guide.

Upgrading from 3.2.x

For the most part, VisiBroker 6.0 is not binary or source compatible with previous releases of VisiBroker. This means that you will need to regenerate your stubs and recompile your applications.

Please refer to Part VIII "Backward compatibility," of the VisiBroker-RT for C++ 6.0 Developer's guide for details on migrating VisiBroker-RT for C++ version 3.2.x. applications.

When upgrading, it is required that VisiBroker-RT for C++ 6.0 be installed in a separate directory from previous installations.

IDL Compiler changes:

To use the idl2cpp development tool you must own a VisiBroker-RT for C++ license. There are no license files needed for deploying a VisiBroker-RT based application on your target systems.

IDL compiler is now case-insensitive: The IDL compiler is now case-insensitive when searching for keywords, and will collide with identifiers if they use the same spelling. This means that a keyword, spelled using a different case, is still recognized as a keyword. For example,

```
interface Interface { };
```

is no longer valid, as Interface is recognized as a keyword. The recommended fix is to rename the interface to another name. However, as a migration strategy, you could initially escape the keyword with an underscore ("_") and cause the compiler to not recognize the keyword. The resulting code will still have the correct name, ("Interface," in the above example).

-no_exceptions option: This option has been deprecated in VisiBroker-RT for C++ 6.0 . The idl compiler will always generate code making use of the VISxxxx exception macros (e.g. VISTRY,VISCATCH,...)

Modules: Modules can no longer contain types with the same name as the module.

New keywords added: New keywords have been added to the IDL language. They are: **abstract**, **custom, factory, native, private, public, supports, truncatable, valuetype, and ValueBase**.

Field names: Field names collide with types defined in the same scope, regardless of case.

Enumeration names: Enumeration names are now only accessible through their container's scope. Enumerations do not open scopes, and therefore names defined in an Enumeration cannot be accessed as though they were scoped inside the enumeration. For example:

```
// IDL module x { enum myEnum { name1, name2, name3 }; };
```

Given the above IDL, name1 inside x::myEnum can be referred to as x::name1 but not x::myEnum::name1

Interface Repository changes: The Interface Repository no longer has a graphical or console mode. Access is primarily through the ir2* and *2ir tools. The irep tool itself runs silently and can be run in the background. Please see the Developer's Guide for more information on using the irep tool.

The Interface Repository is not backward compatible. This means that older clients cannot talk to the 4.0 Interface Repository and new clients cannot talk to a pre-4.0 irep server.

Note: The Interface Repository (i.e. irep command) delivered with VisiBroker-RT for C++ 6.0 *is a Development Host Tool only. The irep is not provided as a run-time library.*

Miscellaneous changes:

- The type of system exceptions thrown by the ORB has changed in several cases to be more consistent with the CORBA 2.5 specification.
- VisiBroker now uses GIOP 1.2, but can drop down for communication with ORBs running earlier versions of GIOP.
- As part of CORBA 2.5 the functionality of CORBA::TypeCode::equal() has changed. In VisiBroker 3.x, using equal() compared the structural equivalence of typecodes. The semantics for equal() has changed in CORBA 2.5. A new function named equivalent() needs to be used wherever equal() was used while using VisiBroker 3.x to obtain the same behavior.
- The ptie approach is no longer supported since the Object Database methodology is being removed. Any use of ptie will no longer work.
- The CORBA::OctetSequence used to have an extension on it, data(), which allowed you to obtain the raw data of the sequence. This is no longer supported. Instead, the usage of OMG specified get_buffer() and replace() should be used.
- As part of CORBA 2.5 the signature of certain functions within the CORBA::Exception classes have changed.
- _narrow() now becomes _downcast() and _throw becomes _raise(). This was done to eliminate confusion of narrowing between interfaces and other classes such as CORBA::Exception.

MIGRATION: COMMON PROBLEMS AND SOLUTIONS

Use the -boa option of the idl2cpp in Makefiles in order to compile BOA over POA applications:

```
prompt> idl2cpp -boa account.idl // from the account example
```

Interoperability with other VisiBroker releases

VisiBroker-RT for C++ 6.0 is fully on-the-wire interoperable with:

- VisiBroker-RT for C++ 3.2.2
- VisiBroker-RT for C++ 4.x
- VisiBroker-RT cor C++ 5.x
- VisiBroker for C++ 3.3 and later versions
- VisiBroker for C++ 4.x
- VisiBroker for C++ 5.x
- VisiBroker for C++ 6.0
- VisiBroker for Java 3.3 and later versions
- VisiBroker for Java 4.x
- VisiBroker for Java 5.x
- VisiBroker for Java 6.0

VisiBroker-RT for C++ 6.0 is "on-the-wire" **interoperable with older versions** of VisiBroker as a **client**. VisiBroker-RT for C++ 3.2.1 as well as VisiBroker 3.2.x (on non-embedded Operating Systems) and older **clients will not interoperate on-the-wire** with VisiBroker-RT for C++ 6.0 servers.

VisiBroker-RT for C++ for Tornado Version 6.0 Supported Platforms:

- **<u>VxWorks Tornado:</u>**
  - Tornado 2.2
  - Supported Development host platforms:
    - Solaris 2.7 & 2.8
    - Windows 2000 and Windows XP
  - Supported VxWorks target CPU architectures:
    - PPC Target architecture environment, which includes:
      - PPC405 (with and without *longcall*)
      - PPC603 (with and without *longcall*)
      - PPC604 (with and without *longcall*
      - PPC860 (with and without *longcall*)
    - x86 Target architecture environment, which includes:
      - Pentium(586)
      - Pentium2
      - Pentium3
      - Pentium4

- STRONGARM Target architecture environment
- XSCALE Target architecture environment
- VxWorks Simulator architecture environment, which includes
  - SIMNT
  - SIMSPARC

CONSTRAINTS AND RESTRICTIONS

- **Non-Exception Handling libraries**
  The Non-Native Exception libraries of the VisiBroker-RT run time libraries have been created using the "-fno-except" compiler option. VisiBroker uses the OMG specified environment class to implement reentrant exception handling.

  For additional information on the VisiBroker-RT for C++ VISTRY and VISCATCH macros refer to Chapter , "Handling exceptions the *VisiBroker-RT for C++ 6.0 Developer's Guide*. *W*hen developing software which requires the use of CORBA defined exception handling, you must follow the guidelines set forth in this section.

KNOWN ISSUES AND LIMITATIONS:

- Use of the Dynamic Skeleton Interface (DSI) in the same client/server process is currently not supported.

Printed in the U.S.A.

VCR0060WW21001
0102030405-9 8 7 6 5 4 3 2 1
PDF