ModCBroker: Administration Guide

DocumentId:GradSoft-CBroker-e-AG503-79-10.05.2000-3.3.0

Ruslan Shevchenko, Alexandr Yanovec, Julia Prokopenko

October 15, 2008

Contents

1 Introduction

`ModCBroker` – Apache module, for translation of HTTP requests to CORBA requests Cbroker is developed and supported by GradSoft company, source distribution `ModCBroker` is available on the web site http://www.gradsoft.kiev.ua. This manual is written for version 3.1.1 of software.

2 Installation and Compilation

2.1 For UNIX

Necessary software packages:

1. Operation System: Sun Solaris 2.8 or higher, Linux with kernel 2.4.0 or higher, FreeBSD 4.7 or higher. Porting to any posix-compatible platform is possible.
2. installed ORB - whether ORBacus 3.x/4.x or TAO or omniORB or Visibroker-4.5 or Orbix/E 2.0 or MICO
3. C++ compiler, compatible with ORB
4. installed Apache distribution with headers, version 2.0.44 or higher.
5. Gnu make 3.76 or higher

2.1.1 Installation of ModCbroker as DSO module

Preconditions:

- ORB must be in form of shared libraries set.
- Apache must be installed with enabled DSO support and header files.
- In case when ORB require using some specific compiler or linker setting, which are incompatible with default options of Apache compilation, you need to rebuild Apache with this specific options set in `EXTRA_CFLAGS` and `LDFLAGS_SHLIB` parameters.

  Partly, this situation arises when using FreeBSD with any multithreading object request brocker: you must recompile Apache with the following settings:

  ```
  EXTRA_CFLAGS=-pthread
  LD_SHLIB=gcc
  ```

  or simple set this flags in you environment before Apache build, if you install it from ports.

- Unzip and untar source distribution `mod_cbroker-<ver>.tar.gz`:

  - `gzip -d mod_cbroker-<ver>.tar.gz`
  - `tar -xvf mod_cbroker.tar`

- go to created directory `mod_cbroker`.
- start configure
  if necessary set configure options as written in chapter 2.1.2:
- `run gmake`
- `run gmake install` After this DSO module with name `mod_cbroker.so` will be installed into directory `<prefix>/libexec/apache2`
- add in Apache configuration file `httpd.conf` line

  ```
  LoadModule cbroker_module <prefix>/libexec/apache2/mod_cbroker.so
  ```

- restart Apache.

From now cbroker module is loaded at the web server startup, so you can begin testing procedures.

2.1.2 configure options

- path to installed apache header files (`--with-apache-headers=<path-to-apache-headers>`
  by default `configure` looks for apache headers in directories `/usr/include/apache`, `/usr/local/include/apache`, `/usr/local/apache/include`, `/usr/local/etc/apache/include`
- type of used ORB and path to ORB installation directory
  - `--with-tao=<tao-root>` for TAO
  - `--with-ob=<ob-prefix>` for ORBacus
  - `--with-omni=<omni-root>` for omniORB
  - `--with-visi=<visi-prefix>` for VisiBroker
  - `--with-mico=<mico-root>` for MICO

  `./configure` try to determine configuration of typical ORB installation, so in some cases with correct installed ORB in usual place you do not need to set this options.

You can view the list of all options of configure by typing

```
./configure --help
```

For details of ORB configuration procedure, please, read corbaconf [3]. Write us, is you've encountered any problems.

2.2 For Windows NT/XP

Necessary software packages:

1. ORB - ORBacus-4.0.x or 4.1.0
2. Installed Apache, ver. 2.0.44 or higher.
3. C++ compiler Microsoft Visual C++, ver. 6.0 or higher.

2.2.1 Installation process:

1. unzip archive `ModCBroker.zip`.
2. cd to the inside directory `ModCBroker/msvc.gen`
3. In `env_inc.mak` set values for the following variables:
   - PROJECT_ROOT - full path to the `ModCBroker`
   - TARGET_ORB - type of ORB you are using, i. e.
     - ORBACUS - for Orbacus (L=. http://www.ooc.com)
     - TAO - for TAO
     - OMNI - for OmniIDL
   - ORB_INCLUDES - set of paths to ORB includes (for example, for ORBACUS, installed in `c:\OOC` : `-Ic:\OOC\include`
   - ORB_LIBDIRS - set of paths to ORB libraries (in our case: `-Lc:\ooc\lib`)
   - ORB_LIBS - set of ORB libraries (in our case: `-lOB -lCosNaming`)
   - APACHE_DIR - directory, where installed Apache is situated.
   - APACHE_MODULE_DIR - Directory, where result mod_cbroker.so will be situated. Usually it is `modules` directory in Apache root directory.
   - INSTALL_DIR - main install directory.
   - INSTALL_IDL_DIR - directory, IDL to be installed.
   - INSTALL_INC_DIR - directory, header files to be installed.
   - INSTALL_LIB_DIR - directory, where libs to be installed.
   - MSVC_DIR - MS Visual C++ root directory.
4. cd to the `ModCBroker/msvc.gen`
5. Start compilation, by running `make.bat` or typing `make`
6. Start installation, by typing `make install`
7. (Install library) by typing `make install-dll`
8. Add to the Apache configuration file (httpd.conf) string

   ```
   LoadModule cbroker_module modules/mod_cbroker.so
   ```

   This directive must be declared after all other `LoadModule` directives.

2.3 Compiling of client library *clcbroker*

Client library is just a set of generated CORBA stub files, so you can create it on any computer and for any language by translating `idl/HTTP.idl` using your idl compiler. Then you can add generated files to you project, or assemble ones into library.

Client library for C++ is automatically generated during `ModCBroker` build process, so if client and server are situated on the same machine, you already have `clcbroker.a (or .sp)` .

Also, you can automatically create client library for C++, using source distribution of `ModCBroker` via next procedures.

2.3.1 For Unix

1. Unzip and untar source distribution of `ModCBroker`
2. Configure, by typing `./configure --without-apache`
3. Compile, by typing `make client`
4. Install, by typing `make install-client`

2.3.2 For NT/XP

1. Unzip distribution of `ModCBroker`
2. Compile, by typing `nmake -f Makefile.nt client`
3. Install, by typing `nmake -f Makefile.nt install-client`

3 Testing

1. After `ModCBroker` installation, assure: that Apache is working, and add to httpd.conf the next block:

   ```
   CbrokerLocation /cbroker
   CbrokerORBArgs -ORBInitRef NameService=corbaloc::your-host-name:10000/NameService
   ```

2. Restart httpd
3. Start CORBA NameService on port 10000. (for ORBACUS command is: `nameserv -OAport 10000`)
4. Cd to directory `mod-cbroker-dir/demo/HelloWorld`
5. Start HelloWorldServlet with set NameService by command:

```
./HelloWorldServlet -ORBInitRef \
          NameService=corbaloc::your-host-name:10000/NameService
For TAO 1.1
./HelloWorldServlet -ORBInitRef \
          NameService=iioploc://your-host-name:10000/NameService
```

6. Type in Web browser the next URL: http://your.host.name/cbroker/Hello/Hello

4 Using

1. Set in httpd cbroker config arguments of ORB, which correspond with location of your NameService.
2. Users programs, which work with `ModCBroker` must be linked with libclcbroker.so

5 Tuning

Tuning of `ModCBroker` consist in setting handler (by directory, host or file extension) and adjust of authorization, in addition you can set filters and scripting interfaces.

5.1 httpd.conf directives:

1. CbrokerORBArgs `<arguments for ORB>`
   - Set initialization parameters for ORB.
   - Default value: `-ORBconfig /etc/orb.cf`
2. CbrokerLocation
   - Set location for cbroker virtual directory.
   - Default value: /cbroker
3. CbrokerDefaultServlet
   - Set name of default servlet: we call this servlet when in request path element, which must be name of servlet, is empty.
   - Default value: `index`
4. CbrokerDefaultHandler
   - Set name of default handler: we call this handler when in request path element, which must be the name of handler, is empty.
   - Default value: `index`
5. CbrokerAddAuthByLocation location servletName handlerName

- Set authorization for directory `location`. Parameters:
  - location - directory, for each we set access.
  - servletName - name of servlet.
  - handkerName - name of handler (call of such handler must perform authorization)
- Default value: no

6. CbrokerPrefix
   - teg for begin of servlet call in CBROKER filter.
   - Default value: `<?cbroker`
7. CbrokerSuffix
   - teg for end of servlet call in CBROKER filter.
   - Default value: `?>`
8. CbrokerPrefix0
   - teg for begin of servlet call in CBROKER0 filter.
   - Default value: `<?cbroker0`
9. CbrokerSuffix0
   - teg for end of servlet call in CBROKER0 filter.
   - Default value: `?>`
10. CbrokerFormAuth on—off
    - Used inside directory, when we want to use form-based authorization mechanizm instead http authorization. In this case, when resource require authorize, mod_cbroker instead returning of HTTP_UNAUTHORIZED will redirect user to authorization form (custom or buildin). Then login and password, entered with this form, will be used during session (stored in crypted cookie)
    - Default value: off
11. CbrokerFormAuthCookieName
    - Name of cookie, where crypted login and password are stored.
    - Default value: cac
12. CbrokerFormAuthKey
    - Key for crypting authorization cookie (with blowfish algorithm).
13. CbrokerFormAuthLoginUrl
    - URL, on which authorization form must be shown.

- Default value: /cbroker-login

14. CbrokerFormAuthLogoutUrl
    - URL, which reset authorization cookies. (i. e. web application initiate redirect to this url for simulate exit from program).
    - Default value: /cbroker-logout
15. CbrokerFormAuthAfterLogoutUrl
    - URL for redirect from CbrokerFormAuthLogoutUrl
    - Default value: /
16. CbrokerFormAuthLoginFormTemplate
    - name of file, where authorization form is situated. If not set: used building standard form.
    - Default value: no
    - Form must set three parameters: login, password and backurl (url to redirect after successfull authentication). Also it is possible in form use expression $(LOGIN_URL) and $(BACK_URL) for `cbrokerFormAuthLoginUrl` and `backurl`.

5.2 Typical examples

5.2.1 Call of servlet for content output

```
CbrokerORBArgs -ORBInitRef NameService=corbaloc::www.name.com:10000/NameService
CbrokerLocation /cbroker
<Location /cbroker>
AuthType Basic
AuthName authorization
require  valid-user
</Location>
```

This example shows that:

1. apache call `ModCBroker`, in virtual directory cbroker (i. e. when request URL starts with `/cbroker`)
2. ORB is initialized by initialization string:

   ```
   -ORBInitRef NameService=corbaloc::www.name.com:10000/NameService
   ```

3. Servlets must be authorized. Note, that actually authorization is processed by users-s selvlet. Though it isn't mentioned in the configuration.

5.2.2 Note about ORB threading model

Before running `mod_cbroker` you must set threading model for ORB, which support callbacks. In some ORB (such as ORBacus) the default threading model does not support callbacks and your client will be hanged up at the first call of `HTTPStream` method.

So, for ORBacus, `CbrokerORBArgs` directive in our example of httpd.conf config must look as follows:

```
CbrokerORBArgs -ORBInitRef NameService=corbaloc::www.name.com:10000/NameService \
        -ORBreactive -OAreactive
```

For other ORB-s, where default threading model is not blocked:

```
CbrokerORBArgs -ORBInitRef NameService=corbaloc::www.name.com:10000/NameService \
```

5.2.3 Servlet usage for authorization

Let's suppose you need to define access to some resource (web application or just set of html pages) with help of `ModCbroker`. In this case you can use directive `CbrokerAddAuthByLocation`. Don't forget to make Apache calling authorization routines.

```
CbrokerAuthLocation  MyProtectedArea AuthServlet AuthHandler
<Location MyProtectedArea>
AuthType Basic
AuthName MyResources
require  valid-user
</Location>
```

Without `AuthType` setting Apache will generate internal error while access of `MyProtectedArea`.

5.2.4 Using of filters

ModCbroker also defines two filters: CBROKER and CBROKER0. You can set this filters to process content of some type with the command `AddOutputFilterByType`. For example:

```
AddOutputFilterByType  CBROKER  cbhtml
```

6 Integration with tcl [2] [1]

Also you can use `cbroker` with script language. In subdirectory `add_on/tcl` of distributive you can find tcl extension, which can be loaded from `websh`.

It must be compiled as Tcl extension in TEA standard and result library (libwebshcbroker.so for UNIX, libwebshcbroker.dll for Windows) should be loaded in section of interpreter initialization. After this it will be possible to call `cbroker` from websh.

7 Integration with mod_ssl

Since in Apache2 mod-ssl is already integrated as usual module, then you don't need any special settings.

8 Platforms matrix

OS	*ORB*	*Ported*	*Checked*	*FormalSupport*
$FreeBSD - 4.x$	$TAO - 1.3$	*Yes*	*Yes*	*Yes*
$FreeBSD - 4.x$	$OmniORB - 3.0.4$	*Yes*	*Yes*	*Yes*
$FreeBSD - 4.x$	$OmniORB - 4.0.0$	*Yes*	*Yes*	*Yes*
$FreeBSD - 4.x$	$ORBacus - 4.1.0$	*Yes*	*Yes*	*Yes*
Linux	$OmniORB - 3.0.4$	*Yes*	*Yes*	*Yes*
Linux	$MICO - 2.3.9$	*Yes*	*Yes*	*Yes*
Linux	$TAO - 1.2$	*Yes*	*No*	*Yes*
Linux	$Orbix/E - 2.0$	*Yes*	*Yes*	*No*
Linux	$Orbacus - 4.1.0$	*Yes*	*Yes*	*No*
Linux	$Visibroker - 4.5$	*Yes*	*No*	*No*
$Solaris - 2.8/Sparc$	$OmniORB - 3.0.4$	*Yes*	*No*	*Yes*

Meaning of columns:

- Ported means, that we such configuration was compiled and tested, but we provide official support and run regression tests not for all configurations;
- Checked means, that we run regression tests before release;
- Formal Support means, that we provide commercial support for this configuration. If you want to use ModCbroker commercially on a platform, which is checked, but not formally supported by us - contact us.

9 Changes

15.10.2008 added form-based authorizarion.

06.02.2003 reviewed for 3.1.1 version. Deleted obsolete support information.

06.02.2003 reviewed for 3.1.0 version.

21.05.2002 reviewed for 3.0.0 version.

18.03.2002 added platform matrix and contribution of Richard Bouska

15.02.2002 added note about using TAO-1.1

16.08.2001 added note about installing Apache on FreeBSD from ports collection.

21.06.2001 Changed order of installation for Windows NT.

02.06.2001 some restructuring.

27.05.2001 DSO module installation instructions added.

09.04.2001 Added section about integration with `mod_ssl`.

17.03.2001 review.

04.02.2001 Added note about ORB threading models; added formal document attributes.

06.09.2000 created.

References

[1] Apache Software Foundation NetCetera AG. *websh home page*, 2000-2002. http://tcl.apache.org/websh.

[2] J. K. Osterhout. Tcl: An embeddable command language. pages 133–146, 1990.

[3] Ruslan Shevchenko. *corbaconf: autoconf-based package for CORBA based applications*, 2000. http://corbaconf.kiev.ua.