Version: 4/12/2002 5:04 PM

# FARGOS/VISTA
Installation Guide

# FARGOS/VISTA Installation Guide

FARGOS Development, LLC
757 Delano Road
Yorktown Heights, NY 10598
http://www.fargos.net
mailto:support@fargos.net

## Notice of Rights

## Trademarks

## Abbreviations

FARGOS Development, LLC is a Limited Liability Company registered with the State of New York. It is required to identify itself as such in its name, hence the ", LLC" suffix. For purposes of readability in this document, the ", LLC" suffix is sometimes dropped. The phrase "FARGOS Development" always denotes "FARGOS Development, LLC" and is not intended to suggest any alternate form of organization.

## Notice of Liability

## Contents

# 1. Installing FARGOS/VISTA Distributions

FARGOS/VISTA provides a high-performance, transparently distributed, multithreaded, architecture-neutral object-oriented environment that supports a variety of hardware/operating system combinations. When possible, FARGOS Development, LLC makes use of the host platform's preferred software deployment technology. Since this varies from operating system to operating system, the installation procedure is unfortunately platform-specific.

While the particulars of installation vary, there is a common deployment structure followed on every platform supported by FARGOS/VISTA. The root of a FARGOS/VISTA distribution can be placed anywhere the system administrator chooses; the default location will vary depending on the target operating system. The layout of the distribution tree is illustrated in Figure 1. It is possible to place all of the files associated with every platform supported by FARGOS/VISTA on a single file server. In an enterprise environment, this can create a single place to apply updates but also creates a single point of failure.

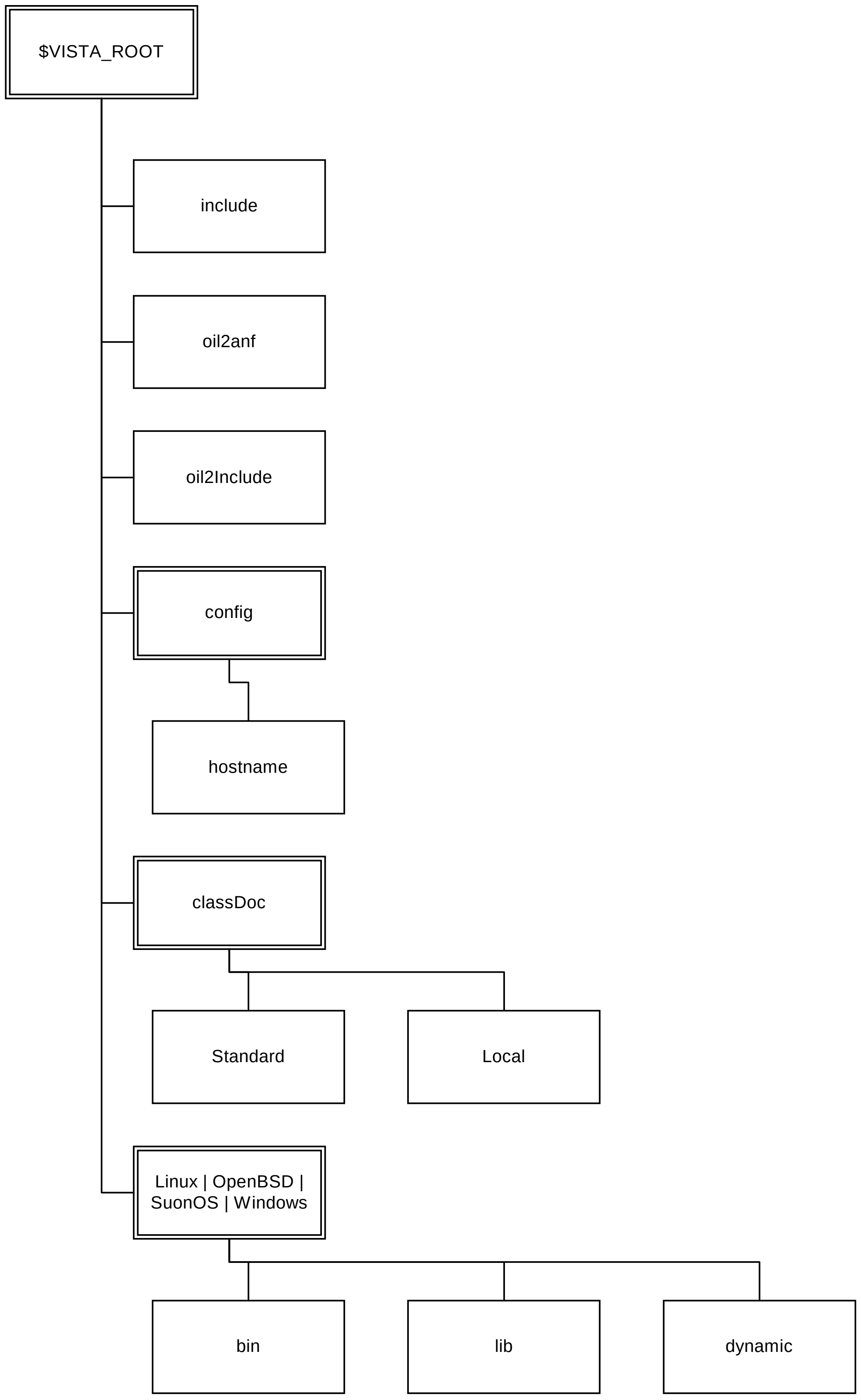
$VISTA_ROOT
include
oil2anf
oil2Include
config
hostname
classDoc
Standard
Local
Linux | OpenBSD | SuonOS | Windows
bin
lib
dynamic

**Figure 1**

## *Installation under Sun Solaris*

Sun Solaris uses the **pkgtrans** and **pkgadd** commands to convert and install a distribution package. Assuming that the FARGOS/VISTA Software Development Kit was downloaded as the file */tmp/vista.pkg*, the following two commands will install it in the default location, */opt/FRGSvista*:

```
    # pkgtrans /tmp/vista.pkg /var/spool/pkg
    # pkgadd FRGSvista
```

The package is prepared so that it is relocatable, thus it can be installed to an alternate location. Use the *–R* option of **pkgadd** to indicate the desired root of the directory tree. For example, the following places the distribution under the directory */usr/local/vista*:

```
    # pkgadd –R /usr/local/vista FRGSvista
```

The package can be removed using the **pkgrm** command:

```
    # pkgrm FRGSvista
```

## *Installation under Linux Variants*

Linux users can use the Red Hat Package Manager tool, **rpm**, to install and remove prepared software packages. While obviously available as part of Red Hat Linux distribution, the **rpm** utility is provided by most every Linux distribution (e.g., SuSE for S/390). RPM file names have a well-defined structure that embeds version, release and CPU architecture information. Assuming that version 2.5, release 1 of the FARGOS/VISTA Software Development Kit for Intel-compatible processors was downloaded as the file */tmp/vistasdk-2.5-1-i386.rpm*, the following command will install it in the default location:

```
    # rpm –i /tmp/vistasdk-2.5-1-i386.rpm
```

The actual RPM file name will be different as the “2.5-1” part of the file name will have been altered to correspond to the current version of the package. The FARGOS/VISTA Software Development Kit is created as a relocatable package, so the install location can be explicitly set using the **rpm** *prefix* option:

```
    # rpm –i –prefix=/home/vista /tmp/vistasdk-2.5-1-i386.rpm
```

A previously installed version can be removed using the **–e** option of the **rpm** command:

```
    # rpm –e vistasdk
```

## *Installation under OpenBSD*

Users of OpenBSD systems can use the **pkg_add** utility to install the FARGOS/VISTA Software Development Kit. Assuming that the package was downloaded as */tmp/vistasdk.tgz*, the following command will install it to the default location:

```
    # pkg_add /tmp/vistasdk.tgz
```

The package is relocatable, so it can be installed to an alternate location using the *–p* option of the **pkg_add** command:

```
    # pkg_add –p /usr/local/vista /tmp/vistasdk.tgz
```

OpenBSD packages are removed using the **pkg_delete** command:

```
   # pkg_delete vistasdk
```

### *Installation under Microsoft Windows*

Users of many Microsoft Windows variants (but not Microsoft Windows 3.1 or Microsoft Windows NT 3.51) can install the FARGOS/VISTA Software Development Kit by running the *VISTAsdk.exe* setup program. The setup program will prompt for a destination directory, which provides an opportunity to install the package anywhere. All files will be placed under the selected directory. No files are copied to the system directories. Once the software is installed, the setup program will set the *VISTA_ROOT* environment variable to point at the chosen installation directory and the *VISTA_UNAME* environment variable to the value of "*Windows*". Under Microsoft Windows NT-derived systems, the environment variables are maintained in the registry and are immediately available for newly spawned programs. Under Microsoft Windows 95-derived systems, the environment variables are placed into the *c:\autoexec.bat* file and the system will require a reboot for the new setting to take effect.

The software can be deleted by accessing the Control Panel (Start → Settings → Control Panel) and invoking "*Add/Remove Programs*". Select "*FARGOS/VISTA Software Development Kit*" from the list of removable software and click on the *Add/Remove* button.

#### Microsoft Windows NT Services

Several of the FARGOS/VISTA components can be used as long-running processes. A special utility, **OMEregNTserv.exe** in the *$VISTA_ROOT/Windows/bin* directory, can be used to register these programs as services. Once registered as a service, they can be managed using the relevant Microsoft Windows Service Configuration Manager. The **OMEregNTserv.exe** utility can also delete previously made registrations of such FARGOS/VISTA daemons. Its prototype is:

```
OMEregNTserv –add servName [options]
OMEregNTserv –delete servName
```

The service name parameter, *servName*, can be specified as a case-sensitive sub-string of the following known services:

- vista.exe
- OMEexecprog.exe
- OMEpersistd.exe

If no option is provided for the **vista** service, it defaults to using an *rc* file of *$VISTA_ROOT/config/service.rc*, where *$VISTA_ROOT* is evaluated at the time of the execution of the **OMEregNTserv.exe** program—if the environment variable's value is changed later, the path name will remain the same. Some examples:

```
OMEregNTserv –add vista
OMEregNTserv –delete vista
OMEregNTserv –add vista.exe d:\vista\config\localHTTP.rc
```

Once started, the processes will respond to status inquiry and stop requests.

# 2. Environment Variables

Several environment variables control the operation of FARGOS/VISTA utilities. The FARGOS/VISTA-specific variables are described in Table 2; variables that are defined by other organizations are described in Table 3 and Table 4. Of all the environment variables, two should be viewed as mandatory:

- VISTA_ROOT
- VISTA_UNAME

The *VISTA_UNAME* environment variable identifies the type of the underlying platform and indicates both operating system and machine architecture. It exists to permit a file server to export a single directory tree that make available multiple FARGOS/VISTA distributions, such as for Linux on Intel x86 and IBM S/390, Sun Solaris for SPARC and Intel x86, OpenBSD for Intel x86 and SPARC, etc. Only shell scripts normally need the *VISTA_UNAME* environment variable, which they would use to locate a platform-specific FARGOS/VISTA executable or library. Setting *VISTA_UNAME* is mandatory on Unix variants, but it can be overlooked under Windows variants if only MS-DOS batch scripts are used. To avoid potential problems in the future that would arise by the release of applications that depend on this environment variable, it is strongly recommended that the *VISTA_UNAME* environment variable be set on all platforms. Some currently used values:

**Table 1**

| Hardware Platform | Operating System | VISTA_UNAME | Notes |
|---|---|---|---|
| Intel 386 | Red Hat Linux | Linux_i386 | Generic x86-based release |
| Intel Pentium III | Red Hat Linux | Linux_i686 | Optimized for Pentium II and above |
| IBM System/390 | SuSE Linux | Linux_s390 | |
| Intel Pentium III | Sun Solaris | SunOS_i386 | Generic x86-based release |
| Intel Pentium III | OpenBSD | OpenBSD_i386 | Generic x86-based release |
| Intel Pentium III | Microsoft Windows | Windows | Generic 32-bit Windows |
| Sun UltraSPARC | Sun Solaris | SunOS_sparc | 32-bit release |
| Sun UltraSPARC | Sun Solaris | SunOS_sparcv9 | 64-bit release |

Note: there is no simple, universal, programmatic mechanism to determine the correct and optimal value of *VISTA_UNAME*. The closest for Unix variants would be:

```
    VISTA_UNAME=`uname –s`_`uname –m`
```

Because no FARGOS/VISTA-related application internally derives a value for *VISTA_UNAME*, site administrators can configure and deploy their own naming conventions. Such site-specific conventions are useful for parallel deployment of production, developer and test configurations.

The root of the FARGOS/VISTA distribution tree should be stored in the environment variable *VISTA_ROOT*. How this environment variable gets set is operating system-specific and users may be required to set the environment variable in their login scripts.

Consider that the default for *VISTA_ROOT* under Solaris is */opt/FRGSvista*. For a user of the Bourne-equivalent shells, (e.g., **sh**, **bash**, or **ksh**), one would add to *$HOME/.profile*:

```
    VISTA_ROOT=/opt/FRGSvista; export VISTA_ROOT
```

For **csh** users, one would add to *$HOME/.login*:

```
    setenv VISTA_ROOT /opt/FRGSvista
```

While the **VISTAsdk.exe** setup program for Microsoft Windows sets the *VISTA_ROOT* environment variable as part of the installation process, a Microsoft Windows user might have occasion to manually configure his current session so as to point at a test release or a CD-ROM distribution. This can be done by altering the default environment (under a Windows NT-derived system) or issuing an MS-DOS SET command in a command shell:

```
    SET VISTA_ROOT=C:\Progra~1\VISTA
```

**Note:** unlike most operating systems, Microsoft Windows file names are permitted to have embedded spaces. Indeed, many of the standard directories defined by Microsoft are comprised of two words separated by a space. Unfortunately, the vast majority of programs use white space to separate command-line parameters. Thus, a directory like "*Program Files*" would almost invariably be parsed as two parameters ("*Program*" and "*Files*"). For this reason, the **VISTAsdk.exe** setup program computes the short form (8.3) of the root directory to ensure that no spaces appear in the path specification stored in *VISTA_ROOT*.

**Table 2**

| Variable | Description |
|---|---|
| VISTA_ROOT | Specifies the directory that is the root of FARGOS/VISTA distribution. For most installations, this will be the only variable that needs to be set as the default settings are based upon this variable and normally yield the desired result. |
| VISTA_UNAME | Specifies the underlying operating system and machine type. For Unix systems, often the value should be set using what is returned by the concatenation of the output from the "**uname** –s" and "**uname** –m" commands, separated by an underscore:<br>VISTA_UNAME=\`uname –s\`_\`uname –m\`<br>For 32-bit Microsoft Windows systems, the value should be set to "*Windows*". No distinction is currently made between the variants of 32-bit Windows, such as Windows 95 vs. Windows 2000. |
| VISTA_LICENSE_PATH | A colon- (or semi-colon-) separated path of directory names that are searched for license files before, and in addition to, the standard locations. The standard locations that are always searched are: the current working directory, *$HOME/.vista/<u>hostname</u>*, *$HOME/_vista/<u>hostname</u>*, *$HOME/.vista*, *$HOME/_vista*, *$VISTA_ROOT/config/<u>hostname</u>*, *$VISTA_ROOT/config*. |
| OIL2_DOCUMENT_ROOT | Specifies the directory under which class documentation generated by the OIL2 compiler will be placed. This is normally not defined and defaults to *$VISTA_ROOT*. |
| OIL2_INCLUDE_PATH | A colon- (or semi-colon-) separated path of directory names that the OIL2 compiler will first examine when it searches for include files. The OIL2 compiler will always search the current working directory and the directory *$VISTA_ROOT/oil2Include*. |

The behavior of some FARGOS/VISTA components is influenced by the settings of environment variables that are defined by other standards bodies. In general, such variables affect the process of locating a particular configuration file.

## User-Specific Configuration Files

While not often exploited, when a search is performed for a configuration file, a per-user location has precedence over system-wide defaults. By convention, such per-user configuration files are stored under the "*.vista*" subdirectory of the user's home directory. The user's home directory is defined using the environment variables described in Table 3. If a user's home directory is maintained by a file server, host-specific files can be stored in a subdirectory of *.vista* that corresponds to the host's name.

**Table 3**

| Variable | Description |
|---|---|
| HOME | Specifies the full, absolute path of the user's home directory. |
| HOMEDRIVE | If *HOME* is not set[1], on Microsoft Windows variants this specifies the drive letter prefix (e.g., "*D:*") of the user's home directory |
| HOMEPATH | If *HOME* is not set, on Microsoft Windows variants this specifies the absolute path name of a user's home directory within a drive (e.g., "*/users/default*"). |

## Native Language Message Configuration

FARGOS/VISTA's Native Language Message support is a superset of several standards (e.g., X/Open XPG3/4, Uniforum) related to the internationalization and localization of application programs. The underlying Native Language Message library recognizes a large number of environment variables, many of which perform identical roles but are accepted in order to comply with various historical schemes and incompatible standards. These are listed below in Table 4.

[1] The use of *HOME* is always preferred; it should be set using a command equivalent to: `set HOME=%HOMEDRIVE%%HOMEPATH%`.

**Table 4**

| **Variable** | **Description** |
|---|---|
| 1. LC_ALL<br>2. LC_MESSAGES<br>3. LANG | Specifies the requested locale. These variables are listed in order of priority, thus the setting of *LC_ALL* overrides the setting of *LANG*. |
| NLSPATH<br>VISTA_NLSPATH[2] | Specifies a set of directories that should be searched for Native Language Message catalogs. |
| TEXTDOMAIN | Not recommended for use. Specifies the default catalog name. |
| TEXTDOMAINDIR | Not recommended for use (use *NLSPATH* instead). Specifies a single directory to search for Native Language Message catalogs |

The value of *NLSPATH* is defined to be a colon-separated (":") list of directories; however, FARGOS/VISTA applications also support semicolon-separated (";") path names, which are often used on Microsoft Windows-based systems due to the inclusion of drive letters, such as "*C:*".

**Note:** the FARGOS/VISTA runtime correctly recognizes colon-separated path names that make use of path specifications that include drive letters. The use of semicolon-separated path elements is thus a matter of taste, not mandatory.

The directory names in an *NLSPATH* or *VISTA_NLSPATH* can make use of the following meta-characters, which are substituted with appropriate values at runtime:

| **Meta Character** | **Value** |
|---|---|
| %% | % (escaped percent sign) |
| %L | Locale specification |
| %l | Language element of locale specification |
| %t | Territory element of locale specification |
| %c | Codeset element of locale specification |
| %N | Catalog (application) name |
| %F[3] | FARGOS/VISTA NLM Catalog name, equivalent to "%N-%L.nlmcat". |

When searching for a Native Language Message Catalog, the following directories are examined in the order illustrated below:

1. Those specified by the expansion of *VISTA_NLSPATH*

[2] *VISTA_NLSPATH* is a FARGOS/VISTA-specific variable that has precedence over the standard variable *NLSPATH*.
[3] This is a FARGOS/VISTA extension, which is not defined by any standard. It is perfectly safe when used with VISTA_NLSPATH, but it may not be accepted by other applications if it is used in NLSPATH.

2. Those specified by the expansion of *NLSPATH*
3. "." (the current directory)
4. $*HOME*/.vista/NLMcatalogs
5. %*HOMEDRIVE*%%*HOMEPATH*%/.vista/NLMcatalogs (if *HOME* was not set)
6. $*VISTA_ROOT*/NLMcatalogs

Within a directory, three distinct files are sought. In order, they are:

1. "%N-%L.nlmcat" (the application's locale-specific NLM catalog)
2. "%N-C.nlmcat" (the application's default NLM catalog, which is in the "C" locale)
3. "%N" (the application's name is treated as the file name, which permits the NLM system to also be used for supporting end-user customizable configuration files)

# 3. Localized Configuration Files

There are two classes of localized data files that are used by FARGOS/VISTA-based applications:

- license files provided by FARGOS Development, LLC
- localized configuration data created by a site's administrator.

A FARGOS/VISTA license file will usually come with instructions as to where it should be stored. Generic issues related to localized data files are discussed below.

Administrators may need to prepare and distribute user/host/site-specific configuration files. These localized data files normally are named with the suffix "*.vld*" (refer the section entitled "Magic Numbers" for specification of the file format's magic number).

The FARGOS/VISTA localization scheme is quite flexible. A file can be restricted to a particular user on a specific host; it can be restricted to a particular user on any host within a given domain; it can be restricted to any user on a particular host or any user within a particular domain.

## FARGOS/VISTA Domain Keys

One of the more common uses of localized data files involves the exchange of FARGOS/VISTA domain key files, which are used to enable the exchange of data between two FARGOS/VISTA Object Management Environments or associated applications. The key files perform two purposes:

1. They provide authentication data for a site.
2. They help restrict attempted communication by unauthorized third parties.

When two FARGOS/VISTA Object Management Environments establish a communications link, information is exchanged that identifies each end of the link and negotiates a variety of parameters, including the session key used to encrypt all data that is transmitted. The following is a high-level explanation of the process.

All data sent from one end of the link is encrypted using a randomly generated session key. The encryption is further skewed by an initialization vector, which must also be known by each side of the link. Thus, successful decryption of communication from one FARGOS/VISTA Object Management Environment process to another requires knowledge of both the randomly generated session key and the previously agreed-to initialization vector.

Because the session key is randomly generated, it cannot be made available to the remote end of a link through some static means, such as the distribution of a file. Instead, the value of the session key is transmitted using public key encryption: the local side of the link encrypts the randomly generated session key using its private key and the remote end of the link decrypts the session key using the corresponding public key.

Successful decryption of the session key using the relevant public key proves to the remote site that it is truly in communication with the local site. The key issue (pun intended) is how each end of a link determines its respective private key/initialization vector) and the public key/initialization vector of the other end. All FARGOS/VISTA Object Management Environments support a predefined "*public*" domain: a set of embedded public/key pairs and a corresponding initialization vector are embedded in the runtime object code. Use of the *public* domain provides no verifiable authentication (because every FARGOS/VISTA Object Management Environment contains the keys), so it is primarily useful for anonymous interactions.

Trust between two peers is enabled using authenticated communication links that are associated with an administratively defined logical domain. The necessary static information is maintained in localized data files that are stored on each the peer hosts. Such files are created by the **OMEmkKeyFiles** utility[4], which generates a pair of private and public key files. The private key file contains a randomly generated private key and a randomly generated initialization vector; the public key file contains the corresponding public key and a copy of the initialization vector.

The private key file is localized so that it can be accessed on a particular host and it is encrypted using an administrator-assigned secret phrase. This secret phrase must be used to access the encrypted data. The private key file itself should be also treated as a secret, which means taking appropriate precautions to keep it from being copied by unauthorized users. Possession of the private key file is not sufficient to impersonate a host, since the knowledge of the secret phrase is required to decrypt it. Nor is knowledge of the secret phrase sufficient to impersonate a host: the actual public/private key pairs and initialization vectors are randomly generated and have no correlation with the secret phrase. It thus takes possession of both the private key file and the secret phrase to enable impersonation of a host.

The public key file must be shared with peer sites with which one wants to interact. If the remote site does not have the local site's public key file, communication will not be possible because the remote site will be unable to decrypt any data that is sent. Public key files are themselves localized so that they can only be decrypted on hosts within a particular domain; however, a hostile third party could configure their own equipment to claim to be a member of said domain. It is important to remember that public files are, by definition, not secret. Restricting their distribution helps prevent unauthorized communication attempts, but the security of the local system does not depend on a remote site retaining the secrecy of a public key file.

# 4. Optional Configuration

This section provides additional information that is not required by any FARGOS/VISTA installation.

[4] The **OMEmkKeyFiles** utility is only present in commercial distributions and its export to certain jurisdictions outside the USA is restricted.

## *Magic Numbers*

The implementation of the **file** command found on many Unix variants can be modified by adding appropriate entries to a local "magic" number file, which is normally */etc/magic*. The following entries can be added to recognize FARGOS/VISTA-related file formats.

```
# FARGOS/VISTA files
0  string      \005\025\013      FARGOS/VISTA License
>3 byte        x                 version %d
>24      belong      1                 using standard encoding
>24      belong      2                 using compressed encoding
0  string      \005\025\053      FARGOS/VISTA Localized Data File
>3 byte        x                 version %d
>24      belong      1                 using standard encoding
>24      belong      2                 using compressed encoding
0  string      o2o               OIL2 Architecture Neutral Format
>3 byte        x                 version %d
```

## *Makefile Rules*

Authors of applications written in OIL2 may find the following **make** rules useful in their Makefiles.

```
# Copyright (C) 1999-2001 FARGOS Development, LLC.  All rights reserved.

# *** NOTE *** External variables used
# OBJ_SUFFIX - typically .o or .obj depending on platform
# LIB_SUFFIX - typically .a or .lib depending on platform
# LIB_PREFIX - typically lib or null depending on platform
# EXE_SUFFIX - typically not defined (or null) or .exe depending on platform

.PRECIOUS: .cpp
.SUFFIXES: .so .dll .obj .o2o

# C++ compilation rules -> native object
%.${OBJ_SUFFIX} : %.cpp ; ${CC_PLUSCOMP} ${OPTIMIZE} -D_REENTRANT -
I${VISTA_ROOT}/include ${CPLUSFLAGS} -c $<
# end C++ rules
# OIL2 -> C++
%.cpp : %.oil ; ${VISTA_ROOT}/${VISTA_UNAME}/oil2 $<
# OIL2 -> OIL2 Architecture Neutral Format object code
%.o2o : %.oil ; ${VISTA_ROOT}/${VISTA_UNAME}/oil2_parse -oil2 $<
```

## 5. Further Reading

Further details about FARGOS/VISTA can be found in the following resources:

*FARGOS/VISTA Overview*

*FARGOS/VISTA Object Management Environment Programmer's Reference*

FARGOS/VISTA Object Management Environment Classes

*Object Implementation Language 2 Reference*

*An Introduction to Programming using OIL2*

*FARGOS/VISTA HTTP Server Programmer's Guide*

*FARGOS/VISTA Examples*

*FARGOS/SolidState HTTP Server Adapter*