OVERVIEW

OPERATION

ADMINISTRATION

HARDWARE

DEVELOPMENT

QUALIFICATION

SyCOS System Configuration Definition

sycos
PROCESS CONTROL SYSTEM

Version: 1.1
Date: November 2002
Author: Wolf Geldmacher/R. Breitenstein

1. Table Of Contents

2. Introduction

2.1. About This Manual

This manual describes the system configuration file and its interaction with the various parts of SyCOS.

2.2. Who This Manual Is For

This manual is intended primarily for the system engineer implementing a SyCOS environment and for the installer/administrator configuring SyCOS for use in a particular plant.

2.3. To What Does This Manual Apply

This manual is applicable to SyCOS version 9.2.0

2.4. How This Manual Is Organized

This manual is organized in three major sections: The first section describes the functionality of the system configuration file and its interaction with the other parts of SyCOS. The second section describes the syntax of the configuration file and explains the different constructs used in the file. The third section introduces the tools (cfgcc and cfg) used to compile the configuration file and to extract information from a compiled configuration file.

2.5. Request for Comments

If you have any problems with this text contained in this manual or you wish to make suggestions for improvements then contact the SyCOS Product Support Group, namely:

Mr. Rolf Breitenstein
Syte GmbH
Seewenweg 5
CH-4153 Reinach, Switzerland

Please provide the full name of the document and its version number.

2.6. Acknowledgments

This manual was written by Wolf Geldmacher/R. Breitenstein

3. Introduction

3.1. Configuration Information

Within SyCOS a lot of configuration information is needed. The configuration information includes answers to the following questions:

- which machines are on the network?
- which IP address is assigned to which machine?
- which machines serve as PIC’s? as FS’s? as MFC’s?
- who is the time server?
- where does alarm information go?
- which server does a given MFC boot from?
- which UNIX device file is associated with a given SyCOS typer?
- which machines have modems?
- where are the machines located in the building?
- what is the authentication strategy used for the net?
- what is the authentication strategy used for a given console?
- which users have access rights to SyCOS components

and many, many more like these.

Configuration information is therefore needed for

- hardware configuration
- software configuration

Configuration information may be further subdivided in information that changes slowly or not at all during the lifetime of SyCOS and information that changes dynamically in response to project requirements or even while a process is being controlled.

3.2. Goal of the configuration file

The configuration file was introduced to collect the more static (as opposed to process dependent) information in a single, central place for ease of maintenance and consistency.

It should be very hard (if not impossible) to create inconsistent or incorrect configuration information for a SyCOS network if all configuration data is kept in a central location (although partial configuration information may be derived from this central location if it can be used to avoid manual actions).

The configuration information must be accessible by all machines in a SyCOS network, regardless of machine architecture and location.

It should be possible to generate all kinds of derived information (e.g. UNIX device entries or entries in the UNIX password file) from the central configuration file and to update the actual configuration accordingly (e.g.: remove a no longer needed device and/or create a new one).

The language used to describe the system configuration should be declarative and unambiguous.

It should be possible to configure a network completely by just designing the correct configuration file and calling a small set of utilities to implement the configuration.

3.3. Design of the configuration environment

In order to achieve the goals stated above the following approach has been chosen:

Configuration information is entered in a human readable from in a text file.

The file is then compiled into a machine independent internal form (XDR) by a configuration file compiler. This allows for consistency and completeness checks and assures that the configuration can be reconstructed at any time from the source. (The direct use of the text file was regarded as too unsafe, because changes in the source could lead to temporary inconsistencies. Also requiring each and every application to parse a potentially errornous configuration file would be too time consuming.)

The XDR representation of the configuration information can be accessed by applications through a set of library routines and macros using NFS as transport to make the information available to all machines on the net.

A utility is provided for shell level access to the configuration information. This utility is used to implement shell procedures that will create/modify/update/remove secondary configuration information on PIC's and FS's in the net.

3.4. State of the configuration file effort

The configuration file development effort is still under way. Some, but not all SyCOS parts retrieve information from the central configuration database. The MFC side use of configuration information is almost complete, the FS/PIC side still lacks major components. The latter is due to the fact that almost all existing programs must be modified in order to use the configuration information and a set of utilities for system configuration must still be developed.

In the rest of this document notes will draw your attention to the areas that still need implementation. You will find that there are quite a few constructs already available within the configuration file that are not yet used by any part of SyCOS.

When you write system configuration files please try to provide all of the information, even though it may currently be unused. This will allow for a smoother migration while more and more of the configuration information is actually taken from the system configuration file.

4. The Configuration File

4.1. Tokens

The language describing the system configuration has the following basic tokens:

`Identifier`

An identifier is a name for some configuration item. It must start with a letter and may be followed by any number of letters, digits or the character '_'. Although the length of an identifier is not limited by the language itself there may be restrictions on the significant length of an identifier depending on the object designated with the name.

Upper and lower case letters are distinguished: The identifiers '`printer01`' and '`Printer01`' designate two different objects.

Examples: `Printer01 sycosnet AHostName`

Constant

A (numeric) constant is a sequence of digits. If the constant starts with one of the digits 1 through 9 it is taken to be a decimal integer constant. If it starts with the digit 0 it is taken to be an octal (base 8) constant and only the digits in the range 1 through 7 may follow. If the constant starts with the sequence '0x' it is assumed to be a sedecimal (base 16) constant. Valid digits for sedecimal constants are the digits 0 to 9 and the letters a or A through f or F for the values 10 to 16 respectively.

Examples: `32 040 0x20` (all of these have the same numeric value!)

String

A string is an arbitrary sequence of characters delimited by double quotes “). Strings cannot be continued over the end of a line and the double quote used to delimit the string must be preceded by a \ character if it is to be a part of the string.

Examples:
```
“This is a string...”
“.. and this string contains a double quote: \””
```

Ipaddr

An internet protocol address. This token consists of 4 decimal constants in the range from 0 to 255 separated by periods.

Examples: `192.1.2.3 255.255.255.0`

Keywords

The following Identifiers have a special meaning in the context of the configuration file and must be specified as given:

`ADI ALARMDESTINATIONS AUTHENTICATION AUTHORIZATION BACKUP BHF BLOCK BOOT BY CAPACITY COSAPPL COSDEVS COSLOGS COSROOT CLIENT COMMENT CONFIGURATION CONSOLE COVER CPU CYCLE DEVICE DEVICENAME DIALIN DIALOUT DISTRIBUTION END ERROR FALSE FILE FILESYSTEM FS FUNCTION GID GLOBAL HARDQUOTA HOME HOST IMPLEMENTED INTERFACE IP LINKCHECK LOCAL LOCATION LOGFILE MAJOR MANUFACTURER MASTER MEMSIZE MFC MINOR MODEM NETWORK NFSAUTH NFSMOUNT NONE PAGER PASSWD PATH PERIOD PIC POS PRINTER PULSWIDTH QUOTA RAW REMOTE SERVER SESSION SHELL SLAVE SOFTQUOTA STORAGE TIME TIMEOUT TREND TRUE TYPE UID UNIQUE USER WATCHDOG`

Delimiters

Tokens are delimited by white space characters (ASCII characters SPACE, TAB, FORMFEED or NEWLINE)

Comments

Comments are introduced by the character **`#`** and continue to the end of the line.

Style

The configuration file is a free format (i.e. not line oriented) text file. You can use arbitrary white space between tokens to improve legibility.

4.2. Syntax and Semantic

This chapter describes the syntax of the configuration file using the tokens (in bold face) as defined in the preceding chapter in BNF. For each construct its meaning is given, potentially enhanced by some notes.

network

```
: NETWORK ':' Identifier net_parts END NETWORK
```

The configuration files names a network and describes the objects that are part of the network as well as properties of the network in general.

net_parts

```
: net_part
| net_parts net_part
```

The network description consists of a list of parts that describe the object and properties of the network. The sequence of the parts is free: `net_parts` may occur in any order.

net_part

```
: location
| manufacturer
| COSROOT ':' String
| COSAPPL ':' String
| COSLOGS ':' String
| COSDEVS ':' String
| PIC ':' Identifier ws_parts END PIC
| FS ':' Identifier ws_parts END FS
| MFC ':' Identifier mfc_parts END MFC
| USER ':' Identifier user_parts END USER
| AUTHENTICATION ':' netauth_list END AUTHENTICATION
| ERROR DEVICE ':' BHF
| ERROR DEVICE ':' Identifier
```

Part of a network description is either the definition of the location of the network (this clause may only appear once for every network), or the definition of the network's provider name (may also appear once only), or the definition of the values for environment variables (**COSROOT**, **COSAPPL**, **COSLOGS**, **COSDEVS**) used throughout the network (only one definition per variable per network), or the description of machines (**PIC**'s, **FS**'s, **MFC**'s) in the network (multiple machines of all types are allowed). A machine description is introduced by a keyword that specifies the type of the machine followed by the machine's host name and the SyCOS relevant parts of the machine.

Other parts of the network description specify the users known in the network by name (multiple users may be defined), define a global authentication strategy for the network (may only occur once and may be overridden for specific consoles) and define one or more system error devices.

The normal sequence of the network parts frequently follows the above order, although it is only necessary to define network elements before they are used (e.g. An **ERROR DEVICE** clause can reference an ADI **Identifier** only after it has been defined in the context of a **PIC** clause).

ws_parts

```
: ws_part
| ws_parts ws_part
```

Both File Servers (FS's) and Process Information Centres (PIC's) are UNIX machines and have many properties in common. A (non-empty) list of properties must be specified for all workstations in a network. The sequence of parts specified is again irrelevant, as long as items are defined before they are used.

ws_part

```
: location
```

```
| manufacturer
| CPU ':' String
| MEMSIZE ':' Constant
| IP ':' Ipaddr
| CONSOLE ':' Identifier console_parts END CONSOLE
| MODEM ':' Identifier modem_parts END MODEM
| PAGER ':' Identifier pager_parts END PAGER
| PRINTER ':' Identifier printer_parts END PRINTER
| STORAGE ':' Identifier mass_parts END STORAGE
| BACKUP ':' Identifier IMPLEMENTED BY Identifier END BACKUP
| ALARMDESTINATIONS ':' adi_list END ALARMDESTINATIONS
| TREND ':' Identifier trend_list END TREND
| SERVER servertype ':' serverclass
```

Every workstation (PIC or FS) can have exactly one location and manufacturer clause.

The **CPU**, memory size and IP address of the machine must be specified exactly once. The string specified for the CPU type must be one defined in a table in the next section ("Host CPUs" on page 17).

The **CPU** information is currently not used by any part of SyCOS but may be used in future versions.

The memory size (**MEMSIZE**) is specified in Megabytes. This information is currently not used by any part of SyCOS.

All **IP** addresses are checked for uniqueness and specification of a duplicate IP address results in an error but are currently not checked for other criteria (e.g. it is possible to define an IP address that is in a different IP network and therefore not reachable).

The **CONSOLE** clause defines a console connected to the machine for SyCOS use. Note that this clause may only appear up to 4 times per FS/PIC description, corresponding to the up to 4 operator consoles connectable to a workstation. It is good practice to only have **CONSOLE** clauses for PIC's (and none for FS's).

The **MODEM** clause names a modem connected to a workstation. This clause is currently not used by any part of SyCOS.

The **PAGER** clause names pager devices connected to a workstation. It is good practice to only have **PAGER** clauses for PIC's, although the configuration compiler does not enforce this (yet).

The **PRINTER** clause defines printers connected to a PIC or FS. If a printer is used as an alarm printer in SyCOS context it is good practice to only define such a printer within a PIC description. Except as targets for **ALARMDESTINATION** clauses the **PRINTER** clause is currently ignored by SyCOS.

The **STORAGE** clause defines mass storage devices (disks, tapes) connected to a PIC or FS.

The **BACKUP** clause designates a STORAGE device for use as system backup device.

These two clauses currently only serve documentation purposes and are ignored by SyCOS.

The **ALARMDESTINATION** clause defines alarm destinations and maps them onto previously defined devices.

The **TREND** clause defines location and behaviour of ad-hoc trend files.

The **SERVER** clause is used to designate the machine as time, distribution or authentication server. This information is currently evaluated by the MFC's in a SyCOS network only.

console_parts

```
: console_part
```

```
| console_parts console_part
```

Any workstation (although, as stated above, this should be limited to PIC type workstations) can have up to four operator consoles attached to it. The construct specifies the properties of an operator or engineering console connected to a PIC.

console_part

```
: location
| manufacturer
| IP ':' Ipaddr
| wsdevice
| AUTHENTICATION ':' conauth_list END AUTHENTICATION
```

For every console the location and manufacturer may be specified exactly once. If the console is an X terminal (or the console of another UNIX machine) the **IP** address of the console must be specified here. The IP address will be checked for uniqueness within the SyCOS network. If the console is connected directly to the PIC the device the console is attached to must be specified, except if the console is the first and only bitmapped display connected to the PIC. If an authentication strategy different from the network default is required the optional **AUTHENTICATION** clause overrides the network default.

modem_parts

```
: modem_part
| modem_parts modem_part
```

For every modem connected to a PIC or FS the properties of the modem are specified in a non-empty list. None of the information is currently used by any part of SyCOS.

modem_part

```
: manufacturer
| location
| DIALIN wsdevice
| DIALOUT wsdevice
```

These specify the location and manufacturer of the modem and the UNIX devices the modem is attached to for incoming and outgoing connections. None of this information is currently used by SyCOS.

pager_parts

```
: pager_part
| pager_parts pager_part
```

For every pager connected to a PIC the properties of the pager are specified in a non-empty list.

pager_part

```
: manufacturer
| location
| wsdevice
| CONFIGURATION ':' String
```

These specify the location and manufacturer of the pager and the UNIX devices the pager is connected to. The **CONFIGURATION** clause specifies the name of the pager configuration file and is the only part that is currently used by SyCOS.

printer_parts

```
: printer_part
| printer_parts printer_part
```

A printer connected to a PIC or FS is described by a non-empty list of properties. None of these is currently used in SyCOS.

printer_part

```
: location
| manufacturer
| IP ':' Ipaddr
| TYPE ':' Identifier
| wsdevice
```

These specify the location and manufacturer of the printer and the UNIX device or network address as well as the type of the printer. None of these is currently used in SyCOS.

mass_parts

```
: mass_part
```

```
| mass_parts mass_part
```

mass_part

```
: location
| manufacturer
| CAPACITY ':' Constant
| wsdevice
```

These constructs describe mass storage devices (disks and tape drives) connected to a PIC or FS. None of the information is currently used by SyCOS.

wsdevice

```
: DEVICE ':' String dev_parts END DEVICE
```

dev_parts

```
: dev_part
| dev_parts dev_part
```

dev_part

```
: TYPE ':' BLOCK
| TYPE ':' RAW
| MAJOR ':' Constant
| MINOR ':' Constant
```

All of these constructs describe UNIX devices. A UNIX device is specified by its name (e.g. "/dev/tty") its **TYPE** (block or raw device) and a major number used to select the appropriate device driver in the UNIX kernel and a minor number used to select the channel or subdevice.

netauth_list

```
: netauth
| netauth_list netauth
```

netauth

```
: NONE
| SESSION
| FUNCTION
| TIMEOUT Constant
| COVER
| UNIQUE
| COMMENT
```

conauth_list

```
: conauth
| conauth_list conauth
```

conauth

```
: NONE
| SESSION
| FUNCTION
| TIMEOUT Constant
| COVER
| UNIQUE
| COMMENT
```

Authentication properties can be specified on the network or on the console level with the console level information overriding the network default. Authentication types are **NONE** (for not authentication required), **SESSION** (for session wide authentication). **FUNCTION** (for authentication per function executed) and **TIMEOUT** (for authentication that is to be valid for the time in seconds specified in this clause). **COVER** will iconify all applications when no authentication is currently active, UNIQUE if specified will assure that an authenticated user is only authenticated on a single operator display within the system, **COMMENT** will negotiate a comment from the operator for every single action performed from the display. The last three properties can be combined with one of the authentication styles **SESSION**, **FUNCTION** and **TIMEOUT**.

Further information of the authentication and authorrization subsystems can be found in the "SyCOS System Authorization" manual.

adi_list

```
: adi
| adi_list adi
```

Within a workstation description there may be specified a non-empty list of alarm destinations and their implementation. Although this may be done for every workstation (FS and PIC) it should only be done for PIC's with the possible exception of the BHF alarm destination.

adi

```
: ADI ‘:’ Constant Identifier IMPLEMENTED BY Identifier
| ADI ‘:’ Constant Identifier IMPLEMENTED BY Identifier
   Identifier
| ADI ‘:’ Constant Identifier IMPLEMENTED BY Identifier
   Identifier Identifier
| LOGFILE ‘:’ Constant Identifier
| PAGER
| BHF
```

This construct defines alarm destinations (**ADI**’s), creates the association between the name of an alarm destination (as specified by the first Identifier in the ADI clause) and the alarm destination index (as specified by the Constant after the ‘:’, used by the database and the alarm subsystem) and defines the implementation on the UNIX side for the ADI thus defined. Up to three previously defined entries of type PRINTER or CONSOLE may be specified as implementations of an ADI entry.

Two ADI’s (**PAGER** and **BHF**) have predefined indices and implementations and only need to be specified in a boolean manner. The **LOGFILE** ADI can have various indices and names but the implementation is system defined.

trend_list

```
: trend
| trend_list trend
```

trend

```
: TYPE ‘:’ trendtype_list
| QUOTA ‘:’ Constant
| PATH ‘:’ String
| PATH ‘:’ Identifier
| HARDQUOTA ‘:’ Constant
| SOFTQUOTA ‘:’ Constant
```

These clauses determine the properties of ad-hoc trending on a given PIC. The **TYPE** clause specifies the type of trending for the PIC or FS (see below). The **QUOTA** clause limits the amount of disk space reserved for ad-hoc trend data to the specified amount (in MB). When the **SOFTQUOTA** limit is reached (specified in % of the **QUOTA**) messages will be generated on the authentication window to notify users of this fact. When the **HARDQUOTA** limit is reached (again, specified in % of the **QUOTA**) no additional trends can be started.

The two types of **PATH** directives specify the path name of a local directory (**GLOBAL** and **LOCAL** type trends) or the name of a host (for **REMOTE** type trends) where the trend data will be stored. Only one of these must be specified.

trendtype_list

```
: trendtype
| trendtype_list trendtype
```

trendtype

```
: GLOBAL
| LOCAL
| REMOTE
```

The trend type determines the visibility of trend information. **GLOBAL** trends store the data locally (if a point is not already trended somewhere else) and are visible on all PIC’s within the network, **LOCAL** trends are stored on the local machine and are only accessible from the machine that ‘owns’ the trend. **REMOTE** trends store the trend data on some remote machine (specified with a **PATH** directive, see above) and are accessible by all machines on the net.

Note that a given machine can provide both GLOBAL and LOCAL trends (other combinations are not supported).

Further information on ad-hoc trending can be found in the “SyCOS Trend Administration” manual.

servertype

```
: TIME
| FILE
| DISTRIBUTION
| AUTHORIZATION
```

Within a server clause this part specifies the type of the server.

serverclass

```
: SLAVE
| MASTER
| CLIENT
```

Within a server clause this specifies the class of a server. Currently the **SLAVE** and **CLIENT** class is provided but not used within SyCOS.

mfc_parts

```
: mfc_part
| mfc_parts mfc_part
```

This list describes the hardware and software components for an MFC. The sequence of items is irrelevant as long as items are defined before they are used.

mfc_part

```
: location
| manufacturer
| CPU ':' String
| MEMSIZE ':' Constant
| IP ':' Ipaddr
| BOOT HOST ':' Identifier
| NFSMOUNT ':' mount_parts END NFSMOUNT
| NFSAUTH ':' nfsauth_parts END NFSAUTH
| INTERFACE ':' Identifier interface_parts END INTERFACE
| WATCHDOG ':' Identifier watchdog_parts END WATCHDOG
| POS ':' Identifier pos_parts END POS
|
```

The description of an MFC has some fields in common with the description of a workstation. As usual the ordering of parts is free, as long as items are defined before they are used.

Every MFC can have exactly one location and manufacturer clause.

The **CPU**, memory size and IP address of the machine must be specified exactly once. The string specified for the CPU type must be one given in a seperate table in the next section (“Target CPUs” on page 17).

The memory size (**MEMSIZE**) is specified in Megabytes. This information is currently not used by any part of SyCOS.

All **IP** addresses are checked for uniqueness and specification of a duplicate IP address results in an error but are currently not checked for other criteria (e.g. it is possible to define an IP addresses that is in a different IP network and therefore not reachable).

The **BOOT HOST** specification gives the host name of the PIC or FS the MFC should boot from and must occur exactly once.

NFSMOUNT clauses describe the Network File System mounts that the MFC must perform at boot time. This clause can be repeated as necessary for every mount needed.

The **NFSAUTH** clause specifies the authorization used to access files on the file system(s) mounted. The current NFS implementation on VxWorks only honours the first **NFSAUTH** specification and does (contrary to the VxWorks documentation!) not support host specific NFS authorization. Future versions of VxWorks may fix this problem.

The **INTERFACE** specification names an interface card and describes the properties of the interface card. This clause can be repeated as needed.

The **WATCHDOG** construct names and specifies the hardware watchdog device and its properties and may only occur exactly once.

The **POS** clause name and configure special interface cards and associated software.

mount_parts

```
: mount_part
```

```
| mount_parts mount_part
```

mount_part

```
: HOST ':' String
| FILESYSTEM ':' String
| DEVICENAME ':' String
```

Each NFS mount performed from an MFC must specify the HOST the file system to mount resides on, the path name of the file system to mount and the local device file name the file system should be mounted under.

nfsauth_parts

```
: nfsauth_part
| nfsauth_parts nfsauth_part
```

nfsauth_part

```
: HOST ':' String
| UID ':' Constant
| GID ':' Constant
```

In order to access files on a UNIX host through NFS each access request must provide authentication information. This information takes the form of a numerical user and a numerical group id and is used to determine access privileges when files are accessed. The current VxWorks NFS implementation uses the same UID and GID for all accesses regardless of the host accessed.

Note that it is good practice to actually create the UNIX user and group (by convention these should be named ‘extol’) on all hosts accessed so that the hosts can provide UID/GID to name translation services.

Future versions of the configuration file compiler will require **USER** directives defining the UID and GID used in NFS authentication clauses.

interface_parts

```
: interface_part
| interface_parts interface_part
```

interface_part

```
: location
| manufacturer
| TYPE ':' Identifier
```

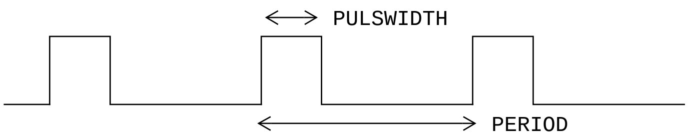
watchdog_parts

```
: watchdog_part
| watchdog_parts watchdog_part
```

watchdog_part

```
: location
| manufacturer
| PULSWIDTH ':' Constant
| PERIOD ':' Constant
| CYCLE ':' Constant
| LINKCHECK ':' TRUE
| LINKCHECK ':' FALSE
```

Every MFC must have exactly one system watchdog device. The location and manufacturer of the watchdog device can be specified by the location and manufacturer clauses. The output square wave is specified by the **PULSWIDTH** and **PERIOD** clauses.



The **CYCLE** time specifies the internal clock frequency used by the watchdog driver and must be set to 1 for the current release.

The **LINKCHECK** clause specifies, whether the system watchdog should also monitor connectivity to PIC’s: If **LINKCHECK** is **TRUE** the watchdog will cease to pulse if no PIC is reachable. If **LINKCHECK** is **FALSE** the watchdog will continue to pulse, even if no PIC can be reached.

pos_parts

```
: pos_part
```

```
| pos_parts pos_part
```

pos_part

```
: location
| manufacturer
| CONFIGURATION ':' String
```

These constructs allow the specification of location and manufacturer information for POS's systems. All of these need specific configuration files, whose path names are specified in the **CONFIGURATION** clauses.

user_parts

```
: user_part
| user_parts user_part
```

user_part

```
: UID ':' Constant
| GID ':' Constant
| HOME ':' String
| SHELL ':' String
| PASSWD ':' String
| COMMENT ':' String
```

These clauses are used to specify a UNIX user on PIC's and FS's. Each clause must be specified exactly once, except for the **COMMENT** field, which may be omitted. The information provided will be used in future versions to create the appropriate UNIX user on the target network using **UID** as the users user-id, **GID** as the users group-id, **HOME** as the users home directory and **SHELL** as the users command interpreter. The **PASSWD** field contains the encrypted password of the user, **COMMENT** information is used to initialize the comment field of the UNIX password file entry. Note that none of the string valued attributes of a user may contain the character ':'.

location

```
: LOCATION ':' String
```

Specifies the location of a piece of equipment.

manufacturer

```
: MANUFACTURER ':' String
```

Specifies the manufacturer of a piece of equipment.

4.3. CPU Types supported

4.3.1. Host CPUs

The following CPU types are available and can be specified as PIC or FS (hostt) CPU types in the configuration file:

Name	Description
HP-B132L	HP Unix Workstation type B132L
HP-B180L	HP Unix Workstation type B180L
HP-B2600	HP Unix Workstation type B2600
HP-D250	HP Unix Server type D250
HP-L1000	HP Unix Server type L1000
HP-WS	HP Unix Workstation - any type
HP-FS	HP Unix Server - any type
PC386	Intel based 386 PC system (Linux)
PC486	Intel based 486 system (Linux)
PENTIUM	Intel based Pentium system (Linux)
PENTIUMPRO	Intel based PentiumPro system (Linux)

Table 1: Host (PIC/FS) CPU names for configuration file

4.3.2. Target CPUs

The following CPU types are available and can be specified as MFC (target) CPU types in the configuration file:

Name	Description
MVME147(ABCDE)	Motorola 68K CPU boards with different sizes of RAM of type MVME 147
MVME167(ABCDE)	Motorola 68k CPU boards with different sizees of RAM of type MVME 167
MVME177	Motorola 68K CPU boards of type MVME 177
X86	Vision Compact Computer from Rotec
I486DX2	Vision Compact II Computer from Rotec

Table 2: Target (MFC) CPU names for configuration file

Name	Description
PC386	Standar 386 type PC using BSP pc386 from Windriver
PC486	Standard 486 type PC, using BSP pc486 from Windriver
PENTIUM	Standard Pentium type PC, using BSP pcPentium from Windriver
PENTIUMPRO	Standard Pentium Pro type PC, using BSP pcPentiumpro from Windriver
CPV5350	Industrial Motorola CPU CPV5350 with pentium processor

Table 2: Target (MFC) CPU names for configuration file

4.4. Example

The following is an annotated example of a system configuration file for a small installation consisting of one FS, one PIC with two consoles and one MFC.

```
NETWORK: GBY_B110
  COSROOT: "/opt/sycos"
  COSAPPL: "/home/b110"
  COSLOGS: "/home/bhf"
  COSDEVS: "/dev"

  AUTHENTICATION:
    TIMEOUT 60
  END AUTHENTICATION

  USER:  sycos
    UID:           1000
    GID:           1000
    COMMENT:       "System Adminstration Account"
    SHELL:         "/bin/ksh"
    HOME:          "/home/sycos"
  END USER

  # File Server Configuration
  FS: fs01
    LOCATION:      "b110 plant at cgg"
    MANUFACTURER:  "Hewlett-Packard"
    CPU:           "HP-D250"
    MEMSIZE:  64   IP:  192.37.248.6

    # Description of devices connected to this FS
    CONSOLE:  fs01
      LOCATION:      "b110 plant at cgg"
      MANUFACTURER:  "Hewlett-Packard"
      AUTHENTICATION:
        SESSION UNIQUE COVER
      END AUTHENTICATION
    END CONSOLE

    STORAGE:  dat
      MANUFACTURER:    "HP"     CAPACITY:   2000
      DEVICE:   "/dev/rmt/0m"
        TYPE: RAW MAJOR: 54 MINOR: 0x0e0302
      END DEVICE
    END STORAGE

    # Ad-hoc trend configuration
    TREND: fs01Trend
      TYPE:        GLOBAL
      QUOTA:       1
      SOFTQUOTA:   80
      HARDQUOTA:   90
      PATH:        "/home/trend"
    END TREND
```

```
    # Alarm destinations
    ALARMDESTINATIONS:
      BHF
    END ALARMDESTINATIONS
  END FS

  # PIC in the network
  PIC: pic01
    LOCATION:          “b110 plant at cgg”
    MANUFACTURER:      “Hewlett-Packard”
    CPU:               “HP-B132L”
    MEMSIZE:  64       IP:  192.37.248.8

    # Description of devices connected to this PIC
    PRINTER: at011
      LOCATION:        “b110 plant at cgg”
      MANUFACTURER:    “alarm typer impact printer”
      DEVICE: ”/dev/at011”
        TYPE: RAW MAJOR: 1 MINOR: 0x090004
      END DEVICE
    END PRINTER

    CONSOLE: pic01
      LOCATION:        “b110 plant at cgg”
      MANUFACTURER:    “Hewlett-Packard”
      AUTHENTICATION:
        SESSION UNIQUE COVER
      END AUTHENTICATION
    END CONSOLE

    CONSOLE: ocx01
      LOCATION:        “b110 plant at cgg”
      MANUFACTURER:    “HP 700/RX”
      IP:  192.37.248.22
      AUTHENTICATION:
        TIMEOUT 1800 UNIQUE
      END AUTHENTICATION
    END CONSOLE

    TREND: pic01Trend
      TYPE:  GLOBAL
      QUOTA:       10
      SOFTQUOTA:  80
      HARDQUOTA:  90
      PATH:        “/home/trend”
    END TREND

    # Application dependent description
    ALARMDESTINATIONS:
      ADI:   10   ocx01  IMPLEMENTED BY ocx01
      ADI:   13   at011  IMPLEMENTED BY at011
    END ALARMDESTINATIONS
  END PIC
```

```
   # The MFC configuration information
   MFC: mfc01
     LOCATION:        "b110 plant at cgg"
     MANUFACTURER:    "Motorola"
     CPU:             "CPV5350"
     MEMSIZE:  128    IP:  192.37.248.12

     # This specifies the host we boot from. It must
     # have been defined already!
     BOOT HOST: fs01

     NFSAUTH:
       HOST: "fs01"  UID: 2200 GID: 1000
     END NFSAUTH
     NFSMOUNT:
       HOST:  "fs01"    FILESYSTEM:"/home/b110/extol"
       DEVICENAME:      "/extol"
     END NFSMOUNT

     WATCHDOG: WD00001
       LOCATION:        "b110 plant at cgg"
       MANUFACTURER:    "Pepperl & Fuchs"
       PULSWIDTH:  1    PERIOD:     5
       CYCLE:      1    LINKCHECK:  FALSE
     END WATCHDOG

     POS: pos
       LOCATION:        "b110 plant at cgg"
       MANUFACTURER:    "Spektra Tek UK"
       CONFIGURATION:
       "/home/b110/database/pos/mfc01pos_mfile.txt"
END POS

   END MFC

   # System error devices. These can only be defined
   # *after* the names used here have been defined.
   ERROR DEVICE:    at011
   ERROR DEVICE:    ocx01
   ERROR DEVICE:    BHF
END NETWORK
```

5. Utilities

5.1. Compiling the configuration file: The cfgcc command

The **`cfgcc`** command is used to compile the system configuration file into an internal (XDR) form. The compiler also checks the configuration information for consistency and flags errors.

The compiler is called as

`cfgcc` `[`**`-o`** *`output_file`*`]` *`input_file`*

If no output file is specified the output file name is derived from the input file name by substitution of any trailing suffix with **`.xdr`**. If the input file does not have any suffix **`.xdr`** is appended to the input file name for the output file name.

Other output file names can be specified with the **`-o`** option. If the output file already exists it will be overwritten without warning, unless the file system permissions deny such access.

SyCOS programs expect the compiled configuration file under the file name *$COSAPPL/database/cfgfile.xdr* so after a successful compilation the configuration file must be moved to this place so the rest of SyCOS can use it.

Alternatively you can also specify *$COSAPPL/database/cfgfile.xdr* as output file name with the -o option.

Note that all running programs must be stopped and restarted and all MFC's must be rebooted after a newly compiled configuration file has been installed as the configuration file is read at start-up (respectively boot) time only.

5.2. Accessing the compiled configuration file: The cfg command

The cfg utility accesses a compiled configuration file and retrieves data from the configuration file for viewing or perusal in shell procedures.

The cfg utility is called as

```
cfg  [-c configfile] [-d] [-n] [-m mastertype]
     [-l list-type] [-t hostname]
```

where the **`-c`** option can be used to specify an alternative configuration file (by default the standard configuration file *$COSAPPL/database/cfgfile.xdr* is used).

The **`-d`** option (partially) regenerates the text representation by dumping a textual representation of the configuration information to the standard output.

The other options are used to extract information of a specific type:

The **`-n`** option returns the name of the SyCOS network.

The **`-m`** option accepts one of **`time`**, **`file`**, **`dist`** or **`auth`** as argument and delivers the name of the master time, file, distribution or authorization server.

The **`-l`** options can be used to list all nodes of type **`P`**(IC), **`M`**(FC), **`F`**(S) or **`W`**(S) (PIC's and FS's are collectively called workstations or WS's).

The **`-t`** option will retrieve the trend directory name for a PIC given as argument.

Note that this utility is not yet fully implemented. Future versions of the utility will allow access to virtually all information stored in a configuration database. The calling conventions currently used may have to change to accommodate the added functionality

6. Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

P

Q

R

S

T

U

W