

# Using the COBOL Wrapper for the Client Side

The COBOL Wrapper provides access to RPC-based components from COBOL applications and enables users to develop both clients and servers. This section introduces the various possibilities for RPC-based client applications written in COBOL.

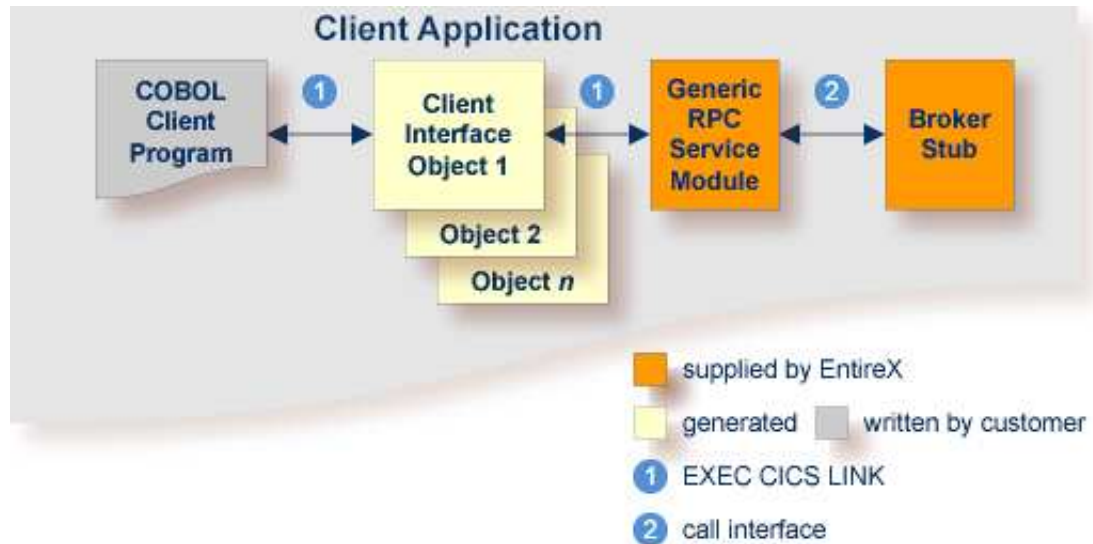
- Using the COBOL Wrapper for CICS with DFHCOMMAREA Calling Convention (z/OS and z/VSE)
- Using the COBOL Wrapper for CICS with Call Interfaces (z/OS and z/VSE)
- Using the COBOL Wrapper for Batch (z/OS, BS2000/OSD, z/VSE and IBM i)
- Using the COBOL Wrapper for IMS (z/OS)
- Using the COBOL Wrapper for IDMS/DC with Call Interfaces (z/OS)
- Using the COBOL Wrapper for Micro Focus (UNIX and Windows)

A step-by-step guide is provided in the section *Writing Applications with the COBOL Wrapper*. Read this section first before writing your first RPC client program.

---

## Using the COBOL Wrapper for CICS with DFHCOMMAREA Calling Convention (z/OS and z/VSE)

This mode applies to z/OS and z/VSE.




In this scenario, the generic RPC services module and the broker stub are linked together to a CICS program. The COBOL client program, every generated client interface object and the generic RPC services module together with the broker stub are installed each as separate individual CICS programs.

Use the COBOL Wrapper for CICS with DFHCOMMAREA calling convention in the following situations:

- You want to have an EXEC CICS LINK DFHCOMMAREA interface to your client interface object(s).
- The restriction of the COMMAREA length suits your purposes. Because the RPC communication area is also transferred in the COMMAREA, the effective length that can be used for IDL data is shorter than the CICS COMMAREA length. Nearly 31 KB can be used for IDL data.
- You wish to separate the generic RPC service module and the broker stub from the client interface object(s).
- You require a program link to the client interface object(s).

### ➤ To use the COBOL Wrapper for CICS with DFHCOMMAREA calling convention

1. Generate the client interface object for the target operating system, for example "z/OS", and use interface type "CICS with DFHCOMMAREA calling convention". See *Generating COBOL Source Files from Software AG IDL Files*. You do not need to generate the generic RPC service module COBSRVI because this is already installed on the mainframe, so clear the check box.  More info
2. If necessary, use FTP to transfer the client interface object(s) and, if required, also the generic RPC service module COBSRVI, to the target platform where you write your client application.

3. If you have generated the generic RPC service module and you plan to (re)install it within CICS, you may need to adapt the broker stub that supports the required transport (TCP, SSL, NET). See *Adapting the Used Broker Stub*.
4. Write your COBOL client program. See *Writing Applications with the COBOL Wrapper*, in particular the section *Using the RPC Communication Area with a Standard Call Interface*, and take into consideration the information given in *Software AG IDL to COBOL Mapping*.
5. Using the CICS translator for COBOL provided with your CICS installation and a COBOL compiler supported by the COBOL Wrapper, translate and compile:
  - the generated client interface object(s)
  - if required, the generic RPC service module COBSRVI
  - your COBOL client program.

Take care the generated copybooks (see *Using the Generated Copybooks*) are accessed correctly by the compiler and not confused with the client interface objects, because the copybooks and client interface objects have identical file names. See your compiler documentation.

6. Using the standard linker (binder) of the target platform, link (bind) the following programs to separate CICS programs:
  - every generated client interface object
  - if required, the generic RPC service module COBSRVI together with a broker stub
  - your COBOL client program.
7. Install every client interface object, if required the CICS RPC service module COBSRVI and your COBOL client program as separate CICS programs.
8. Make sure the correct broker stub is used and can be called dynamically by the CICS generic RPC service module COBSRVIC.

Under **z/OS**:

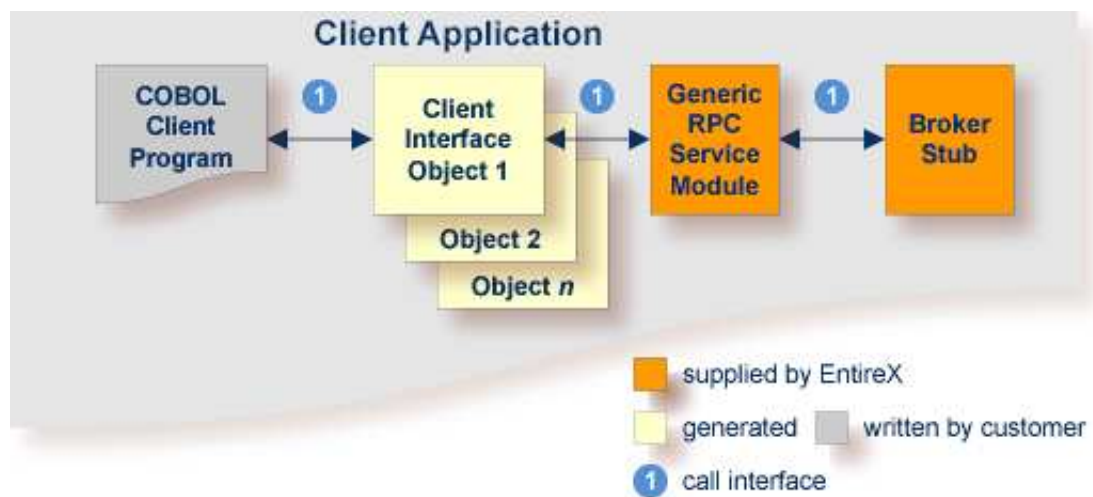
- See the broker installation documentation and use a broker stub for CICS (for example CICSETB) from the common load library EXX970.LOAD. See also *Administering Broker Stubs*.

Under **z/VSE**:

- See the broker installation documentation and use a broker stub for CICS (for example BKIMC), see sublibrary EXX970.

## Using the COBOL Wrapper for CICS with Call Interfaces (z/OS and z/VSE)

This mode applies to z/OS and z/VSE.




The COBOL Wrapper can be used with a call interface, even in CICS. This means you can build a client application where the COBOL client program, every generated client interface object, the generic RPC services module and the broker stub are linked together, similar to the batch scenario. See *Using the COBOL Wrapper for Batch (z/OS, BS2000/OSD, z/VSE and IBM i)*.

Using a call interface within CICS may be useful if

- the restriction of the COMMAREA length (about 31 KB) prevents you from using the *Using the COBOL Wrapper for CICS with DFHCOMMAREA Calling Convention (z/OS and z/VSE)* scenario (see above)
- you do *not* require a distributed program link (CICS DPL) to your client interface object(s)
- you prefer a call interface instead of EXEC CICS LINK to your client interface objects.

### ➤ To use the COBOL Wrapper with a call interface within CICS

1. Generate the client interface object(s) for the target operating system, for example "z/OS", and use the interface type "CICS with standard calling convention". See *Generating COBOL Source Files from Software AG IDL Files*. You do not need to generate the generic RPC service module COBSRVI, so clear the check box.  More info
2. If necessary, use FTP to transfer the client interface object(s) and, if required, also the generic RPC service module COBSRVI, to the target platform where you write your client application.
3. If you have generated the generic RPC service module and you plan to (re)install it within CICS, you may need to adapt the broker stub that supports the required transport (TCP, SSL, NET). See *Adapting the Used Broker Stub*.

4. Write your COBOL client program. See *Writing Applications with the COBOL Wrapper*, in particular the section *Using the RPC Communication Area with a Standard Call Interface*, and take into consideration the information given in *Software AG IDL to COBOL Mapping*.
5. Using the CICS translator for COBOL provided with your CICS installation and a COBOL compiler supported by the COBOL Wrapper, translate and compile:
  - the generated client interface object(s)
  - if required, the generic RPC service module COBSRVI
  - your COBOL client program

Take care the generated copybooks (see *Using the Generated Copybooks*) are accessed correctly by the compiler and not confused with the client interface objects, because the copybooks and client interface objects have identical file names. See your compiler documentation.

6. Using the standard linker (binder) of the target platform, link (bind) all translated and compiled modules, and, if required, the broker stub, together to the client application (that is, a CICS program), using the standard linker (binder) of the target platform.
7. Install the client application within CICS.
8. Make sure the correct broker stub is used and can be called dynamically by the generic RPC service module COBSRVI.

Under **z/OS**:

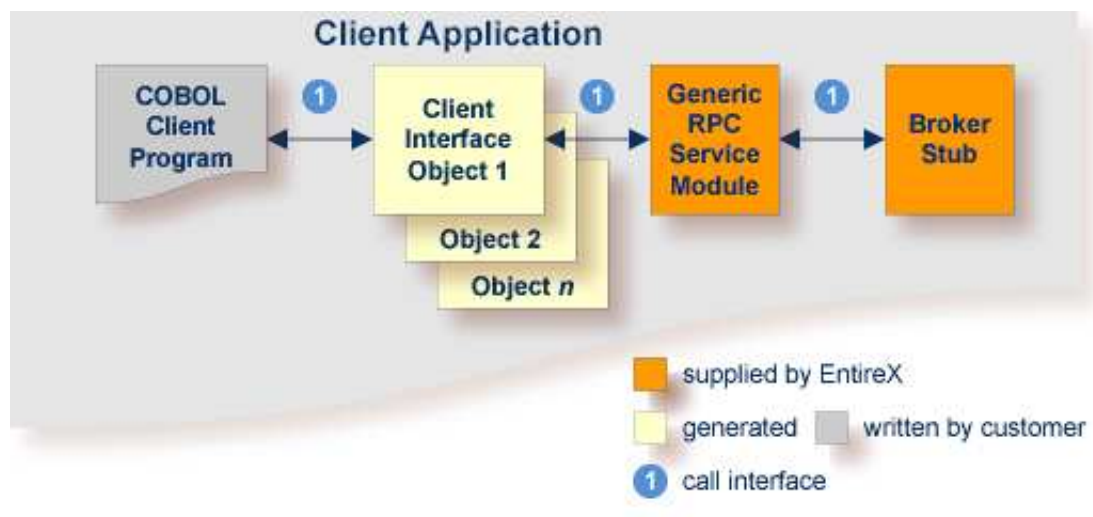
- See the broker installation documentation and use a broker stub for CICS (for example CICSETB) from the common load library EXX970.LOAD. See also *Administering Broker Stubs*.

Under **z/VSE**:

- See the broker installation documentation and use a broker stub for CICS (for example BKIMC), see sublibrary EXX970.

## Using the COBOL Wrapper for Batch (z/OS, BS2000/OSD, z/VSE and IBM i)


This mode applies to z/OS, BS2000/OSD, z/VSE and IBM i.



In this scenario, the COBOL client program, every generated client interface object, generic RPC services module and the broker stub are linked together to the client application.

Use the COBOL Wrapper for batch if you need to embed the client interface object into your application with a standard linkage calling convention.

### ➤ To use the COBOL Wrapper for batch


1. Generate the client interface object(s) for the target operating system, for example "z/OS", and use interface type "Batch with standard linkage calling convention". See *Generating COBOL Source Files from Software AG IDL Files*. You do not need to generate the generic RPC service module COBSRVI because it is already installed on the mainframe, so clear the check box.  More info
2. If necessary, use FTP to transfer the client interface object(s) and, if required, also the generic RPC service module COBSRVI, to the target platform where you write your client application.
3. If you have generated the generic RPC service module and you plan to (re)install it, you may need to adapt the broker stub that supports the required transport (TCP, SSL, NET). See *Adapting the Used Broker Stub*.
4. Write your COBOL client program. See *Writing Applications with the COBOL Wrapper*, in particular the section *Using the RPC Communication Area with a Standard Call Interface*, and take into consideration the information given in *Software AG IDL to COBOL Mapping*.
5. Using a COBOL compiler supported by COBOL Wrapper, compile:
  - the generated client interface object(s)

- if required, the generic RPC service module COBSRVI
- your COBOL client program

Take care the generated copybooks (see *Using the Generated Copybooks*) are accessed correctly by the compiler and not confused with the client interface objects, because the copybooks and client interface objects have identical file names. See your compiler documentation.

Under **BS2000/OSD**:

- 

The IDL types U or UV require a compiler that supports COBOL data type NATIONAL. See *BS2000/OSD Prerequisites* for more information on supported compilers.  [More info](#)

Under **IBM i**:

- Use the command CRTCBMOD (create COBOL module) and compile all modules above to ILE modules.
- Use the IBM i compiler command with the options shown below:

```
CRTCBMOD
OPTION(*NOMONOPRC) EXTDSOPT(*NODFRWRT) LINKLIT(*PRC)
```

On all **other platforms**:

- Use the standard COBOL compiler of the target platform.
6. Using the standard linker (binder) of the target platform, link (bind) the following programs:
- the generated client interface object(s)
  - if required, the generic RPC service module COBSRVI
  - if required, the broker stub
  - your COBOL client program

Under **IBM i**:

- Use the IBM i command CRTPGM to bind all compiled modules to an executable ILE program of type \*PGM.  
To link the main program, use the following create program command with the options shown:

```
CRTPGM
MODULE(*LIB/myapplication mystub1 mystub2 ..)
BNDSRVPGM(EXX/EXA) ...
```

where EXX is the EntireX product library and EXA the broker stub.

On all **other platforms**:

- Refer to your standard linker (binder) documentation.
7. Make sure that the correct broker stub module is used and, if linked (bound) dynamically, that it can be called dynamically.


Under **BS2000/OSD**:

- The broker stub module BKIMBTIA is located in the broker LMS load library.

Under **IBM i**:

- The broker stub EXA is located by default in the EntireX product library EXX.

Under **z/OS**:

- See the broker installation documentation and use a broker stub for batch (for example BROKER) from the common load library EXX970.LOAD. See also *Administering Broker Stubs*.  
 More info

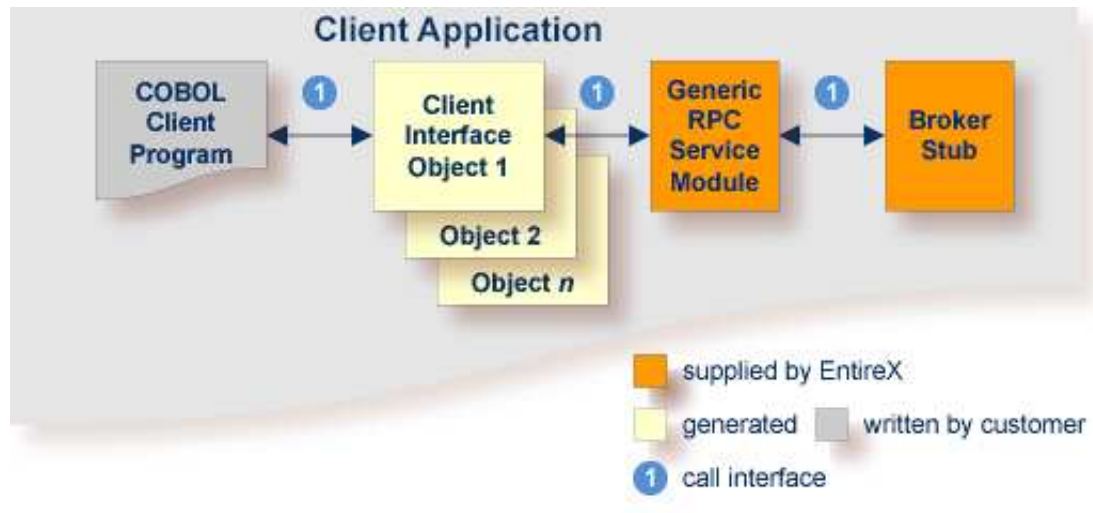
Under **z/VSE**:

- See the broker installation documentation and use a broker stub for batch (for example BKIMB), see sublibrary EXX970.



## Using the COBOL Wrapper for IMS (z/OS)


This mode applies to z/OS IMS modes BMP and MPP.



In this scenario, the COBOL client program, every generated client interface object, the generic RPC services module and the broker stub are linked together to the client application.

Use the COBOL Wrapper for IMS if you need to embed the client interface object into your IMS BMP or IMS MPP application with a standard linkage calling convention.

### ➤ To use the COBOL Wrapper for IMS

1. Generate the client interface object(s) for the target operating system "z/OS" and use the interface type "IMS BMP with standard linkage calling convention" or "IMS MMP with standard linkage calling convention". See *Generating COBOL Source Files from Software AG IDL Files*. Check the option "Generate the generic RPC service module COBSRVI".  More info.
2. If necessary, use FTP to transfer the client interface object(s) and, if required, also the generic RPC service module COBSRVI, to the target platform where you write your client application.
3. If you have generated the generic RPC service module and you plan to (re)install it, you may need to adapt the broker stub that supports the required transport (TCP, SSL, NET). See *Adapting the Used Broker Stub*.
4. Write your COBOL client program. See *Writing Applications with the COBOL Wrapper*, in particular the section *Using the RPC Communication Area with a Standard Call Interface*, and take into consideration the information given in *Software AG IDL to COBOL Mapping*.
5. Using a COBOL compiler supported by the COBOL Wrapper, compile:
  - the generated client interface object(s)
  - if required, the generic RPC service module COBSRVI

- your COBOL client program.

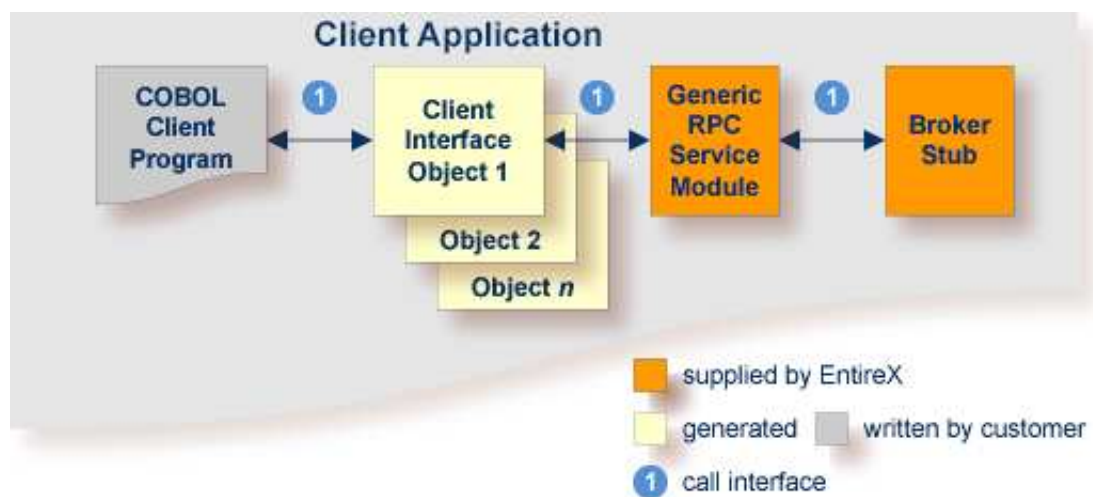
Take care the generated copybooks (see *Using the Generated Copybooks*) are accessed correctly by the compiler and not confused with the client interface objects, because the copybooks and client interface objects have identical file names. Do not assign the data set with the client interface objects prior in sequence to the copybooks to SYSLIB. See your compiler documentation.

6. Link (bind) all compiled modules and, if required, the broker stub, together to an executable program, using the standard linker (binder) of the target platform.
7. Make sure the correct broker stub is used and can be called dynamically. In the common load library EXX970.LOAD you can find broker stubs that can be used for
  - IMS BMP (for example BROKER)
  - IMS MPP (for example MPPETB)

See *Administering Broker Stubs*.

## Using the COBOL Wrapper for IDMS/DC with Call Interfaces (z/OS)


This mode applies to z/OS.



The COBOL Wrapper can be used with a call interface in IDMS/DC. This means you can build an application where the COBOL client program, every generated client interface object, the generic RPC services module and the broker stub are linked together, similar to the batch scenario. See *Using the COBOL Wrapper for Batch (z/OS, BS2000/OSD, z/VSE and IBM i)*.

### ➤ To use the COBOL Wrapper with a call interface within IDMS/DC

- 1.

Generate the client interface object(s) for the target operating system "z/OS", and use the interface type "IDMS/DC with standard calling convention". See *Generating COBOL Source Files from Software AG IDL Files*. Check the option "Generate the generic RPC service module COBSRVI". 

More info

2. If necessary, use FTP to transfer the client interface object(s) and, if required, also the generic RPC service module COBSRVI, to the target platform where you write your client application.
3. If you have generated the generic RPC service module and you plan to (re)install it, you may need to adapt the broker stub that supports the required transport (TCP, SSL, NET). See *Adapting the Used Broker Stub*.
4. Write your COBOL client program. See *Writing Applications with the COBOL Wrapper*, in particular the section *Using the RPC Communication Area with a Standard Call Interface*, and take into consideration the information given in *Software AG IDL to COBOL Mapping*.
5. Using the IDMS/DC translator for COBOL provided with your IDMS/DC installation and a COBOL compiler supported by the COBOL Wrapper, translate and compile:
  - the generated client interface object(s)
  - if required, the generic RPC service module COBSRVI
  - your COBOL client program

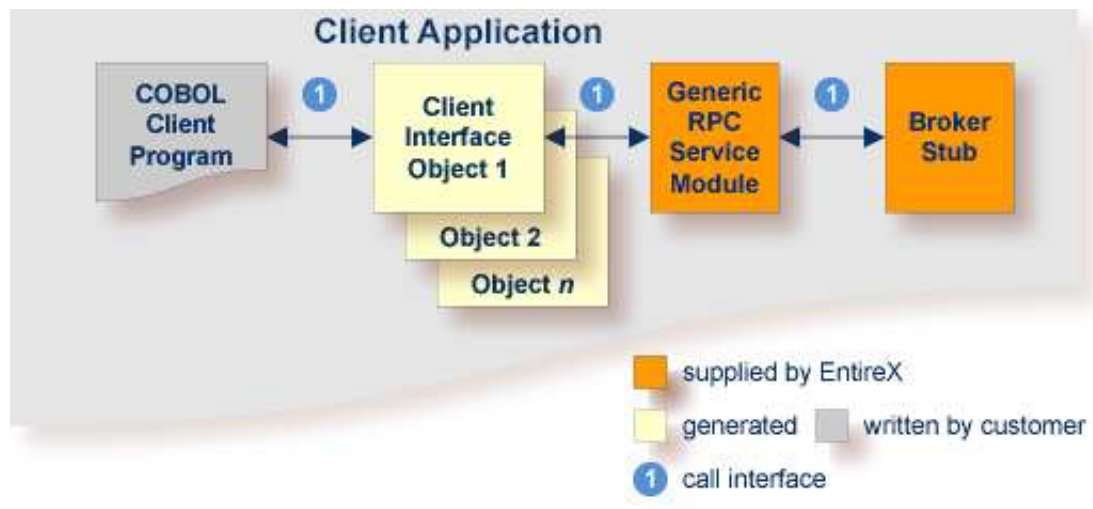
Take care the generated copybooks (see *Using the Generated Copybooks*) are accessed correctly by the compiler and not confused with the client interface objects, because the copybooks and client interface objects have identical file names. See your compiler documentation.

6. Using the standard linker (binder) of the target platform, link (bind) all translated and compiled modules, and, if required, the broker stub, together to a IDMS/DC program, using the standard linker (binder) of the target platform.
7. Install the IDMS/DC program within IDMS/DC.
8. Make sure the correct broker stub is used and can be called dynamically by the generic RPC service module COBSRVI.

See the broker installation documentation and use a broker stub for IDMS/DC (for example IDMSETB) from the common load library EXX970.LOAD. See also *Administering Broker Stubs*.

## Using the COBOL Wrapper for Micro Focus (UNIX and Windows)

This mode applies to UNIX and Windows.



In this scenario, the COBOL client program, every generated client interface object, generic RPC services module and the broker stub are linked together to the client application.

Use the COBOL Wrapper for Micro Focus if you need to embed the client interface object into your client application with a standard linkage calling convention.

### ➤ To use the COBOL Wrapper for Micro Focus

1. Generate the client interface object(s) for the target operating system, for example "Windows", and use interface type "Micro Focus with standard linkage calling convention". See *Generating COBOL Source Files from Software AG IDL Files*. Check the option "Generic the RPC service module COBSRVI". More info
2. If necessary, use FTP to transfer the client interface object(s) and, if required, also the generic RPC service module COBSRVI, to the target platform where you write your client application.
3. Import the modules into your Micro Focus IDE. The file names of the generated copybooks (see *Using the Generated Copybooks*) are derived from the IDL program name or its alias if present. The file names are the same as the file names of the client interface objects. They are distinguished by their extension, ".cbl" for the client interface objects and ".cpy" for the copybooks. If you import the generated copybooks and client interface objects into your Micro Focus development environment, take care the copybooks are accessed correctly by the compiler and not confused with the client interface objects. This may happen if you copy the generated copybooks and the client interface objects into one directory. See your Micro Focus documentation for more information.
4. Write your COBOL client program. See *Writing Applications with the COBOL Wrapper*, in particular the section *Using the RPC Communication Area with a Standard Call Interface*, and take into consideration the information given in *Software AG IDL to COBOL Mapping*.
5. Compile and link (bind) all modules together to an executable program:
  - the generated client interface object(s)
  - if required, the generic RPC service module COBSRVI

- your COBOL client program

For target operating system **UNIX** (i.e. the modules are generated for UNIX):

- The broker library from the EntireX UNIX installation must be linked to your client application, e.g. by defining the symbol "broker" as a linker option and linking the module *broker.o* from the EntireX UNIX installation.
- See your Micro Focus documentation for more information.

For target operating system **Windows** (i.e. the modules are generated for Windows):

- no additional compiler directives and linker options are required

6. Make sure the broker stub module can be called dynamically.

Under UNIX:

- The broker stub shared library or object *libbroker.so/sl* is accessible according to the rules of the UNIX system used, e.g. the directory of the library is defined in the `LD_LIBRARY_PATH` environment variable.

Under Windows:

- The broker stub DLL *broker.dll* is accessible, for example with the `PATH` environment variable.