

# HostProcControl : Administration Guide

DocumentId:GradSoft-AG503-e-79-02.09.2000-v1.0b

GradSoft <http://www.gradsoft.com.ua>

July 8, 2001

## Contents

<b>1</b>	<b>General information</b>	<b>2</b>
1.1	Overview . . . . .	2
1.1.1	Project mission . . . . .	2
1.1.2	Platforms . . . . .	2
1.1.3	Support . . . . .	2
1.2	Project components . . . . .	2
1.2.1	Supply . . . . .	2
1.2.2	Installation . . . . .	3
1.2.3	UFO . . . . .	3
<b>2</b>	<b>Installing procedure</b>	<b>4</b>
2.1	Necessary software . . . . .	4
2.2	Installing procedure for UNIX . . . . .	4
2.3	Installing procedure for Windows NT . . . . .	5
<b>3</b>	<b>Testing</b>	<b>5</b>
3.0.1	General information . . . . .	5
3.0.2	Test procedure . . . . .	5
<b>4</b>	<b>Usage</b>	<b>6</b>
4.1	Access permissions . . . . .	6
4.2	How to run the server . . . . .	7
4.2.1	General information . . . . .	7
4.2.2	Service options . . . . .	8
4.2.3	Two ways to run HostControlServer for Windows NT . . . . .	9
4.2.4	Model start procedure . . . . .	10
4.2.5	Output messages being able to appear when server starts . . . . .	11
4.3	Compiling client applications . . . . .	11
4.4	Running client applications . . . . .	11
4.5	Security risks . . . . .	11
<b>5</b>	<b>Remoiving package</b>	<b>12</b>

## 1 General information

### 1.1 Overview

#### 1.1.1 Project mission

HostProcControl is a CORBA service, which provides API for remote managing of an operating system. Using HostProcControl, you can create CORBA-objects able to do:

1. *regarding to the host file system:*
  - (a) reading and writing of files;
  - (b) deleting files;
  - (c) checking ones;
2. *regarding to the process control:*
  - (a) receiving of the list of processes;
  - (b) starting processes;
  - (c) killing ones.

#### 1.1.2 Platforms

You can use HostProcControl with several ORB realizations such as TAO, omniORB, and ORBacus for UNIX and with ORBacus only for Windows NT.

#### 1.1.3 Support

The package is created and supported by GradSoft company, the home page of GradSoft is <http://www.gradsoft.com.ua/>. The current release number of the project is **HostProcControl-1.0**.

### 1.2 Project components

#### 1.2.1 Supply

Supplied package consist of next of principle components:

1. Host control API in the form of IDL-module HostControl.idl
2. C++ source code of:
  - (a) Server **HostControlServer**;
  - (b) Accessories consisting of:

- i. **NtInstaller** for installing HostControlServer as Windows NT service;
  - ii. Samples of client applications;
- 3. Compilation management tools for Windows NT;
- 4. Compilation management tools for UNIX;
- 5. User guides:
  - (a) Programmer guide;
  - (b) Administrator guide (this file);

All these components are joint in archive named **HostProcControl-1.0** which may be **.zip** or **.tar.gz** depending on the choice.

### 1.2.2 Installation

Installed software consist of:

- 1. IDL module HostControl.idl;
- 2. Stub header file;
- 3. C++ static libraries:

platform	library name	contents
UNIX	libHostControlClient.a	stub
UNIX	libHostControl.a	stub, skeleton, servants
Windows NT	HostControlClient.lib	stub
Windows NT	HostControl.lib	stub, skeleton, servants

- 4. Executable files:
  - HostControlServer for UNIX
  - HostControlServer.exe and NtInstaller.exe for Windows NT

All these components are created from the source code at the time of installing procedure (except for the HostControl.idl) and then are copied to personal sub-directories of some directory of installation selected by user. Subdirectory names for coping HostControl.idl, .h files, C++ libraries and executable files are fixed to be **idl**, **include**, **lib**, and **bin** correspondingly. These subdirectories should be created by user itself *before* begining installation.

### 1.2.3 UFO

Next UFO :-> component are created automatically when server is installed as Windows NT service: the file **hpcservice.ini** in Windows directory. This file is used for unattended startup of HostControlServer.exe.

## 2 Installing procedure

### 2.1 Necessary software

1. CORBA ORB:
  - Unix: omniORB-3.0 or higher, *or* TAO-5.9b or higher, *or* ORBacus 4.0.2 or higher.
  - Windows NT: ORBacus 4.x or higher.
2. C++ compiler:
  - Unix: gcc-2.95.2 or higher, *or* SunProc C++ 4.2
  - Windows NT: Microsoft Visual C 6.0 or higher.
3. make:
  - Unix: gnu make
  - Windows NT: nmake from MSVC++
4. Special software: package GradC++ToolBox with components ProgOptions and ServiceOptions version 1.0.3 or higher

### 2.2 Installing procedure for UNIX

1. Make sure that necessary software is installed and run in the current environment.
2. Extract files from archive **HostProcControl-1.0** (being **.zip** or **.tar.gz**) to some directory, hereinafter referred to as a `<project_root>`.
3. Move out to `<project_root>`.
4. Run "configure" by `./configure` command;
  - you can set usual configure options previously;
  - you can get a list of options available by launching the `./configure` with `--help` option
5. Start compilation process by command `gmake`
6. Become superuser.
7. Run installation process by command `gmake install`
8. If you want to remove automatically generated files, type `gmake clean`

## 2.3 Installing procedure for Windows NT

1. Make sure that necessary software are installed and are in working state. Add pathes to "nmake" and "cl" programs into environment variable PATH.
2. Extract files from archive **HostProcControl-1.0** (being **.zip** or **.tar.gz**) to some directory, hereinafter referred to as a `<project_root>`.
3. Go to `<project_root>\config` directory and edit file `env_inc.nt.mak`. You have to set follow nmake values:

name	description
PROJECT_ROOT	<code>&lt;project_root&gt;</code>
INSTALL_DIR	Directory of installation
ORB_DIR	ORB root directory
MSVC_DIR	Microsoft Visual Studio root directory
GEN_DIR	GradC++ToolBox root directory

4. Go to `<project_root>`.
5. Start compilation process by command `make`
6. Run installation process by command `make install`
7. If you want to remove automatically generated files (such as `*.obj`, `*.lib`, `*.exe`) from `<project_root>` and its subdirectories, go to `<project_root>` and type `"make clean"`

## 3 Testing

### 3.0.1 General information

We provide 6 test/pattern applications disposed in 6 subdirectories 1,2,..,6 of `<project_root>/test` catalogue and named **step1,step6,..,step6** correspondingly. Each application's description is placed to the same sub-directory and named **readme<n>.txt**, where **n** is number of the test application.

### 3.0.2 Test procedure

1. Run compilation process according to previous chapter of the manual 2
2. Create test/pattern applications - move out to the `<project_root>/test` catalogue and call command `make`
3. Go to the `<project_root>/src` catalogue and run `HostControlServer` with following options:

```
--userlist <project_root>/test/HostControlUserlist.ini -OApport 16001
```

(these options meaning is described in chapter **Usage** 4 of the present manual);

4. Check tests as follows - perform with `<number>` from 1 to 6:
  - Move out to the `<project_root>/test/<number>` catalogue
  - Read test description in the `readme<number>.txt`
  - Run `step<number>` executable file with following option:  
`-ORBInitRef HostControlService=corbaloc::127.0.0.1:16001/HostControlService`
  - Make sure that `step<number>` application is functioning, as it was described in the `readme<number>.txt`
5. If you want to remove automatically generated test files, go to `<project_root>/test` and use `"make clean"`

## 4 Usage

### 4.1 Access permissions

A next outline of access permissions control is used:

- There is a list of granted users here. This is a file should be created *before* starting server, and may be corrected while the server run. This file consists of the formatted strings like follow:

```
John:Johnson
nsh:gsh
  anoter name : another password
```

and exact description of the strings is that:

```
[space]<first>[space]:[space]<second>[space]
```

Here `<first>` and `<second>` are substrings not framed by space characters; whole string is no lengthy than 255 characters; and the `<first>` substring does not contain `'` character.

Pair `<first>` and `<second>` of a single string is considered as some "client ID" where `<first>` is "user name" and `<second>` - "password".

To obtain access to the server functionality, client application must transmit to server its own "client ID" which is checked then in user list. Access to server is permitted if ID correspondent is found, and is not permitted otherwise.

*Note path to user list must be set ( - full name is strongly recommended! ) when server starts.*

## 4.2 How to run the server

### 4.2.1 General information

- What should you do before:
  1. Set the path to the user list;
  2. Adjust the way how the client applications will get an initial reference of the service;
  3. Set ORB parameters, if it necessary.
- How to do it:

You must define a set of options (see **Service options** key below 4.2.2) and pass them to the program by means of either a command line or a configuration file.
- About a command line:
  - There are two kinds of options may be present in command line:
    1. *service options* 4.2.2 to determine service features;
    2. *starting options* to determine details of server start procedure only.
  - Two kinds of options are not compatible: in view of essence of starting options available, using of service options in command line is unmeaning if any starting options present.
  - There are two starting options:
    1. `--config <filename>` to point service options are not in command line (they must be present in the file instead);
    2. `--help` to print of help message instead of service start.
- About configuration file:
  1. Configuration file is used if command line is empty or consists of starting option

```
--config <filename>
```

where `<filename>` is name of it.
  2. In the case of command line is empty, the name of configuration file assigns to be `HostControlConfig.ini`
  3. If the name of file is NOT full name (always in the case of command line is empty), this name is supposed to be relative to the "base". The "base" is setted to be Windows catalogue for Windows NT ( `C:\WINNT` is frequently occurring), and catalogue `/etc` for UNIX. To change this settings it is necessary to save new path to configuration file in system variables `HPC_CONFIG`

4. In any case configuration file must contain full set of options required to determine path to user list, order of IOR publication, parameters of ORB.
5. Format of configuration file is described in ProgOptions programming guide (<http://www.gradsoft.kiev.ua/eng/Products/ToolBox/ProgOptions/ProgGuide/Programming>). An example of configuration file is given below 3.

#### 4.2.2 Service options

1. Full name of user list:

The name of user list must be **NECESSARILY** defined when server starts (the name undefined or the file is not exist leads to starting break). To fix the name, the option

```
--userlist <filename>
```

is provided. If the name specified is **NOT** full name, it is supposed to be relative to the same "base" catalogue as the configuration file.

2. Initial object reference:

Publication of HostProcControl initial object reference is realized by means of module ServiceOptions derivable from package GradC++ToolBox. ServiceOptions provides facilities to use *corbaloc* style IOR to initial object and allows to publish IOR via next ways at user's option:

- (a) if `--with-naming` option used, then initial object of service is reflected in NamingService;
- (b) if `--ior-stdout` option used, then IOR is transformed to string and printed on standard output;
- (c) if `--ior-file-HostControlService <filename>` option used, then IOR is transformed to string and placed into file `<filename>`.

3. Parameters of ORB:

In order to client be able to use *corbaloc* IOR later, you must set at least one ORB parameter when server starts: you must fix the port to communications via common ORB option such as

- `-OApport` for ORBacus
- `-ORBport` for TAO
- `-ORBpoa_iiop_port` for omniORB

Another parameters of ORB may be fixed too, and common ORB option for that must be used. However keeping in mind necessary: An options syntax in our program is the same irrespective of way these options has been obtained via. That is the spelling of ORB options in configuration file must be the same as the spelling in the command line. If you want to

use syntax accepted for the ORB-like configuration file, you must create additional ORB-like configuration file and connect it via common ORB option `-ORBconfig <filename>`. For example, it is possible to use:

```
./HostControlServer --userlist HostControlUserlist.ini -ORBconfig orb.cf
```

where orb.cf provide following settings:

```
ooc.orb.server_timeout=2
ooc.orb.oa.port=1025
```

### 4.2.3 Two ways to run HostControlServer for Windows NT

There are two ways to run HostControlServer for Windows NT:

1. to run in capacity of console application
2. to run in capacity of Windows NT service

On the first way you may simply call `HostControlServer.exe` with some options described. To install program in capacity of Windows NT service (and run it) dedicated application `NtInstaller.exe` may be used. The usage of `NtInstaller.exe` is following:

1. Start `NtInstaller.exe` ( pop-up dialog box will be reflected )
2. To installing service fix next parameters:
  - (a) the name of NT service ("HostControl" is proposed);
  - (b) the path to `HostControlServer.exe` executable file;
  - (c) the command line to send into the program.

and press `{Install and Start}`. If all right, service will be installed and started with parameters present.

*Details:*

- (a) If the service has been already installed, it will be removed and installed anew.
  - (b) Autorun mode established. That is to say service being automatically runned when system starts.
3. To get an information about service installed press the button `{Service information}`. In that case service status and starting parameters for working service will be presented.
  4. To removing service press the button `{Uninstall}`.

#### 4.2.4 Model start procedure

Suppose the operation system is Windows NT and server must be started as a console application. The model start procedure is that:

1. To fix a base - to choose a <base> directory and save its name in system variable HPC\_CONFIG.
2. To create a user list in <base> catalogue (NB: it may be empty file being filled after start). Suppose the name of it is that: HostControlUserlist.ini
3. To create <base>\HostControlConfig.ini configuration file with following contents (see <http://www.gradsoft.kiev.ua/eng/Products/ToolBox/ProgOptions/ProgGuide/Program> for more details):

```
@"ProgOptions config file" // [first string] This stuff must create the file
```

```
# This is HostProcControl configuration file to use by executable:
```

```
HostControlServer.exe /* in the first significant word's capacity  
                        the name of program must be used */
```

```
# users list:
```

```
--userlist HostControlUserlist.ini /* the name of user list  
                                     relative to the base */
```

```
# options for ORB:
```

```
-OApport 16001 /* ORB option spelling is the same  
               as option spelling in the command line */
```

```
: this is EXAMPLE, but it may be copied and used directly.
```

4. Run HostControlServer.exe without options.

#### 4.2.5 Output messages being able to appear when server starts

	message	situation	actions connected
1	error of StartServiceCtrlDispatcher	for Windows NT only: error of starting NT service	program termination
2	error of parsing of <filename>	erroneous format of configuration file specified	program termination
3	filename necessary is NULL	argument of either "--config" or "--userlist" option is not assigned	program termination
4	file to read <filename> does not exist	full name of either configuration file or user list is erroneous	program termination
5	file to read <filename> does not exist in base directory <directory_name>	either configuration file or user list isn't found in base catalogue	program termination
6	CORBA System Exception	erroneour options of ORB	program termination
7	error while orb destroing	error of ORB destroing, be able when program terminating	no result to you
8	Server has been activated	normal start	starting server

### 4.3 Compiling client applications

Client applications must be linked with next HostProcControl libraries:

- For Unix:  
libHostControlClient.a
- For Windows NT:  
HostControlClient.lib

in addition to common ORB libraries including NamingService

### 4.4 Running client applications

Keep in mind options of server has been started!

### 4.5 Security risks

Note, that running HostProcControl allows client applications to get full access to server operation system.

So, to minimize security risks, please, follow next rules:

- Use this service only within a safe environment.
- When running HostProcControl-based system over unsecure network, use some technique of traffic crypting (VPN, or CORBA Security Service).

## 5 Remoiving package

1. For Windows NT only:
  - (a) use NtInstaller to remove HostControl NT service if it is installed;
  - (b) check file **hpcservice.ini** in Windows catalogue (mostly C:\WinNT) and remove it if it's exist.
2. Remove configuration files and user lists been created;
3. Remove the system variables **HPC\_CONFIG** if it is used;
4. Remove directory of installation or HostControl<smth> files from it; see chapter **Project components** 1.2.2 for the list of such files to be removed.
5. Remove directory in wich archive supplied has been opend.

## 6 History of the document

09.04.2001 current release; main changes relative to the previous release **HostProcControl-1.0b2**:

1. Chapter **General information** is extended;
2. New chapter **Remoiving package** is added.

24.02.2001 first public version.