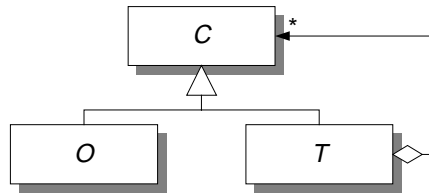


*Evaluation of COM Support in Visual J++*  
*COT/3-8-V1.0*



Centre for Object Technology

Revision history: V1.0 19-08-98 First version.

Author(s): Morten Grouleff, Aarhus University

Status: Final

Publication: Public

Summary:

This document presents the answer to a number of specific questions regarding the support for COM based development using Microsoft Visual J++.

© Copyright 1998 Aarhus University

## Introduction

This document presents the answer to a number of specific questions regarding the support for COM based development using Microsoft Visual J++.

This report is part of an evaluation of four development environments with respect to their support for COM based development. The main document of this evaluation is COT-3-4: Evaluation of COM Support in Development Environments.

## 1. Component Server

1.1 Which types of components does the tool support creation for?

*(1.1.1) No. It may be possible to hack something, though.*

---

- 1.1.1 Support for visual Controls ?  
*(1.1.1) No. It may be possible to hack something, though.*
- 

- 1.1.2 Support for non-visual Controls ?  
*(1.1.2) Yes.*
- 

1.2 Types of servers

*(1.2.1) Yes.*

---

- 1.2.1 Can the tool generate/create In-process servers?  
*(1.2.1) Yes.*
- 

- 1.2.2 Can the tool generate/create Out-of-process servers?  
*(1.2.2) Haven't been able to do so. Should be possible through "JavaReg" according to the documentation.*
- 

- 1.2.3 How can you make transition between in-process and out-of-process servers?  
*(1.2.3.1) No.*
-

- 1.2.3.1 You can do it through your IDE by clicking in some menu (where) ?  
*(1.2.3.1) No.*
- 

- 1.2.3.2 Changing the code by hand ?  
*(1.2.3.2) No.*
- 

- 1.2.3.3 Restarting the project and moving the code manually ?  
*(1.2.3.3) No.*
- 

- 1.2.3.4 Other:  
*(1.2.3.4) During registration in the registry, give "JavaReg" some extra options.*
- 

- 1.2.4 Can multiple components be contained in the same server ?  
*(1.2.4) All Java components are served by the same server, msjava.dll. It is not possible to have two servers in one .class and .Java file.*
- 

- 1.2.5 Are there any limit on the number of different components in a server ?  
*(1.2.5) A Java class file can only contain one public class. So considering the class file a server gives the answer one. But all Java classes are loaded using the same Java Virtual Machine, msjava.dll. If you consider this the server, the answer is any number of components.*
- 

### 1.3 Component construction (tool support)

- 1.3.1 How are the COM components created ?  
*(1.3.1.1) Hand coding.*
- 

- 1.3.1.1 Not using wizard. Comments:  
*(1.3.1.1) Hand coding.*
-

- 1.3.1.2 using wizard. Comments:
    - 1.3.1.2.1 Which code-generation-wizards exist (list names and usages)?
    - 1.3.1.2.2 Description of the available options for generating the code (parameters, code-templates etc.)
    - 1.3.1.2.3 Description of the architecture of the generated code ?
    - 1.3.1.2.4 Is it easy to customize the generated code ?
    - 1.3.1.2.5 Do the wizards allow for reverse-engineering ? To answer this question you should answer the following
      - 1.3.1.2.5.1 Are customizations in the generated code preserved when reverse-engineering the code ?
      - 1.3.1.2.5.2 Is it possible to change the fundamental strategies for the generated code ?
      - 1.3.1.2.5.3 To evaluate the maintenance of the generated code (customizations, additions, changes, etc.): What changes are allowed in the generated code while still retaining the possibility of reverse-engineering it ?
    - 1.3.1.2.6 Evaluation of the quality of the generated code
      - 1.3.1.2.6.1 How is the performance, stability and coding standard of the generated code ?
      - 1.3.1.2.6.2 Comments:
  - 1.3.2 Component creation using handcoding  
*(1.3.2) By writing a mostly normal public Java class. You should "import" the interfaces you wish to implement in the usual manner for Java. You should specify a ClassID by stating private static final String CLSID = "97723080-9e15-11d1-ae78-0020af72f3d6"*
- 

- 1.3.2.1 Is it easy?  
*(1.3.2.1) Reasonably easy. Apart from the magic mentioned above, the rest is normal Java programming.*
- 

## 1.4 Support for interfaces

- 1.4.1 Which types of interfaces do the tool support:  
*(1.4.1.1) No.*
- 
- 1.4.1.1 Custom/vtable ?  
*(1.4.1.1) No.*
-

- 1.4.1.2 Dispatch ?  
(1.4.1.2) Yes.
- 

- 1.4.1.3 Dual ?  
(1.4.1.3) Yes. This is the default.
- 

- 1.4.2 How do you define an interface?  
(1.4.2) If you want your class to support only one interface, you just write your class with the interface functions declared public. If you want to support multiple interfaces or need to control the chosen GUIDs, you must describe your interfaces in an IDL file.
- 

- 1.4.3 Is it possible to implement multiple interfaces on a component ?  
(1.4.3) Yes
- 

- 1.4.3.1 If Yes, how ?  
(1.4.3.1) By describing the interfaces in an IDL file.
- 

- 1.4.4 Predefined interfaces:  
(1.4.4.1) Yes.
- 

- 1.4.4.1 Is it possible to implement predefined interfaces, i.e. implementing interfaces defined by other people (preserving IID's)?  
(1.4.4.1) Yes.
- 

- 1.4.4.2 What do you use as foundation for the predefined interface ?  
(1.4.4.2) See below.
- 

- 1.4.4.2.1 Type lib ?  
(1.4.4.2.1) Yes
-

- 1.4.4.2.2 IDL ?  
*(1.4.4.2.2) Yes*
- 

- 1.4.4.2.3 Source code ?  
*(1.4.4.2.3) No.*
- 

- 1.4.4.2.4 other  
*(1.4.4.2.4) You can also use a compiled .class file that contains the definitions from an IDL file.*
- 

- 1.4.4.3 How do you use a predefined interface?  
*(1.4.4.3) By compiling the IDL and typelibrary into a class-file which defines the interfaces. You then implement them in the usual manner for Java interfaces.*
- 

- 1.4.5 Interface maintenance  
*(1.4.5.1) Yes, if you modify the IDL file.*
- 

- 1.4.5.1 Which types of changes in a published interface are allowed ?  
*(1.4.5.1) Yes, if you modify the IDL file.*
- 

- 1.4.5.1 Insert new methods and/or parameters ?  
*(1.4.5.1) Yes, if you modify the IDL file.*
- 

- 1.4.5.2 Change existing methods and/or parameters ?  
*(1.4.5.2) Yes, if you modify the IDL file.*
- 

- 1.4.5.3 Delete methods and/or parameters ?  
*(1.4.5.3) Yes, if you modify the IDL file.*
-

- 1.4.5.4 What are the consequences of changes on the generated component ?  
*(1.4.5.4) It is regenerated and the previous component is forgotten without warning.*
- 

- 1.4.6 Interface optimizations:  
*(1.4.6.1) None.*
- 

- Which type of interface optimization are supported:
    - 1.4.6.1 None?  
*(1.4.6.1) None.*
- 

- 1.4.6.2 Tear-of interfaces (a tear-of interface is part of an aggregated component. The component will not be instantiated until the interface is actually queried), How: ?  
*(1.4.6.2) None.*
- 

- 1.4.6.3 Other:  
*(1.4.6.3) None.*
- 

## 1.5 Reference counting

*(1.5.1) Reference counting is automatic. It is taken care of by the Java Virtual Machine (JVM)*

---

- 1.5.1 A description of aspects of reference counting  
*(1.5.1) Reference counting is automatic. It is taken care of by the Java Virtual Machine (JVM)*
- 
- 1.5.1.1 This question is intentionally left blank  
*(1.5.1.1) N/A.*
-

- 1.5.1.2 Is reference counting automatically supported for aggregated components?  
*(1.5.1.2) Yes. Not all Java components can be aggregated, nor can Java aggregate all components.*
- 

- 1.5.2 How is reference counting implemented in the tool ?  
*(1.5.2) Reference counting is automatic. It is taken care of by the Java Virtual Machine (JVM)*
- 

- 1.5.2.1 Is AddRef called automatically by a generated component when returning an interface reference ?
- 1.5.2.2 Is Release called automatically by a generated component when exiting scope for an interface reference received as a parameter ?
- 1.5.2.3 What happens when the reference count drops to 0 ?
  - 1.5.2.3.1 Is it possible to customize what happens when the reference count drops to 0 ?
- 1.5.2.4 How does the tools mechanism for implementing reference counting affect flexibility etc. ?

## 1.6 Threading

- 1.6.1 Which threading models are supported ?  
*(1.6.1.2) Not Specified.*
- 

- 1.6.1.2 Not specified ?  
*(1.6.1.2) Not Specified.*
- 

- 1.6.1.3 Single ?  
*(1.6.1.3) Not Specified.*
- 

- 1.6.1.4 Apartment ?  
*(1.6.1.4) Not Specified.*
- 

- 1.6.1.5 Free ?  
*(1.6.1.5) Not Specified.*

- 1.6.1.6 Other:  
(1.6.1.6) *Not Specified.*
- 

- 1.6.2 Multi threaded components:  
(1.6.2) *Not Specified.*
- 

- 1.6.2.1 Can a component be multi-threaded ?  
(1.6.2.1) *Not Specified.*
- 

- 1.6.2.2 If no, is multi-threading supported through language facilities (i.e. in native java) or is it near API ?  
(1.6.2.2) *Not Specified.*
- 

## 1.7 Throwing errors (1.7.1) *no.*

---

- 1.7.1 Do generated components support:  
(1.7.1) *no.*
- 

- 1.7.1.1 ISupportErrorInfo ?  
(1.7.1.1) *No.*
- 

- 1.7.1.2 this question is intentionally left blank
  - 1.7.2 Direct support for HRESULT  
(1.7.2) *No.*
- 

- 1.7.2.1 Does the tool support the use of existing HRESULT's ?  
(1.7.2.1) *No.*
-

- 1.7.2.2 Does the tool support the use of interface dependent HRESULTS ?  
(1.7.2.2) *No.*
- 

- 1.7.2.3 Does the tool provide support for HRESULT creation ?  
(1.7.2.3) *No.*
- 

- 1.7.2.4 Is it easy to create/use HRESULTS ?  
(1.7.2.4) *No.*
- 

- 1.7.3 Are there mappings between HRESULTS and native language exceptions ?  
(1.7.3) *Yes.*
- 

- 1.7.3.1 Is the mapping manual ?  
(1.7.3.1) *No.*
- 

- 1.7.3.2 Is the mapping automatic ?  
(1.7.3.2) *Yes.*
- 

- 1.7.4 How does the mapping work ?  
(1.7.4) *HRESULTS are completely hidden. Standard error HRESULTS are mapped into the corresponding exceptions. How well it works with custom errors has not been investigated.*
- 

## 1.8 Datatypes

(1.8.1) *Automatically for those supported.*

---

- 1.8.1 How are data-types of the development tools mapped to COM data types ?  
(1.8.1) *Automatically for those supported.*
-

- 1.8.1.1 Simple datatypes ?  
*(1.8.1.1) Yes: 8 bit and 16 bit characters, 32 bit integers and 64 bit floating point numbers are automatically mapped.*
- 

- 1.8.1.2 Strings ?  
*(1.8.1.2) BSTR is translated to String in Java. No other kinds of strings are allowed. Not even WCHAR\*.*
- 

- 1.8.1.3 Predefined types (automation compliant types)?  
*(1.8.1.3) No.*
- 

- 1.8.1.4 User-defined types (e.g. struct, enum)?  
*(1.8.1.4) no.*
- 

- 1.8.1.5 Object parameters (interface references) ?  
*(1.8.1.5) Yes.*
- 

- 1.8.1.6 Variants ?  
*(1.8.1.6) Some support through library functions, but quite hard work to use.*
- 

- 1.8.1.7 SafeArrays ?  
*(1.8.1.7) Some support through library functions, but hard work to use.*
- 

## 1.9 Marshalling

*(1.9.1) It is not visible to the programmer.*

- 1.9.1 How can marshalling be handled:  
*(1.9.1) It is not visible to the programmer.*
    - 1.9.1.1 Don't know! ?  
*(1.9.1.1) No.*
-

- 1.9.1.2 OLE32.DLL ?  
(1.9.1.2) No.
- 

- 1.9.1.3 Own Proxy/Stub DLL ?  
(1.9.1.3) No.
- 

- 1.9.1.3.1 Can it be attached to the component DLL?  
(1.9.1.3.1) No.
- 

- 1.9.1.4 Directly made by tool ?  
(1.9.1.4) *It is handled by the JVM probably in combination with the compiler.*
- 

- 1.9.1.5 Work around ?  
(1.9.1.5) *Not as far as I know.*
- 

- 1.9.2 Does the tool support IMarshal ?  
(1.9.2) *No support, but it may be possible to implement it yourself.*
- 

1.10 How does the server support typelibraries  
(1.10.1) Yes.

---

- 1.10.1 IDL Integration ?  
(1.10.1) Yes.
- 

- 1.10.1.1 IDL/ODL ?  
(1.10.1.1) *Yes again?*
-

- 1.10.2 Implicit creation of typelibraries ?  
(1.10.2) *Yes.*
- 

- 1.10.3 Integration of typelibraries as resource in server ?  
(1.10.3) *No.*
- 

- 1.10.4 Are errors in the type library able to "cheat" the tools ?  
(1.10.4) *Don't know.*
- 

## 2. Compilation and Distribution

2.1 Is version compatibility checking performed during compilation and how ? 2.2  
Interface compatibility  
(2.2.1) *Yes.*

---

- 2.2.1 Is it possible to change an interface and keep the same IID?  
(2.2.1) *Yes.*
- 

2.3 Which types of target code does the tool produce (i.e. native/optimized native +  
runtime, p-code, nothing/interpreted) ?  
(2.3.1) *Usually. It is up to the JVM what to do.*

---

2.4 How does the tool support registration of components ?  
(2.4.1) *No.*

---

- 2.4.1 Components are selfregistering ?  
(2.4.1) *No.*
- 

- If Yes then
  - 2.4.1.1 DLL RegSvr32 ?  
(2.4.1.1) *No.*

- 2.4.1.2 EXE: selfregistering when run once ?  
(2.4.1.2) *No.*
- 

- 2.4.1.3 EXE: selfregistering when compiled ?  
(2.4.1.3) *No.*
- 

- 2.4.2 Registration is done through  
(2.4.2.1) *No.*
- 

- 2.4.2.1 Installation ?  
(2.4.2.1) *No.*
- 

- 2.4.2.2 Has to be done manually ?  
(2.4.2.2) *Yes.*
- 

- 2.4.3 Does the tool support unregistration of components ?  
(2.4.3) *No. There is a command line tool for that purpose.*
- 

- 2.4.4 Does the tool support registration using ProgID ?  
(2.4.4) *No. There is a command line tool for that purpose.*
- 

- 2.4.4.1 Can the developer freely choose a components ProgID ?
  - 2.4.5 Does the tool support version independent ProgID's, and if yes how ?  
(2.4.5) *No. There is a command line tool for that purpose.*
- 

2.5 Support for component categories:  
(2.5.1) *Not sure, probably no.*

---

- 2.5.1 Is it possible to define and implement your own component categories ?  
(2.5.1) *Not sure, probably no.*

- 2.5.1.1 How ?  
(2.5.1.1) ???
- 

- 2.5.2 Does the tool have support for predefined component categories and how ?  
(2.5.2) No
- 

2.6 Can the developer freely choose CLSID for a component ?  
(2.6) Yes. It is done through the IDL file.

---

## 3. Component Client

### 3.1 Support for COM clients

#### 3.1.1 Creating and deleting components

(3.1.1.1) Yes. Components are created using the same mechanism as for regular Java classes.

---

- 3.1.1.1 Is it easy to create new components ?  
(3.1.1.1) Yes. Components are created using the same mechanism as for regular Java classes.
- 

- 3.1.1.2 Is it easy to delete existing components ?  
(3.1.1.2) Yes.
- 

- 3.1.1.3 Does the tool delete registry entries for deleted components after recompilation ?  
(3.1.1.3) No.
- 

- 3.1.1.4 This question is intentionally left blank

### 3.1.2 COM Library Support

(3.1.2.1) *No. This does not make any sense.*

---

- 3.1.2.1 Does the tool allow direct access to COM functionality, like calling COM functions direct (CoCreateInstance, CreateInstance, ...). ?  
(3.1.2.1) *No. This does not make any sense.*
- 

- 3.1.2.2 Does the tool encapsulate COM functionality (indirect COM support) ?  
(3.1.2.2) *Yes.*
- 

- 3.1.2.3 Does the tool provide COM Helper Classes ?  
(3.1.2.3) *Yes. There are helper classes for SafeArray and Variant.*
- 

- 3.1.2.4 Is it possible to combine native and encapsulated COM support ?  
(3.1.2.4) *No.*
- 

### 3.1.3 Acquisition of interfaces

(3.1.3.1) *No.*

---

- 3.1.3.1 Is it possible to call QueryInterface directly ?  
(3.1.3.1) *No.*
- 

- 3.1.3.2 Does the tool encapsulate QueryInterface ?  
(3.1.3.2) *Yes. Use an ordinary typecast in Java.*
- 

- 3.1.3.3 Reference counting ?  
(3.1.3.3) *Handled by the JVM Garbage Collector.*
- 

- 3.1.3.3.1 Is AddRef called automatically by a generated client when using an interface as parameter for a (server) method ?

- 3.1.3.3.2 Is Release called automatically by a generated client when exiting scope for an interface reference ?

### 3.1.4 Component binding

(3.1.4.1) You must know the CLSID of the class to instantiate.

---

- 3.1.4.1 Is early binding supported through: ?  
(3.1.4.1) You must know the CLSID of the class to instantiate.
- 

- 3.1.4.1.1 VTable binding ?  
(3.1.4.1.1) Yes.
- 

- 3.1.4.1.2 DispID binding ?  
(3.1.4.1.2) No.
- 

- 3.1.4.2 Is late binding supported and how ?  
(3.1.4.2) No.
- 

- 3.1.4.3 Binding through ProgID
    - 3.1.4.3.1 Does the tool support binding of (server) component through version independent ProgID ?  
(3.1.4.3.1) No.
- 

- 3.1.4.3.2 Does the tool support binding of (server) component through version specific ProgID ?  
(3.1.4.3.1) No.
- 

3.1.5 Is the process of catching errors supported through:  
(3.1.5) Yes.

---

- 3.1.5.1 native COM support (i.e. using HRESULT) ?  
(3.1.5.1) No
-

- 3.1.5.2 encapsulated support (in the tool language/environment/library) ?  
(3.1.5.2) *Yes.*
- 

## 3.2 Aggregation

3.2.1 Can the tool control COM-aggregation for produced components ?  
(3.2.1) *Yes. Only one class may be aggregated.*

---

- If yes, Supported methods
    - 3.2.1.1 Can you specify that a component is not aggregatable ?  
(3.2.1.1) *Don't know. Is possible in IDL, but it is unknown if the IDL to Java compiler respects it.*
- 

- 3.2.1.2 Is the aggregation support Wizard oriented (GUI) ?  
(3.2.1.2) *No.*
- 

- 3.2.1.3 Is Manual programming of aggregation Easy/Difficult ?  
(3.2.1.3) *Using normal inheritance in Java, which is why only one class may be aggregated. (Java does not support multiple inheritance)*
- 

## 3.3 Delegation

3.3.1 Support for delegation is:  
(3.3.1.1) *Yes.*

---

- 3.3.1.1 Manual ?  
(3.3.1.1) *Yes.*
- 

- 3.3.1.2 Tool ?  
(3.3.1.2) *No.*
-

## 4. Development Environment support

### 4.1 Description and evaluation of supporting tools/facilities in the development tool

(4.1.1) *None.*

---

- 4.1.1 What kind of object browser facilities are provided ?  
(4.1.1) *None.*
- 

- 4.1.2 What kind of syntax help/info for COM-components and methods herein ?  
(4.1.2) *None.*
- 

- 4.1.3 Does the tool provide context sensitive help for components (to be used by component integraters) ?  
(4.1.3) *Nothing related to COM.*
- 

- 4.1.4 Support for interface prototypes (components implementing interfaces specified externally)  
(4.1.4) *No. But a Java class implementing an interface must implement all functions in it, and that goes for COM interfaces as well.*
- 

- 4.1.4.1 How can you make this manually ?  
(4.1.4.1) *You have to type the code yourself.*
- 

- 4.1.4.2 What kind of toolsupport is provided (the tool provides prototypes for the methods of the interface). Filling out the slots ?  
(4.1.4.2) *No.*
- 

- 4.1.5 Debugging  
(4.1.5) *Source-level.*
-

- 4.1.5.1 Which facilities exist for debugging COM-components ?  
*(4.1.5.1) Apparently no special support.*
- 

- 4.1.5.2 Is it possible in the development to debug from client into component ?  
*(4.1.5.2) Yes, when it is also in Java. Otherwise no.*
- 

- 4.1.5.3 If any, what are the differences between debugging in-process and out-of-process servers ?  
*(4.1.5.3) Don't know, as out-of-process has never worked.*
- 

#### 4.2 Evaluation of the development tool

*(4.2.1) Generally, the Developer Studio is reasonably easy to use. It takes a slight bit of getting used to. The support for Java specifics is not very strong. Programming Java basically requires you to type in the code from scratch.*

---

- 4.2.1 A description of the work load  
*(4.2.1) Generally, the Developer Studio is reasonably easy to use. It takes a slight bit of getting used to. The support for Java specifics is not very strong. Programming Java basically requires you to type in the code from scratch.*
- 

- 4.2.1.1 Does the COM-based part of the development integrate nicely and consistently into the tool ?  
*(4.2.1.1) COM is pretty much invisible, except when exporting a Java class as a COM component. On the other hand there is not much assistance to help you do COM-based development.*
- 

- 4.2.1.1.1 Is it easy to find out *how* to do COM-based development in the tool ?  
*(4.2.1.1.1) You must read the documentation, but there isn't much to learn.*
-

- 4.2.1.1.2 Is the COM-based development in the tool similar to non-COM-based development ?  
*(4.2.1.1.2) Yes.*
- 

- 4.2.1.2 Is it easy for novices/experts with respect to the COM-development in general/the development tool specifically to do COM-based development in the tool.  
*(4.2.1.2) An seasoned Java programmer will probably find it relatively easy to do COM programming. Experienced C++ COM programmers will probably find the restrictions on datatypes and the lesser degree of flexibility troubling at first, but what annoys them in COM-in-Java probably annoys them in pure Java as well, as there is so little difference.*
- 

## 5. Development Processes

- 5.1 Is it possible to develop and test at the same time ?  
*(5.1.1) No.*
- 

- 5.1.1 Will recompilation require change of server GUID and IID's ?  
*(5.1.1) No.*
-