HP NonStop ASAP Hybrid Manual

Abstract

HP NonStop™ ASAP Hybrid is an extension to the ASAP product family that integrates application availability information from external systems running the Linux operating system into ASAP on one or more HP NonStop servers running the NonStop operating system.

This document describes how to install, configure, and use NonStop ASAP Hybrid.

Product Version

V01

Supported Release Version Updates (RVUs)

This manual supports D46.00 and subsequent D-series RVUs and G06.06 and subsequent G-series RVUs until otherwise indicated in a new edition.

Part Number	Published
529729-001	January 2005

Document History

Part Number	Product Version	Published
529729-001	V01	January 2005

HP NonStop ASAP Hybrid Manual

Index Figures

3. Managing ASAP Hybrid for Linux (continued)

4. The ASAP Hybrid for Linux API

A. ASAP Hybrid EMS Event Messages

Index

Figures

What's New in This Manual

Manual Information

Abstract

HP NonStop™ ASAP Hybrid is an extension to the ASAP product family that integrates application availability information from external systems running the Linux operating system into ASAP on one or more HP NonStop servers running the NonStop operating system.

This document describes how to install, configure, and use NonStop ASAP Hybrid.

Product Version

V01

Supported Release Version Updates (RVUs)

This manual supports D46.00 and subsequent D-series RVUs and G06.06 and subsequent G-series RVUs until otherwise indicated in a new edition.

Part Number	Published
529729-001	January 2005

Document History

Part Number	Product Version	Published
529729-001	V01	January 2005

New and Changed Information

This is a new manual.

About This Manual

Audience

The intended audience for this document is system managers or administrators responsible for deploying ASAP Hybrid, as well as developers who are instrumenting applications to use the ASAP Hybrid API on Linux.

Related Documents

ASAP Server Manual
ASAP Extension Manual
ASAP Client Manual
ASAP Messages Manual

Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

> This requirement is described under Backup DAM Volumes and Physical Disk Drives on page 3-2.

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
MAXATTACH
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

```
file-name
```

computer type. `Computer type` letters within text indicate C and Open System Services (OSS) keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
myfile.c
```

italic computer type. *Italic computer type* letters within text indicate C and Open System Services (OSS) variable items that you supply. Items not enclosed in brackets are required. For example:

```
pathname
```

[ ] Brackets. Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name

INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num  ]
   [ -num ]
   [ text ]

K [ X | D ] address
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }

ALLOWSU { ON | OFF }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address [ , new-value ]...

[ - ] {0|1|2|3|4|5|6|7|8|9}...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;

LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]...
```

Change Bar Notation

Change bars are used to indicate substantive differences between this manual and its preceding version. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

> The message types specified in the REPORT clause are different in the COBOL environment and the Common Run-Time Environment (CRE).
>
> The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

1 Overview

Introduction to ASAP Hybrid

ASAP Hybrid extends the capabilities of ASAP application monitoring to remote systems running the Linux operating system. Application metrics from the remote systems are seamlessly integrated into ASAP on one or more NonStop servers, thereby providing an overall view of the entire application as it spans one or more NonStop servers and one or more Linux servers.

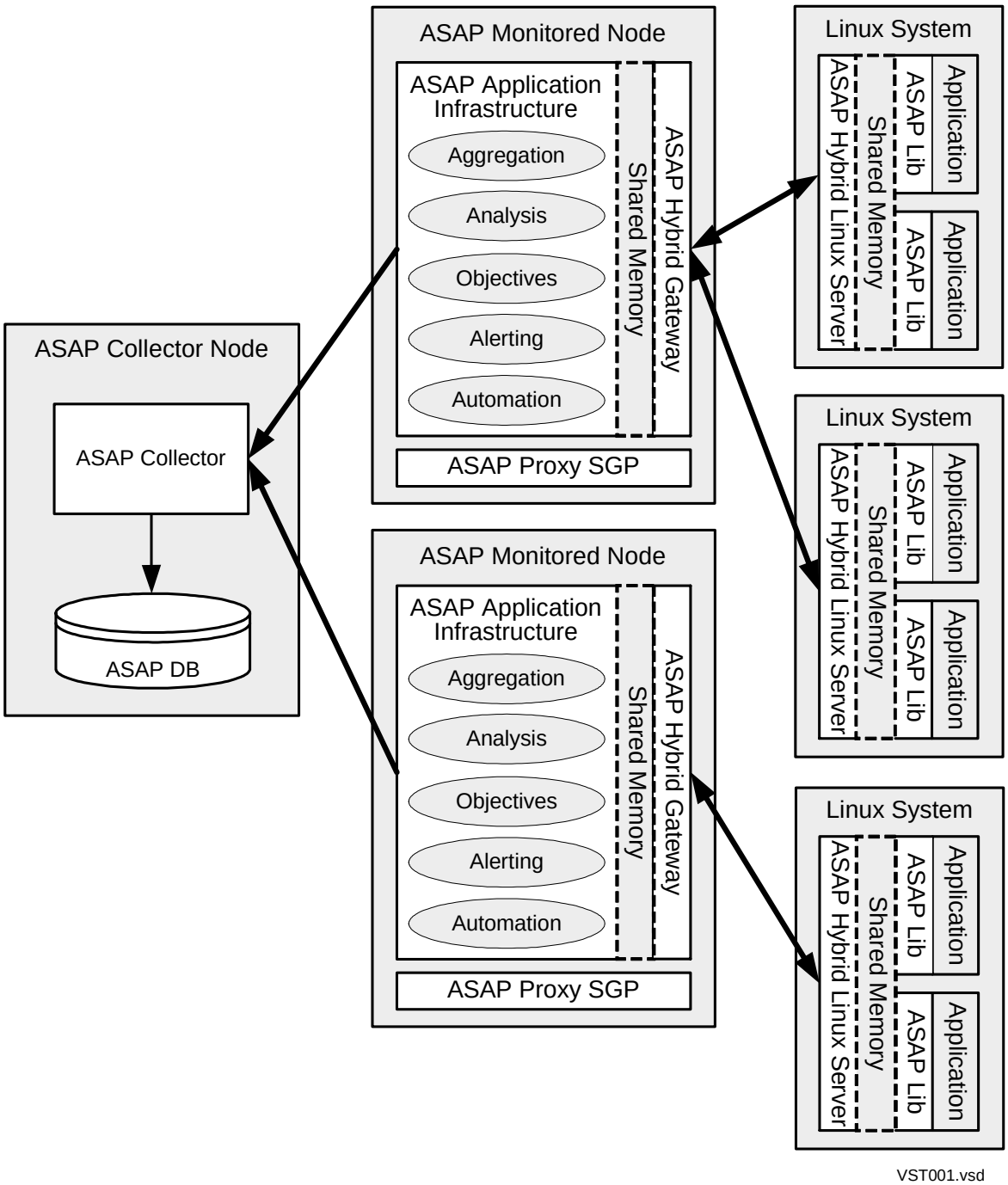
For an application, using ASAP on Linux is the same as using ASAP on a NonStop server. The procedure calls and interactions are fundamentally the same, and the same shared-memory architecture is used on both platforms to provide an ultra-high performance, nonblocking interface.

Metrics from the remote Linux systems are handled within ASAP in the same manner as metrics from NonStop systems or applications. All standard ASAP capabilities and features can be used with any of these metrics, including domain-based aggregation, alerting, data retention, automation, and access to all published ASAP interfaces, including HP OpenView. ASAP does not treat data received from a Linux system any differently than it treats data received from a NonStop system.

ASAP Hybrid Architecture

ASAP Hybrid consists of several different components, some residing on the Linux system and others residing on the NonStop server. The host components are collectively referred to as ASAP Hybrid for NonStop Server (T0404), and the Linux components are referred to as ASAP Hybrid for Linux (T0405). See Figure 1-1, ASAP Hybrid Components and Architecture: ASAP Hybrid Components and Architecture.

Figure 1-1. ASAP Hybrid Components and Architecture



Each of these components has a specific role within the overall ASAP infrastructure:

- ASAP Hybrid Linux API Library

  The ASAP Hybrid Linux API Library provides the interface layer between Linux applications and the ASAP subsystem. Applications link to this library and call ASAP Hybrid API procedures in order to register domains and update metric counters in shared memory. Both thread-safe and non-thread-safe versions are supplied for most procedures. The thread-safe variants allow application counters to be shared among multiple threads of a single process.

- ASAP Hybrid Linux Server

  The ASAP Hybrid Linux Server is responsible for capturing application metrics from the local Linux system and routing them to ASAP on the NonStop server. It also monitors the domains registered on the Linux machine in order to detect application failures and permit recovery of metric data.

- ASAP Proxy SGP

  The ASAP Proxy SGP is a NonStop server-based component responsible for startup, shutdown, and command processing for the ASAP Hybrid subsystem. The SGP provides full integration of other ASAP Hybrid components into the main ASAP environment.

- ASAP Hybrid Gateway

  The ASAP Hybrid Gateway is a NonStop server-based component that provides gateway services to ASAP Hybrid Servers running on Linux. One or more Hybrid Gateways receive metrics from one or more Hybrid Linux Servers and pass them to ASAP Version 2.5 or higher (minimum) via shared-memory.

Once the ASAP subsystem receives the data from the Linux systems, it processes that data in the same fashion as all other ASAP data. It computes application availability statistics based on user-defined counters and formulas defined in the ASAP application Entity Definition Language (EDL) and compares those computed values with any user-defined objectives specified via ASAP GOAL commands. When objectives are not met, ASAP can alert the condition using a variety of mechanisms, including EMS, the ASAP Client, Web ASAP, HP OpenView, email, and many others. Furthermore, all Linux metric data is displayed in the various ASAP client interfaces and can be retained for historical purposes in the ASAP database.

2

Managing ASAP Hybrid for NonStop Server

This section covers the procedures for installing, configuring, and managing the NonStop server-based components of ASAP Hybrid.

Installing ASAP Hybrid on the NonStop Server

ASAP Hybrid is packaged as a 2-CD set. The first CD contains the Linux components, and the second contains the NonStop server components.

To install ASAP Hybrid on a NonStop server, proceed as follows using the CD containing the NonStop server components:

1. Use the IPSetup program to place the Hybrid object, EDL, and configuration files into a product subvolume, into the subvolume where ASAP Server has been installed, and/or into the SOFTDOC subvolume. IPSetup places these files:

File Name	Placed	Description
ASAPPXY	A,B	Hybrid SGP (proxy SGP) object file
ASAPGATE	A,B	Hybrid Gateway object file
ASAPPCNF	A.B	Default Hybrid SGP configuration file; it defines one Gateway process
ASAPGCNF	A,B	Configuration file for the one Gateway process
ASAP2HYB	A,B	ASAP Hybrid entity EDL definition
INSTALL	A	Installation macro for reinstalling ASAP Hybrid from the installation subvolume
SOFTDOC	A,C	The product SOFTDOC file

   A - The product subvolume; for example, $DSMPM.ZASAPH.
   B - The installation subvolume; normally $SYSTEM.SYSTEM.
   C - The SOFTDOC subvolume; for example, $SYSTEM.ZSOFTDOC.

2. Edit the ASAP configuration file, ASAPCONF, and add this line before any START commands:

   ```
   SET PROXY ON
   ```

3. If the Hybrid object files described in Step 1 were not installed to $SYSTEM.SYSTEM, also add this line to the ASAPCONF file:

   ```
   SET PROXYOBJECT $vol.subvol.ASAPPXY
   ```

   where $*vol*.*subvol* specifies the volume and subvolume where the ASAP Hybrid object files were placed in Step 1.

4. Create a new configuration file for the ASAP Proxy SGP to define the characteristics of the overall Hybrid environment. By default, this file should be named ASAPPCNF and should be located in $SYSTEM.SYSTEM. If the file has a different name or is located in a subvolume other than $SYSTEM.SYSTEM, add this line to the ASAPCONF file:

   ```
   SET PROXYCONFIG $vol.subvol.filename
   ```

   where $*vol.subvol.filename* references the fully qualified name of the configuration file you just created.

   For detailed information on the contents of the Proxy SGP configuration file, see Configuring the Proxy SGP on page 2-4.
5. Create a new configuration file for each Hybrid Gateway you defined in the Proxy SGP configuration file created in Step 4. For detailed information on the contents of the Gateway configuration files, see Configuring Hybrid Gateways on page 2-5.
6. Restart ASAP, and ensure that no configuration error messages are written to the EMS event log while ASAP Hybrid processes its configuration files during startup.

Configuring ASAP Hybrid on the NonStop Server

ASAP Hybrid for NonStop servers is configured in three different locations. ASAP is configured to know about and control ASAP Hybrid; the Hybrid SGP is configured to start Hybrid Gateways; and each Hybrid Gateway is configured independently.

Configuring ASAP to Enable ASAP Hybrid

ASAP Hybrid is configured within ASAP by placing SET commands in the ASAPCONF file.

SET PROXY Command

The SET PROXY ON command enables ASAP Hybrid within ASAP Server. Hybrid settings are not visible within ASAP until this command is executed. SET PROXY ON is required.

```
SET PROXY ON
```

Example

```
SET PROXY ON
```

SET PROXYCONFIG Command

The SET PROXYCONFIG command defines the location of the Hybrid Proxy SGP configuration file. The default is $SYSTEM.SYSTEM.ASAPPCNF. SET PROXYCONFIG *filename* is required if the location of the configuration file is not the default location.

```
SET PROXYCONFIG filename
```

filename

is the fully qualified name of the Proxy SGP configuration file.

Example

```
SET PROXYCONFIG $SYSTEM.ASAP.ASAPPCNF
```

SET PROXYCPU Command

The SET PROXYCPU command defines the CPU in which the Hybrid Proxy SGP runs. The default is the CPU where ASAPMON executes.

```
SET PROXYCPU number
```

number

is a valid CPU number from 0 through 15.

Example

```
SET PROXYCPU 2
```

SET PROXYOBJECT Command

The SET PROXYOBJECT command defines the object file for the Hybrid Proxy SGP. The default is $SYSTEM.SYSTEM.ASAPPXY. SET PROXYOBJECT *object-filename* is required if the location of the object file is not the default location.

```
SET PROXYOBJECT object-filename
```

object-filename

is the name of the Proxy SGP object file.

Example

```
SET PROXYOBJECT $SYSTEM.ASAP.ASAPPXY
```

SET PROXYPARAM Command

The SET PROXYPARAM command defines the parameters for the Hybrid Proxy SGP.

```
SET PROXYPARAM "parameter  [ parameter …]"
```

parameter

is a Hybrid parameter. This item is for future use. Currently there are no user-configurable Hybrid parameters.

Configuring the Proxy SGP

The Proxy SGP is configured by placing GATEWAY statements in a configuration file that the SGP reads at startup.

GATEWAY Statement

The GATEWAY statement defines Hybrid Gateway processes to be started by the Proxy SGP. There must be at least one GATEWAY statement in the Proxy SGP configuration file. The GATEWAY statement must be entered on a single line.

```
GATEWAY  NAME     [=] $name
         CONFIG   [=] filename
         [ CPU    [=] cpu ]
         [ OBJECT [=] object-filename ]
         [ PRI    [=] priority ]
```

NAME $*name*

is the process name to use when creating the Hybrid Gateway. Hybrid Gateway processes must be named. There is no default.

CONFIG *filename*

is the name of the configuration file for the Hybrid Gateway. Each Hybrid Gateway requires its own unique configuration file. There is no default.

CPU *cpu*

defines the CPU in which the Gateway runs. This value must be a valid CPU number from 0 through 15. Hybrid Gateways should be spread across processors to take advantage of the parallelism inherent on the NonStop platform. The default is CPU 0.

OBJECT *object-filename*

is the object filename for the Hybrid Gateway. The default is $SYSTEM.SYSTEM.ASAPGATE.

PRI *priority*

is the execution priority for the Hybrid Gateway. The default is 150.

Considerations

The number of Hybrid Gateway processes that need to be started depends on several factors: the number of external systems and domains that need to be processed; the number of ASAPX domains that are available in a processor; the amount of processor time a Gateway can obtain to process data; and problems in the network that place upper limits on the amount of data that can flow to a single IP address or port.

When a Hybrid Gateway reaches a limit or begins to show that all domains are not being updated, you should consider starting another Gateway process and repartitioning the domain load. Each Hybrid Gateway should execute in a separate processor, if possible, and should use a separate TCPIP process name, IP address, and port to achieve maximum parallelism.

Example

This example shows the contents of a Proxy SGP configuration file that starts three Hybrid Gateway processes:

```
GATEWAY CONFIG $SYSTEM.ASAP.GATEWAY1 NAME $GATE1
GATEWAY CONFIG = $SYSTEM.ASAP.GATEWAY2 CPU = 1 NAME $GATE2
GATEWAY CONFIG $SYSTEM.ASAP.GATEWAY3 CPU 2 PRI 185 NAME =
$GATE3
```

Configuring Hybrid Gateways

Hybrid Gateway processes are configured by placing statements in a configuration file that is read by each Gateway at startup time. Each Hybrid Gateway requires its own unique configuration file.

CONFIG Statement

The CONFIG statement is used to control memory sizing and other configuration options within the Hybrid Gateway. The CONFIG statement is required. Each Gateway configuration file must contain a CONFIG statement that defines a unique combination of TCP/IP process name and port number to be used by that Gateway process. The CONFIG statement must be entered on a single line.

```
CONFIG [ ASAPID     [=] $asapid ]
       [ AUDITRATE  [=] seconds ]
       [ ENTITY     [=] ON ]
       [ MAXSYSTEMS [=] systems ]
       [ MAXDOMAINS [=] domains ]
       TCPIP        [=] $name
       PORT         [=] port
```

`ASAPID $`*`asapid`*

specifies the ASAP ID into which Hybrid domains are registered when you use the ENTITY ON option. The default is $ZOO.

`AUDITRATE` *`seconds`*

is the number of seconds between Gateway audit intervals. At each audit interval, the Gateway process computes and updates its statistics for the Hybrid entity if ENTITY ON has been specified, and it might perform other functions such as preregistering external domains. The default value is 60 seconds.

`ENTITY ON`

causes the Hybrid Gateway to register into the ASAP Hybrid entity and report detailed statistics about itself. If you choose this option, you must include the ASAP Hybrid EDL file in the EDL definitions for the ASAP environment into which the Hybrid Gateway will register. To do so, place an INCLUDE statement in the ASAPUSER file that references the name of the Hybrid EDL file. That file is normally located at $SYSTEM.SYSTEM.ASAP2HYB but might have been placed in another location during the installation process. For more information on using the Hybrid entity, see The The ASAP Hybrid Entity on page 2-11.

`MAXSYSTEMS` *`systems`*

specifies the maximum number of external systems that the Hybrid Gateway can support. Valid values are 1 through 32767. The default is 1000.

`MAXDOMAINS` *`domains`*

specifies the maximum number of domains that the Hybrid Gateway can support. This value depends on the ASAP Extension MAXDOMAINS setting. Valid values are 1 through 10000. The default is 1000.

`TCPIP $`*`name`*

is the name of the TCPIP process to open when you create a socket to receive data from external systems. The default is $ZTC0.

`PORT` *`port`*

is the port number to use when you create a socket to receive data from external systems. The default is 6301.

Example

```
CONFIG MAXSYSTEMS 100 TCPIP $ZTCP2 PORT 7001
```

SYSTEM Statement

The optional SYSTEM statement is used to preregister application domains from an external system into ASAP. The Gateway registers domains defined with SYSTEM statements at the first audit interval after it starts up. The SYSTEM statement must be entered on a single line.

```
SYSTEM  NAME     [=] name
        DOMAIN   [=] domain
        [ ASAPID [=] $asapid ]
        [ VSN    [=] version ]
        [ FLAGS  [=] flags ]
```

NAME *name*

is the name of the external system. All external systems must be named.

DOMAIN *domain*

is the unique domain name to register. Domain names can be from 1 through 64 bytes in length and can contain 2 through 5 hierarchical levels separated by a backslash character. The first level of the domain name must be an ASAP entity name that is defined to ASAP using the ASAP Entity Definition Language. For more information on defining customer application entities, see the ASAP Client, Server, and Extension manuals.

ASAPID $*asapid*

defines the ASAP ID into which the domains are registered. The default is $ZOO.

VSN *version*

specifies the application version, if any. The built-in Version attribute should be defined in the application EDL file when you use this option.

FLAGS *flags*

is the domain flags value. If this value is 1, it instructs the Gateway to register the domain in a deactivated state.

Examples

These examples show SYSTEM statements that define one real domain from each of the external systems to be monitored:

```
SYSTEM NAME = LINUX01 DOMAIN ATM\SERVER\LINUX01\A00
SYSTEM NAME = LINUX02 DOMAIN ATM\SERVER\LINUX02\A00
SYSTEM NAME = LINUX03 DOMAIN ATM\SERVER\LINUX03\A00
SYSTEM NAME = LINUX04 DOMAIN ATM\SERVER\LINUX04\A00
SYSTEM NAME = LINUX05 DOMAIN ATM\SERVER\LINUX05\A00
SYSTEM NAME = LINUX06 DOMAIN ATM\SERVER\LINUX06\A00
```

ASAP Hybrid Commands

The ASAP CI STATUS command facility is used to retrieve information from running ASAP Hybrid components, to start and stop traces, and to manually remove domains.

```
STATUS PROXY

STATUS PROXY $name

STATUS PROXY $name REMOVE [ FORCE ] domain

STATUS PROXY $name TRACE $filename [ DETAIL | DATA ]

STATUS PROXY $name TRACE STOP
```

The STATUS PROXY command without any parameters returns status information from the Proxy SGP. It returns information on the status and number of Hybrid Gateways and other processes that it manages.

The STATUS PROXY $*name* form of the command directs the command to the Hybrid Gateway process named $*name*. This form of the command is used to return status from a Hybrid Gateway process. It is also used to send TRACE and REMOVE commands to a Hybrid Gateway process.

$*name*

specifies the name of a running Hybrid Gateway process to send the command to. This form of the command returns the status of the Hybrid Gateway process when it is entered without any other command options.

REMOVE

sends a domain removal request to the Hybrid Gateway process. Domains that are up or inactive are removed immediately from the NonStop server. If they are not removed from the Linux server first, however, they will be restored by ASAP on

Linux. Thus domains that are up or inactive must be removed from the Linux system prior to removing them from the NonStop server. Domains in the down state are removed once the external Linux system has been notified of the removal. Domains that are registering or restarting are not removed. For more information on manual domain removal, see the Considerations section.

`FORCE`

immediately removes the domains regardless of their last known state.

`domain`

specifies a complete domain name, or a wild-card domain name ending in an asterisk character, to be removed.

`TRACE`

sends a trace start or stop command to the Hybrid Gateway.

`$`*`filename`*

defines the fully qualified name of a file, terminal, or spooler device to which trace information is written.

`DETAIL`

causes domain detail information to be traced.

`DATA`

causes domain detail information and data values to be traced.

`STOP`

stops the trace.

Examples

- To retrieve status information from the Proxy SGP:

```
STATUS PROXY
```

- To retrieve status information from a Hybrid Gateway process named $GATE1:

```
STATUS PROXY $GATE1
```

- To remove an inactive, up, or down domain named Linuxapp\Domain\Server from Hybrid Gateway process $GATE1:

```
STATUS PROXY $GATE1 REMOVE LINUXAPP\DOMAIN\SERVER
```

- To remove all inactive, up, and down domains whose names start with L:

```
STATUS PROXY $GATE1 REMOVE L*
```

- To remove all inactive, up, and down domains from the Hybrid Gateway process named $GATE1:

```
STATUS PROXY $GATE1 REMOVE *
```

- To remove a down domain that is named Test\App\Server where the domain is no longer known on the external system:

```
STATUS PROXY $GATE1 REMOVE FORCE TEST\APP\SERVER
```

- To start a trace to a file named $Vol.Subvol.File:

```
STATUS PROXY $GATE1 TRACE $VOL.SUBVOL.FILE
```

- To start a trace containing detail domain information:

```
STATUS PROXY $GATE1 TRACE DETAIL $VOL.SUBVOL.FILE
```

- To start a trace containing detailed domain information that also includes the data values:

```
STATUS PROXY $GATE1 TRACE DATA $VOL.SUBVOL.FILE
```

- To stop a running trace:

```
STATUS PROXY $GATE1 TRACE STOP
```

Considerations

At times, domains might need to be manually removed from ASAP. When a Linux application registers one or more domains but then fails or stops without first removing those domains, ASAP reports the domains as down at each sample interval thereafter unless they are reregistered by the Linux application. If the Linux application will never reregister the domains in question, they must be removed manually using the STATUS PROXY $*name* REMOVE command. In this case, ASAP components on both the remote Linux system and on the NonStop server are aware of the down domains. As a result, the REMOVE option removes the domain in both locations by first notifying the Gateway process, which in turn reports the removal to ASAP on the Linux system in response to the next update it receives for that domain.

If the entire Linux system or the network connection fails, ASAP on the NonStop system no longer receives updates on the status of domains impacted by the outage. In this case, ASAP continues to report the state of these domains:

- For domains whose last known state was Down, ASAP continues to report the domains as Down until the outage is corrected and new status is received.
- For domains that had set their own operational state via the ASAP Hybrid for Linux *asap_opstate* procedure, ASAP continues to report that operational state until the outage is corrected and new status is received.
- For all other domains, ASAP reports the domains as Inactive until the outage is corrected and new status is received.

If the failed Linux system is never restarted, all the domains it registered might require manual removal. In this case, the REMOVE option immediately removes inactive domains and those setting their own operational state, but it does not remove down domains. ASAP on the NonStop system continues to wait for the Linux system to report on the domain so that it can notify that system of the removal. Use the FORCE option to remove a down domain from ASAP on the NonStop server when the domain is no longer known on the Linux side.

Domains that are up or inactive must be removed from the Linux system prior to being removed from ASAP on the NonStop server. ASAP on the NonStop server does not notify ASAP on Linux when up or inactive domains are removed; it assumes that the Linux side no longer has knowledge of the domains. If the domains are not removed from the Linux system first, they will be reregistered and processed normally by ASAP on the NonStop server when the next sample is received from that Linux system. For more information on manually removing domains on Linux, see Cleaning Up Domains Interactively on page 3-14.

There is a rare case wherein both ASAP Hybrid on NonStop and on Linux could no longer know about a domain, but ASAP Server continues to report on the domain. This situation can occur if the Linux system or network fails and then the Hybrid Gateway process is stopped or fails. When a domain does not appear anywhere in ASAP Hybrid but does appear in ASAP Server, the ASAP MONITOR command must be used to remove the domain.

Note. Using the ASAP MONITOR command to forcibly remove a domain registered with ASAP Hybrid (MONITOR *entity*, REMOVE, FORCE) can cause the Hybrid Gateway process to abnormally terminate as its memory is taken away. It will recover and continue to operate normally after the termination, but the MONITOR *entity*, REMOVE, FORCE command should be used only in extreme circumstances when there is no other way to manually remove an ASAP domain.

The ASAP Hybrid Entity

Each Hybrid Gateway process has the ability to register itself into ASAP to report detailed statistics about its own performance. The Gateway processes register into an ASAP EDL-defined entity named Hybrid that reports this information:

Hybrid Attribute	**Description** (page 1 of 2)
Datagram Count	The total number of datagrams received
Sample Count	The number of sample datagrams received
Registration Count	The number of domain registrations during the interval
Update Count	The number of domain updates
ASAPX Removal Count	The number of domain removals during the interval
API Error Count	The count of API errors during the interval
Network Turnaround Time	The average turnaround time for an ACK datagram to external systems

Hybrid Attribute	**Description** (page 2 of 2)
System Count	The number of external systems that sample data was received from during the last interval
OpState Count	The number of calls to set the operational text and state
Control Count	The number of calls to activate or deactivate ranking

To configure and enable the Hybrid Entity within ASAP:

1. Use the ENTITY ON option in the CONFIG statement in each Gateway's configuration file.
2. Edit the ASAPUSER file and add an INCLUDE statement pointing to the Hybrid ASAP EDL file, which is normally located at $SYSTEM.SYSTEM.ASAP2HYB. For example:

   ```
   INCLUDE $SYSTEM.SYSTEM.ASAP2HYB;
   ```

3. Shut down and restart the ASAP Server and all ASAP Clients in order to pick up the new EDL.
4. Allow each ASAP Client to synchronize EDL with the ASAP Server when they prompt to do so, or select the Download and Compile Server EDL function to pick up the new EDL within ASAP Client.

3

Managing ASAP Hybrid for Linux

This section covers the procedures for installing, configuring, and managing the Linux-based components of ASAP Hybrid.

Installing ASAP Hybrid for Linux

Installing ASAP Hybrid for Linux essentially involves getting the ASAP Hybrid files onto the Linux system. Once the files are placed, no further steps are required. ASAP Hybrid does not need to be installed to a specific location, nor are additional install scripts necessary to perform initial configuration.

ASAP Hybrid is packaged as a 2-CD set. One CD contains the NonStop server components, and the other contains the Linux components. All files to be installed to the Linux system are contained in a single gzip-ed tar archive file named **asaphybridforlinux.1.0.7.tar.gz** located in the WS_SW directory on the Linux CD.

To extract and install the ASAP Hybrid files:

1. Load the archive onto the target Linux system. The file can be located in any directory to which you have access, but ideally the directory should be empty and should be named appropriately (for example, **asaphybrid.1.0.7**). This directory is referred to throughout the remainder of this section as the *installation directory*.
2. Extract the files contained in the archive using the tar command with the -xzvf options. For example:

   ```
   tar -xzvf asaphybridforlinux.1.0.7.tar.gz
   ```

   This command creates the five subdirectories **bin**, **doc**, **include**, **lib**, and **sample** within the installation directory and places the appropriate files in them. The files installed are:

File Name	Directory	Description (page 1 of 2)
asap.conf.sample	bin	Sample server configuration file
asapremove	bin	Symbolic link to domain removal utility
asapremove.1.0.7	bin	Domain removal utility
asapxserver	bin	Symbolic link to ASAP Hybrid Linux server
asapxserver.1.0.7	bin	ASAP Hybrid for Linux server (daemon) executable
HybridManual.doc	doc	ASAP Hybrid manual in Microsoft Word format
HybridManual.pdf	doc	ASAP Hybrid manual in PDF format
asapx.h	include	ASAP Hybrid for Linux header file
libasapx.a	lib	Symbolic link to 32-bit static API library
libasapx.a.1	lib	Symbolic link to 32-bit static API library
libasapx.a.1.0.7	lib	32-bit static API library

File Name	Directory	Description (page 2 of 2)
libasapx.so	lib	Symbolic link to 32-bit shared API library
libasapx.so.1	lib	Symbolic link to 32-bit shared API library
libasapx.so.1.0.7	lib	32-bit shared API library
libasapx64.a	lib	Symbolic link to 64-bit static API library
libasapx64.a.1	lib	Symbolic link to 64-bit static API library
libasapx64.a.1.0.7	lib	64-bit static API library
libasapx64.so	lib	Symbolic link to 64-bit shared API library
libasapx64.so.1	lib	Symbolic link to 64-bit shared API library
libasapx64.so.1.0.7	lib	64-bit shared API library
sampleclient.cpp	sample	Sample client source code

You can leave the files in these locations, which is the recommended approach. This approach makes it easier to manage the ASAP Hybrid file set and simplifies the process of upgrading to newer versions of ASAP Hybrid as they are released. However, you can move the files to the standard Linux locations (/usr/include for the header file, /usr/lib for library files, and so on) or any other location if you prefer. The only restrictions are:

- The server configuration file (**asap.conf**), if you choose to create one, must reside in the same directory as the Linux server executable (**aspaxserver.1.0.7**). The server reads this configuration file at startup and expects to find it in the same directory as the program executable. For more information on configuration options contained in the **asap.conf** file, see Configuring the ASAP Hybrid for Linux Server on page 3-4.
- The directory containing the Linux server executable must be secured such that files can be created and written in that directory. The server can optionally create files at run time, including log files, trace files, and domain status files. These files will be located in the same directory as the program executable.
- Both **asapremove*** files must be located in the same directory.
- Both **asapxserver*** files must be located in the same directory.
- The three **libasapx.a.*** files must be located in the same directory.
- The three **libasapx.so.*** files must be located in the same directory.
- The three **libasapx64.a.*** files must be located in the same directory.
- The three **libasapx64.so.*** files must be located in the same directory.

Once the ASAP Hybrid files are placed on the Linux system, you can run the server process and begin utilizing the API.

Removing ASAP Hybrid for Linux

To remove ASAP Hybrid for Linux from the system, simply delete all the files and directories identified in Installing ASAP Hybrid for Linux. Because the ASAP Hybrid for Linux installation process does not automatically update any system configuration files or startup scripts, removing the product involves nothing more than removing the component files from the system.

Starting the ASAP Hybrid for Linux Server

The ASAP Hybrid for Linux server is a daemon and is responsible for providing ASAP services to the API. It must be running in order for any client to register a domain and also to forward domain metrics to the ASAP subsystem running on the NonStop hosts. If the server is not running, client applications can still run, but they cannot register domains, and metric values are not communicated to NonStop ASAP.

Ideally the ASAP Hybrid for Linux server should be launched as part of system startup and stopped at system shutdown. However, you can also start the server manually or via a script as needed.

To start the server, from the **bin** directory created at installation, enter this command at a shell prompt:

```
./asapxserver
```

The command returns immediately. This outcome is normal. Because the ASAP Hybrid for Linux server is a daemon, it does not communicate directly with any tty device. The server remains running until the Linux system is shut down or until the server process itself is explicitly stopped via a system signal.

Stopping the ASAP Hybrid for Linux Server

The ASAP Hybrid for Linux server can be stopped in several different ways. First, the server is stopped if the Linux system is shut down or restarted. In addition, the server can be stopped interactively using standard Linux system signals. Depending on which signal is sent, the server process either goes through a controlled shutdown or stops immediately.

Controlled Shutdown

To stop the ASAP Hybrid for Linux server in an orderly manner, send it a SIGTERM, SIGINT, or SIGQUIT signal. For example:

```
kill -s SIGTERM server PID
```

where *server PID* is the process ID of the ASAP Hybrid for Linux server process.

When the server process receives one of these signals, it immediately stops all secondary threads, flushes registration information to its segment information file, cleans up any down or removed domains if configured to do so, and then exits.

Immediate Shutdown

To stop the server immediately without performing an orderly shutdown, send it a SIGKILL signal:

```
kill -s SIGKILL server PID
```

where *server PID* is the process ID of the ASAP Hybrid for Linux server process.

In this case, the server is stopped immediately and does not perform any cleanup-related operations.

Configuring the ASAP Hybrid for Linux Server

Server configuration settings are stored in the **asap.conf** file located in the server's program directory. The server process loads this configuration information immediately at startup. To control server behavior, alter the settings in this file before starting the server process, or else alter the settings and signal the server process to reload the file as described in Dynamically Altering Configuration Options on page 3-14.

The asap.conf Configuration File

The configuration file consists of several distinct sections, each with options pertaining to a particular functional area.

Sections are denoted by a section name contained in square brackets: for example, [Server]. The configuration file can contain any or all of these sections:

- [Comm]

  The [Comm] section contains settings related to communication with the NonStop server.

- [Logging]

  The [Logging] section contains settings related to the server's dedicated logging mechanism. These settings affect the server's logging mechanism only and do not control what information is written to the Linux system log.

- [Server]

  The [Server] section contains settings related to general server operations.

- [Tracing]

  The [Tracing] section contains settings related to the server's built-in trace facility.

Each section of the configuration file can contain one or more option settings that relate to that section. These options are specified using keyword/value pairs, with one such pair given per line. The format for a keyword/value pair is:

```
option keyword [=] option value
```

The equal sign (=) is optional. Some examples of valid configuration option specifications would be:

```
LocalPort = 5000
DomainCleanup False
```

Once the ASAP Hybrid for Linux server encounters a section in the configuration file, all subsequent option specifications are treated as part of that section until another section is encountered or until the end of the configuration file is reached.

Comments can be added to the configuration file wherever desired. Any line beginning with a # character is treated as a comment line by the server and is ignored, as are all lines containing only white space characters.

Configuration Options

The **asap.conf** file can contain any or all of the options described below. If the configuration file does not specify a given option, the server uses the default value defined for that option.

When ASAP Hybrid for Linux is first installed, a sample configuration file named **asap.conf.sample** is created in the server's program directory. You can use this file as the basis for creating a configuration file tailored to your particular environment.

Comm Options

Comm configuration options are contained in the Comm section of the configuration file. These options specify communication-related parameters for the ASAP Hybrid for Linux server, including the name or IP address of the NonStop system to which metric data will be sent, network message size, and pacing interval. Each option is described in detail below.

LocalMachine

Description: The local Linux machine name. By default, this name is retrieved from the system itself. However, in some cases this name might not be set or might be too long to be useful within ASAP. In those situations, the predefined machine name can be overridden within ASAP Hybrid by specifying a LocalMachine name in the configuration file. If specified, this name is used by ASAP Hybrid only and does not impact any other aspect of system operation or management.

Value Range: An ASCII-printable character string from 1 to 255 bytes in length. It cannot contain any backslash, space, quote, comma, colon, semicolon, or asterisk characters.

Default Value: The local machine name

Example: `LocalMachine = MySystemName`

LocalPort

Description: The port number on the local Linux system on which ASAP Hybrid status information will be received.

Value Range: 0 through 32767

Default Value: 6301 (the standard ASAP Hybrid port)

Example: `LocalPort = 5555`

MessageSize

Description: The maximum size, in bytes, of messages exchanged between the Linux system and the NonStop server. Messages larger than 1472 bytes might be fragmented by the IP network layer.

Value Range: 512 through 32000

Default Value: 512

Example: `MessageSize = 1472`

Pacing

Description: This value specifies the delay, in milliseconds, between messages sent to the NonStop server. Increasing this value reduces spikes in network activity and equalizes the flow of traffic, but it increases the overall time it takes to get sample data to the NonStop server.

Value Range: 0 through 2147483647

Default Value: 20

Example: `Pacing = 50`

RemoteIPAddress

Description: The IP address or host name of the NonStop server to which application metric data will be sent.

Value Range: Valid IP addresses or host names

Default Value: 127.0.0.1

Example: `RemoteIPAddress = 192.168.120.99`

RemotePort

Description: The TCP/IP port number on which the NonStop-based ASAP Hybrid Gateway is running.

Value Range: 0 through 32767

Default Value: 6301 (the standard ASAP Hybrid port)

Example: `RemotePort = 5555`

Logging Options

Logging configuration options are contained in the Logging section of the configuration file. These options control the ASAP Hybrid for Linux server's built-in logging mechanism. These options affect the server's logging facility only, and do not in any way relate to messages logged by the server to the Linux system log. Each option is described in detail below.

DeleteOnClose

Description: If True, the log file is deleted when the server process shuts down. Deleting log files when the server process shuts down is an effective means of keeping log files from growing too large. However, doing so also eliminates any logging history, and such history could be useful for debugging purposes. Therefore, set this value be to True only if your environment typically generates a large amount of log activity.

Value Range: True or False

Default Value: False

Example: `DeleteOnClose = True`

Enabled

Description: If True, logging is enabled for the server. If False, logging is disabled, and no messages are generated.

Value Range: True or False

Default Value: False

Example: `Enabled = True`

FileName

Description: The name of the server log file. If logging is enabled, log records are written to this file, in HTML format. If you change this value, be sure to fully qualify the file name. The ASAP Hybrid for Linux server changes its default directory to the root directory at startup. Therefore, specifying a file name only, without path information, causes the log file to be created in the system's root directory, which might not be desirable.

Value Range: A valid Linux file name

Default Value: asaplog.html, located in the server program directory

Example: `FileName = /var/asap/asaplog1.html`

Level

Description: The logging detail level for the server. Level = 4 causes all error, warning, and informational messages to be logged. Level = 3 causes all error and warning messages to be logged. Level = 2 causes all error messages to be logged. Level = 1 causes only critical error messages to be logged. The default value of 3 should be used under normal circumstances.

Value Range: 1 through 4

Default Value: 3 (All error and warning messages are logged.)

Example: `Level = 4`

Server Options

Server configuration options are contained in the Server section of the configuration file. These options control the behavior of the core portions of the ASAP Hybrid for Linux server, including how often the server sends sample data to NonStop ASAP, how many domains can be monitored on the Linux system, and whether or not HTML domain reports are produced every sample interval. Each option is described in detail below.

AuditInterval

Description: Interval, in seconds, at which the server performs automated status and cleanup operations.

Value Range: 0 through 2147483647

Default Value: 60

Example: `AuditInterval = 120`

DomainCleanup

Description: If True, down and removed domains are cleaned up when the ASAP Hybrid for Linux server shuts down. If False, these domains remain in shared memory, and the server reattaches to them the next time it is started. (Domains that are not down or removed always remain in shared memory, and the server always reattaches to them whenever it is restarted, unless the Linux system is shutting down, in which case no domain data is preserved.)

Value Range: True or False

Default Value: False

Example: `DomainCleanup = True`

FlushRegistrations

Description: If True, domain information is preserved in the server's segment information file immediately upon domain registration. This approach reduces the likelihood of a domain registration being lost if the server is stopped or fails shortly after completing a registration. If False, data is only written to the segment information file at each audit interval. The advantage of the latter approach is that performance is improved when large numbers of registrations occur simultaneously. The drawback is a slightly increased risk of a domain registration being lost if a failure occurs.

Value Range: True or False

Default Value: False

Example: `FlushRegistrations = True`

HTMLStatus

Description: If True, the server generates a domain report every sample interval. This file is named **asapdomains.html** and is located in the same directory as the server executable.

Value Range: True or False

Default Value: False

Example: `HTMLStatus = True`

MaxDomains

Description: The maximum number of domains allowed on the Linux system. Attempts to register additional domains result in an error being returned by the API to the calling application.

Value Range: 1 through 10000, though the actual limit is dictated by Linux system performance and resources. In addition, each domain on the Linux system allocates one host domain from the NonStop ASAP environment into which it registers. Thus a given NonStop ASAP environment must be configured to support the total number of domains that will be registered into it across all Linux systems. For example, if 3000 domains, scattered across 10 Linux systems, will report metrics into the ASAP environment $ZOO, the $ZOO environment must be configured to support at least 3000 domains.

Default Value: 1000

Example: `MaxDomains = 200`

SampleInterval

Description: Interval, in seconds, at which the server sends data to NonStop ASAP. If HTMLStatus is set to True, domain status data is also written at this interval to the **asapdomains.html** file. The SampleInterval value should not exceed the smallest ASAP RATE value among all the ASAP environments into which applications on the

Linux system are registered. For example, assume that some of the domains on the Linux system are registered into the ASAP environment $ZOO and that $ZOO has been configured with a RATE of 4 minutes. Assume also that the remainder of the domains on the Linux system are registered into the environment $ZAS and that $ZAS has been configured with a RATE of 1 minute. In this case, the SampleInterval should be configured to be 60 seconds or less because the lowest RATE value in the two ASAP environments is 1 minute (from $ZAS).

Value Range: 0 through 2147483647

Default Value: 60

Example: `SampleInterval = 120`

Tracing Options

Tracing configuration options are contained in the Tracing section of the configuration file. These options define what data is traced, where trace data is written, and how much data is captured and retained. Each option is described in detail below.

Enabled

Description: True if tracing is enabled and False if tracing is disabled. Tracing is disabled by default and should remain disabled unless you are trying to diagnose a problem. In that case, contact HP support so that they can advise you how to best define trace parameters to capture information on the problem you are encountering.

Value Range: True or False

Default Value: False

Example: `Enabled = True`

FileName

Description: The name of the ASAP Hybrid for Linux server trace output file. If tracing is enabled, trace data is written to this file in HTML format. All trace data is captured in memory in order to reduce the impact of tracing on the server process. Trace data is written to the trace output file only when tracing is stopped or when the server process shuts down. If you change this value, be sure to fully qualify the file name. The ASAP Hybrid for Linux server changes its default directory to the root directory at startup. Therefore, specifying a file name only, without path information, causes the trace file to be created in the system's root directory, which might not be desirable.

Value Range: A valid Linux file name

Default Value: asaptrace.html, located in the server program directory

Example: FileName = /var/asap/asaptrace1.html

Mask

Description: Defines the trace mask for the server. This mask is set by combining (adding together) values from the trace mask table below. To trace more than one type of data, add the mask values for the relevant types together. For example, to trace UDP data sent and UDP data received, the mask would be 0x01000000 + 0x02000000 = 0x03000000. Possible mask values are:

Trace Mask Value	Description
0x00000001	Critical errors
0x00000002	Noncritical errors
0x00000004	Warnings
0x00000008	Informational messages
0x00000010	Data received via interprocess communication
0x00000020	Data sent via interprocess communication
0x00000040	Insertion of queue items
0x00000080	Removal of queue items
0x00000100	Creation of objects (Do not enable unless instructed to do so by HP support. It could significantly affect the performance of the server.)
0x00000200	Destruction of objects (Do not enable unless instructed to do so by HP support. It could significantly affect the performance of the server.)
0x00000400	Internal object interface calls
0x00000800	External low-level object interface calls
0x00001000	Calls from external clients to the server low-level object interface
0x00002000	External high-level object interface calls
0x00004000	Calls from external clients to the server high-level object interface
0x00008000	Items added to cache
0x00010000	Items removed from cache
0x00020000	Audit checks
0x00040000	Data received from management applications
0x00080000	Data sent to management applications
0x00100000	Thread creation
0x00200000	Thread destruction
0x00400000	Timer activity
0x00800000	General socket operations
0x01000000	Data received via UDP
0x02000000	Data sent via UDP
0x04000000	Data received via TCP
0x08000000	Data sent via TCP

Value Range: 0x00000000 - 0x0FFFFFFF

Default Value: 0x00000003 (critical and noncritical errors)

Example: `Mask = 0x0F80003F`

RecordLimit

Description: Defines the maximum number of trace records to capture, where 0 = no limit. Once this limit is reached, no further trace records are captured unless Wrap is True.

Value Range: 0 through 2147483647, where 0 = no limit

Default Value: 1000

Example: `RecordLimit = 2500`

Wrap

Description: If True, the server trace wraps when the record limit is reached. Whenever a new trace record is captured, the oldest trace record is deleted.

Value Range: True or False

Default Value: False

Example: Wrap = True

Sample Configuration File

The ASAP Hybrid for Linux installation archive contains a sample configuration file named **asap.conf.sample**. You can use this file as a basis for creating a configuration for your specific environment. At minimum, you will need to provide an asap.conf file containing the RemoteIPAddress value of the NonStop server to which the Linux system will connect.

The **asap.conf.sample** file follows. All option settings are commented except RemoteIPAddress in the [Comm] section:

```
##############################################################################
#
# Configuration File:      asap.conf
# Creation Time:           2004-11-29 17:00:00
#
##############################################################################


##############################################################################
# Section: [Server]
##############################################################################
[Server]
# AuditInterval      60
# SampleInterval     60
# MaxDomains         1000
# HTMLStatus         False
# DomainCleanup      False
# FlushRegistrations False

##############################################################################
# Section: [Comm]
##############################################################################
[Comm]
RemoteIPAddress    modify.this.value
# RemotePort         6301
# MessageSize        512
# LocalMachine       MyMachineName
# LocalPort          6301
# Pacing             20

##############################################################################
# Section: [Logging]
##############################################################################
[Logging]
# FileName           asaplog.html
# Level              3
# DeleteOnClose      False
# Enabled            True

##############################################################################
# Section: [Tracing]
##############################################################################
[Tracing]
# FileName           asaptrace.html
# Wrap               True
# RecordLimit        1500
# Enabled            False
```

Dynamically Altering Configuration Options

You can make configuration changes to the ASAP Hybrid for Linux server while it is running. To do so, update the **asap.conf** file using any editor, and then send the server process a SIGHUP signal to cause it to reload configuration information. For example:

```
kill -s SIGHUP server PID
```

where *server PID* is the process ID of the ASAP Hybrid for Linux server process.

On receipt of the SIGHUP signal, the server process reloads the asap.conf file and applies any changed option settings.

Cleaning Up Domains Interactively

In general, applications that utilize the ASAP Hybrid API operate as follows:

- Register one or more domains
- Update domain data items, operational state, and so on
- Remove domains at shutdown or if the domains no longer need to be monitored

When the application finally removes a domain, the ASAP Hybrid for Linux server notifies NonStop ASAP. NonStop ASAP then cleans up domain information and sends an acknowledgement back to the ASAP Hybrid for Linux server, which in turn cleans up its information on the domain. This approach ensures that the Linux side stays in sync with the rest of ASAP.

In order for this removal and acknowledgement scheme to function as intended, several conditions must be met:

- The application must remove the domain before exiting. If an application does not remove a domain prior to exiting (or failing), the ASAP Hybrid for Linux server detects that the application has stopped and marks the domain as down. ASAP on the host is notified of this and reports the down domain via its standard reporting and alerting mechanisms. Thus operators are immediately notified if a Linux client application fails. The domain continues to be marked as down until the client application restarts and reregisters the domain.
- A communication link must exist between the Linux machine and the NonStop host. If the link does not exist, the ASAP Hybrid for Linux server cannot notify NonStop ASAP that a domain has been removed. In this case, the server keeps the removal pending until it successfully notifies NonStop ASAP that the domain has been removed.
- ASAP must be running on the NonStop host. If it is not running, the ASAP Hybrid for Linux server cannot notify NonStop ASAP that a domain has been removed, and it keeps the removal pending until such notification can take place.

If any of these conditions are not met, domains cannot be removed and/or cleaned up, resulting in the presence of unused domains on the NonStop host, on the Linux system, or both.

To address this issue, ASAP provides several mechanisms for removing and cleaning up domains interactively: the host-based ASAP command interpreter, use of Linux signals, and the Linux-based asapremove utility.

Removing Down Domains Using the ASAP CI

If an application stops or fails prior to removing a domain, and that application will not be restarted or will not reregister that domain, the domain must be removed interactively. The preferred mechanism for doing so is to use the ASAP CI's STATUS PROXY $*name* REMOVE command. This command allows ASAP on the host to remove the domains in question and notifies ASAP Hybrid for Linux to remove the domains as well, thereby keeping the host and Linux sides synchronized. For more information on this command, see ASAP Hybrid Commands on page 2-8.

Removing Down Domains Using Linux Signals

If the ASAP CI cannot be used to remove one or more down domains, a second approach is to use Linux signals. Sending a Linux SIGUSR1 signal to the ASAP Hybrid for Linux server causes it to remove all down domains on that Linux system. For example:

```
kill -s SIGUSR1 server PID
```

where *server PID* is the process ID of the ASAP Hybrid for Linux server process.

The down domains are marked as removed, and notifications of the removals are sent to the host. Once the host acknowledges the removals, the Linux server cleans up the removed domains.

Like the CI, this mechanism ensures that the Linux and NonStop sides remain in sync. However, if the communication link to the NonStop system is down, or ASAP is not running on that system, the removals remain pending on the Linux system until communication is restored or ASAP is restarted.

Note. Sending a SIGUSR1 signal to the ASAP Hybrid Linux server causes **all** down domains to be removed. There is no means for picking and choosing which down domains to remove if you use this approach.

Removing Down Domains Using the asapremove Utility

The asapremove utility supplied with ASAP Hybrid for Linux is intended to be used to remove individual domains that are down. It works by accepting a domain name, registering that domain, and then immediately removing it. Like the other methods described previously, it ensures that the host and Linux sides remain synchronized. If the Linux server cannot notify the host that a domain has been removed, the removal remains pending until the notification can be delivered.

By default the asapremove utility is located in the **bin** directory that contains the ASAP Hybrid for Linux server. The executable file name is **asapremove**. The command syntax is:

```
./asapremove [ domain name [ domain name …]]
```

where *domain name* specifies a complete domain name currently registered on the Linux system.

If one or more domain names are specified on the command line, asapremove attempts to remove each and then exits. If no domain names are supplied on the command line, the utility runs in interactive mode and prompts for the domain names to be removed.

Note. Most ASAP domain names contain backslash (\) characters. In interactive mode, these characters do **not** need to be escaped by preceding them with an additional backslash. The asapremove utility does not give any special meaning to backslash characters though many Linux-based tools and utilities do. Thus, for example, if you want to remove the domain My\DomainName, specify to asapremove in interactive mode the value My\DomainName, not My\\DomainName. However, if you specify domain names on the asapremove command line, you might need to precede each embedded backslash with an additional backslash character. For example, to remove the domain name My\DomainName:

```
./asapremove My\\DomainName
```

Most Linux command shells interpret backslashes as special escape characters, so two are needed to indicate a single backslash.

Cleaning Up Removed Domains Using Linux Signals

If the communication link to the NonStop host is down, or ASAP is not running on the host, all domain removals performed on the Linux system remain pending until communication is restored, ASAP is started, and the ASAP Hybrid for Linux server can notify the host that the removals have occurred.

In some circumstances, leaving the removals as pending could become a hindrance, particularly if the communication link is down for a prolonged period. To force the ASAP Hybrid for Linux server to clean up these removed domains immediately, you can send the server process a SIGUSR2 signal:

```
kill -s SIGUSR2 server PID
```

where *server PID* is the process ID of the ASAP Hybrid for Linux server process.

This command causes the server to immediately clean up all removed domains, without requiring that the NonStop host be notified. This in turn frees resources on the Linux system and also allows a client application to reregister a domain that was previously removed.

The host is not notified that the removed domains have been cleaned up, so you might need to manually remove them on the host using the ASAP CI. For more information on how to do so, see ASAP Hybrid Commands on page 2-8.

Recovery

ASAP Hybrid for Linux incorporates mechanisms to prevent the loss of metric data in the event of a failure of either a client application or the ASAP Hybrid for Linux server itself.

Client Recovery

If a client application stops or fails for any reason, without first removing all domains it has registered, the data for those domains is preserved by the Hybrid for Linux server. When the application is restarted and reregisters the domains, it is reconnected to the preserved domain data. Application failures do not result in the loss of valuable metric information. In addition, this feature permits applications to be stopped and restarted when necessary without requiring that metric values be reset. This feature can be particularly useful if an application needs to be taken down temporarily in order to make a configuration change or to install a new version of an application component.

If an application stops or fails without removing all domains, those domains are marked as down by the Hybrid Linux server, and appropriate alerts are issued by NonStop ASAP.

ASAP Hybrid for Linux Server Recovery

If the ASAP Hybrid for Linux server fails or is stopped for any reason, all domain data remains active. Client applications can continue to update data item values, set the operational state of domains, and even remove domains while the server is down. The only thing applications cannot do in this case is register new domains because registration requires that the server be running. When the server is restarted, it automatically rediscovers all active domains and begins communicating that data to NonStop ASAP.

This feature lets client applications continue to collect metric data even if the server is not running and also allows the server to be upgraded to a new version without impacting running applications.

ASAP Hybrid for Linux Server Output

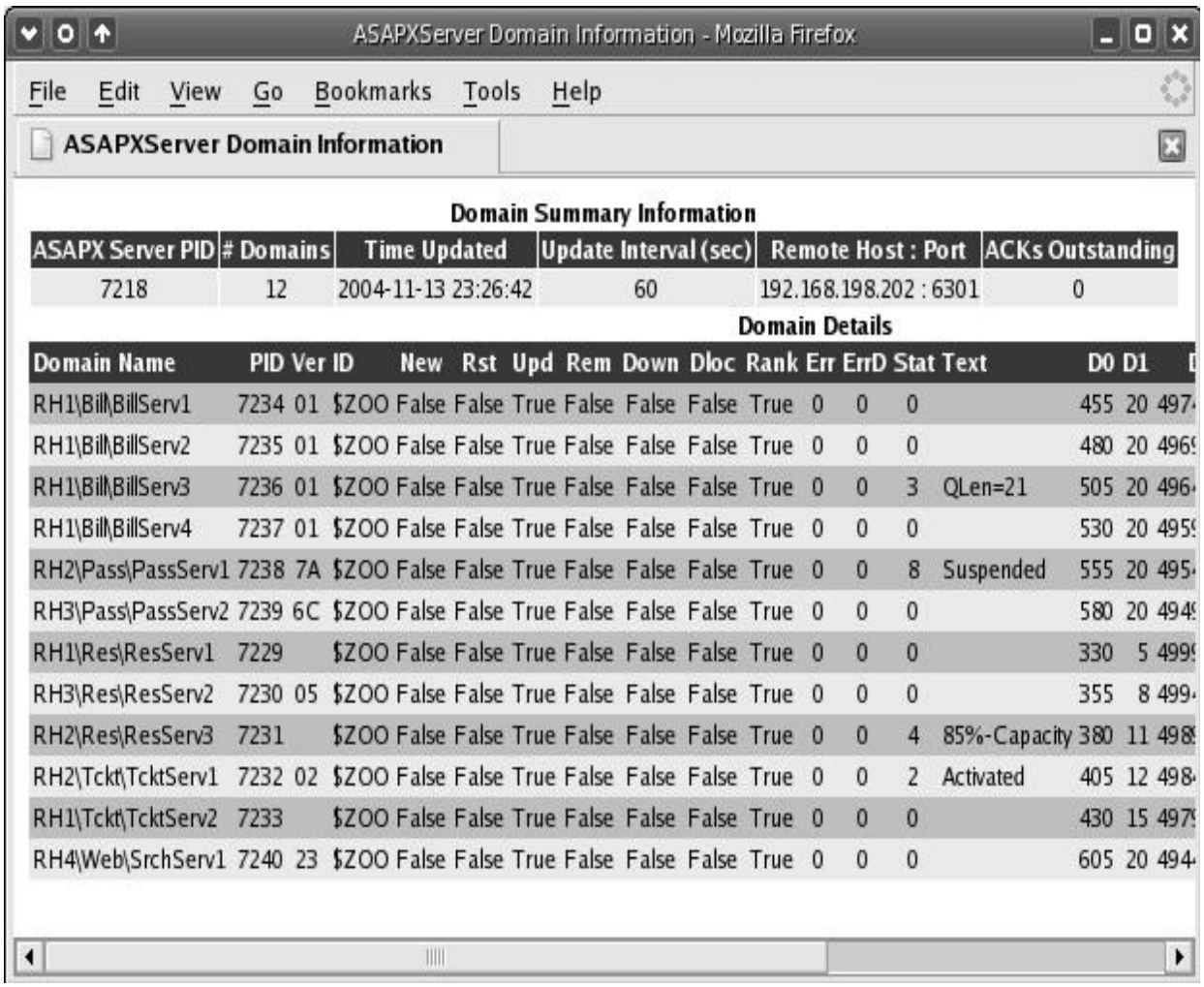
The ASAP Hybrid for Linux server generates several types of output:

- The server writes messages to the Linux system log as necessary. In general the server attempts to be judicious about the use of this log, and only writes messages at startup, shutdown, and if significant errors are encountered while the server is running. The ASAP Hybrid for Linux API library also writes messages to the Linux system log if it encounters a major error.
- The server writes messages to its own log file if logging is enabled via the **asap.conf** file. The default name for the log file is **asaplog.html**, and the file is located in the same directory as the server executable. This log contains varying

amounts of detail based on the Level setting defined in the configuration file. For more information on specifying logging settings, see Logging Options on page 3-7.

- The server writes trace information to a trace output file if tracing is enabled via the **asap.conf** file. The default name for the trace file is **asaptrace.html**, and the file is located in the same directory as the server executable. As with the log file, the amount of detail contained in the trace file can vary. In general, the log file contains higher-level information on errors and other significant events, and the trace file contains internal details on the server and the data it exchanges with client applications. For more information on specifying tracing settings, see Tracing Options on page 3-10.
- If HTMLStatus is set to True in the **asap.conf** file, the server creates a domain report in HTML format at every sample interval. This file is named **asapdomains.html** and is located in the same directory as the server executable. This file is valuable for debugging while you are developing client applications because it provides details on all domains and data items within the local ASAP Hybrid environment.

This example shows a sample report produced if this option is enabled:



VST002.vsd

These columns appear in the report:

Column Name	**Description** (page 1 of 2)
ASAPX Server PID	PID of the ASAP Hybrid for Linux server process
# Domains	Number of domains registered
Time Updated	The last time the domain sample was performed
Update Interval	The update interval, in seconds
Remote Host : Port	The IP address and port of the NonStop server to which the Linux system is connected
ACKs Outstanding	Acknowledgement requests pending on this Linux machine; typically this value should be 0 or 1. If 3 or more, the remote host is probably unreachable
Domain Name	The name of the registered domain
PID	The PID of the client application that registered the domain
Ver	The version registered by the client for that domain

Column Name	Description (page 2 of 2)
ID	The ASAP ID registered by the client for that domain
New	True if the domain just registered this sample; false otherwise
Rst	True if the domain restarted this sample; false otherwise
Upd	True if any domain updates have occurred this sample (via calls to any of the **asap_update**, **asap_control**, or **asap_opstate** procedures); false if not
Rem	True if the domain has been removed; false if not. If a domain has been removed, it no longer appears in the domain report once the removal is acknowledged by the NonStop server
Down	True if the domain is down; false if not. For the purposes of this report, a domain is considered down if the client application that registered the domain has stopped or failed without calling **asap_remove**
Dloc	True if the domain will be deallocated on removal; false if not
Rank	True if ranking is enabled for the domain; false if not
Err	Last NonStop server error for the domain
ErrD	Last NonStop server error detail for the domain
Stat	Status of the domain, set via **asap_opstate** calls
Text	Status text for the domain, set via **asap_opstate** calls
D0 - D11	Data items for the domain, set via **asap_update** calls

The HTML report is generated with an automatic refresh tag, so the browser automatically reloads the report contents on a regular basis.

In addition, various fields are color-coded to indicate significant conditions. For example, down domains are highlighted in red, domains that are updated in a given interval are highlighted in green, and so on.

4

The ASAP Hybrid for Linux API

The ASAP Hybrid for Linux API Library allows Linux applications to interface to the ASAP subsystem. Applications utilize the API to register domains and update metric counters in shared memory. No other interaction with ASAP is required. The ASAP Hybrid for Linux server extracts domain and counter data from shared memory and passes it to NonStop ASAP where it is processed along with all other ASAP data.

Key features of the ASAP Hybrid for Linux API include:

- Nonblocking API

  Most API calls are nonblocking, so application performance and responsiveness are not affected by instrumenting for ASAP.

- Low overhead

  Shared memory is used for storing all domain information, and the API calls access this memory directly. Virtually no overhead is involved in using the ASAP API.

- Support for multithreaded or single-threaded applications

  Thread-safe and non-thread-safe variants of each API procedure (except asap_register) are provided. The thread-safe versions permit different threads of the same process to share ASAP counter space, and the non-thread-safe versions allow single-threaded applications to call ASAP API procedures without incurring any incremental overhead associated with support for multiple threads.

- Built-in recoverability

  The ASAP Hybrid for Linux subsystem retains all domain and metric data if an application stops or fails prior to removing a domain. This capability allows the application to restart and re-register that domain without losing any accumulated metric data.

- Support for *x*86 and AMD64 platforms

  Both 32-bit and 64-bit versions of the API library are provided, allowing you to build either 32-bit applications for the x86 platform or 64-bit applications for the AMD64 platform.

- Multiple libraries provided

  Both static and shared library versions are provided. You can choose to use the one that makes the most sense for your environment.

Developing Applications Using the ASAP Hybrid API

To use the ASAP Hybrid for Linux API from a client application:

1. Define client application entities using the ASAP Entity Definition Language (EDL), and load these definitions into your ASAP environment. For more information on EDL, see the *ASAP Extension Manual* and the *ASAP Client Manual*.
2. Include the **asapx.h** header file in your application source. For example:

   ```
   #include "asapx.h"
   ```

3. Modify your make file or compiler commands to search for the **asapx.h** header file in the **include** directory created when ASAP Hybrid for Linux was installed. For more information on installation locations, see Installing ASAP Hybrid for Linux on page 3-1.
4. Modify your make file or compiler/linker commands to search for the appropriate API library files in the **lib** directory created when ASAP Hybrid for Linux was installed. For more information on installation locations, see Installing ASAP Hybrid for Linux on page 3-1. For details on linking to the various library versions, see ASAP Hybrid for Linux API Libraries on page 4-3.
5. Modify your make file or compiler/linker commands to link to the Linux Pthreads library (**libpthread**). Because the ASAP Hybrid for Linux API utilizes Pthreads to support thread-safe callers, the Pthreads library must be available at all times. This step is required even if you are developing a single-threaded application and are not using thread-safe ASAP API calls. However, in that case, no Pthreads library procedures would actually be called at run time by the ASAP API, so no extra overhead would be introduced into the calling application.
6. Modify your application source to call the necessary ASAP Hybrid for Linux API procedures. For a description of the various API calls, see API Details on page 4-5. The basic rules follow:
   - Register at least one domain using the asap_register procedure.
   - Optionally update data items for the domains. Single-threaded applications would use the asap_update, asap_updatelist procedures, or both. Multithreaded applications would use the asap_update_ts, asap_updatelist_ts procedures, or both.
   - Optionally update the operational state and text for the domains. Single-threaded applications would use the asap_opstate procedure, and multithreaded applications would use the asap_opstate_ts procedure.
   - Optionally activate and deactivate ranking for the domains as needed. Single-threaded applications would use the asap_control procedure, and multithreaded applications would use the asap_control_ts procedure.

- Remove all registered domains prior to shutting down. Single-threaded applications would use the asap_remove procedure, and multithreaded applications would use the asap_remove_ts procedure.

ASAP Hybrid for Linux API Libraries

ASAP Hybrid for Linux provides libraries for both 32-bit x86 and 64-bit AMD64 platforms, with static and shared variants of each. A given application must link to one of these libraries, depending both on the platform you are developing for and the system management strategy employed for your particular environment.

To establish which library to use, you must first determine the target platform for your application. If you are developing a 32-bit application, you need to link to a 32-bit API library. If you are developing a 64-bit application, you need to link to the 64-bit API library.

Once you determine the platform, you must decide whether to use the static library or the shared library for that platform. Each type of library has its advantages and disadvantages, depending on how you typically manage your environment:

- The advantage of the static library is that it encapsulates ASAP functionality within the given executable. This approach eliminates potential versioning issues and makes it easy to move application executables from system to system because no additional ASAP libraries are necessary. (However, the ASAP Hybrid for Linux server must reside on each of those systems in order to provide services to the client applications.) The disadvantage of using the static library is that all client applications must be recompiled in order to use a new version of the library, which can be problematic if a large number of applications are affected.
- The advantage of the shared library is that client applications do not have to be recompiled in order to use new versions of the library. All new features are immediately available to any applications that use the shared library. The disadvantage of the shared library is that management of applications and libraries becomes more complicated because the applications and libraries are deployed independently of each other. This lack of coordinated deployment can lead to issues with file placement, library search paths, and versioning.

After deciding which type of library to use, you must link the application to the correct library file. Use the following table to determine the library file name for your platform and environment:

Platform \ Library Type	Static	Shared
32-bit	libasapx.a	libasapx.so
64-bit	libasapx64.a	libasapx64.so

If you choose to use a shared library and do not copy the library files to the standard **/usr/lib** location, you need to modify the Linux dynamic loader's search path so that it can locate the library at run time. Update the **/etc/ld.so.conf** file, or modify the **LD_LIBRARY_PATH** environment variable, to reference the directory containing the

library files. By default, this directory is **lib** in the original ASAP Hybrid for Linux installation directory. For more information on default installation locations, see Installing ASAP Hybrid for Linux on page 3-1.

Differences from NonStop ASAPX

Although the ASAP Hybrid for Linux API is nearly identical to the ASAPX API on the NonStop server, there are some differences. In general, these changes were made to tailor the API to the Linux environment. For example, procedure names are slightly different in order to follow standard Linux practices, in some cases the number or types of parameters have changed slightly, and in every case the range of returned values varies from the range used in ASAPX because the error scenarios on Linux differ widely from those on the NonStop server. However, the general purpose of each procedure has not changed, and there is a one-to-one relationship between ASAPX API procedures and ASAP Hybrid for Linux API procedures.

For details on the differences between the Linux and NonStop versions of each procedure, see the description of that procedure in the API Details section.

Threading

To support both single-threaded and multithreaded client applications, two procedures are frequently supplied for a given purpose: a standard version and a thread-safe version. The thread-safe version is named the same as the standard version, with ts appended to the name. For example, **asap_update** is a standard call, and **asap_update_ts** is the thread-safe variant.

The main difference between the standard version of a call and the thread-safe version of a call is that the thread-safe version automatically protects access to shared counter space via a Pthreads mutex. Multiple threads can attempt to update the same metrics simultaneously, and the ASAP Hybrid library ensures that such updates are serialized.

In general, you should use the thread-safe procedures only if you are truly sharing ASAP data items between threads., The thread-safe variants can introduce some blocking into call processing while waiting for a mutex to free and also incur extra system overhead in order to utilize the mutex.

Note. There is no risk of deadlock in using the thread-safe API procedures. Because the ASAP Hybrid library manages all mutex access, any mutex-related resources are freed by the time the ASAP API procedure returns control to the calling thread. Furthermore, the API procedures themselves never own or wait on more than one resource at a time. There is no opportunity for the thread or the ASAP API to deadlock while waiting for an ASAP-related resource.

API Details

This section describes each of the ASAP Hybrid for Linux API procedure calls in detail.

asap_register

Register a new domain with ASAP. This procedure must be called once for each domain before you attempt to update data items for that domain. Once a domain has been successfully registered, the application can use the other ASAP Hybrid library procedures to update information for the domain, including data items and operational state and text.

An application can register as many domains as needed, but each domain name must be unique within ASAP. The application must also maintain the shared memory handle returned for each domain in order to be able to update and eventually remove that domain.

Declaration

```
short asap_register (
   const char       *ptr_domain_name,           // IN REQUIRED
                                                // App domain name being
                                                // registered
   void            **ptr_handle,                // OUT REQUIRED
                                                // Shared memory handle for
                                                // this domain
   short            *ptr_error_detail = NULL,   // OUT OPTIONAL
                                                // Detailed error number, if
                                                // any
   const char       *ptr_asap_id = NULL,        // IN OPTIONAL
                                                // ASAP ID to register into
   const bool        rank_enabled_ind = true,   // IN OPTIONAL
                                                // Rank enabled flag
   const bool        concat_ind = true,         // IN OPTIONAL
                                                // PID concatenation
                                                // indicator
   const int         io_timeout = 2000,         // IN OPTIONAL
                                                // Registration timeout value
   const char       *ptr_version = NULL         // IN OPTIONAL
                                                // Process/application
                                                // version ID
   );
```

Parameter Descriptions

const char *ptr_domain_name

A unique identifier for the application domain. The domain name is a logical representation of the application work being performed. Domain names must follow these rules:

- They can contain no more than five logical, hierarchical levels, using the backslash (\) character as a level separator. The name cannot begin with a backslash.

- The ASAP Entity name for this domain must be the leftmost level of the domain name.
- The entity must be defined in ASAP EDL and must be loaded into the copy of ASAP referenced by the ASAP ID parameter on the NonStop servers configured to accept data from the Linux system.
- The name must be null-terminated.
- The name cannot exceed ASAP_MAX_DOMAIN_NAME_LENGTH bytes in length, not including the terminating null.
- The name cannot contain a space, quote, comma, colon, semicolon, or asterisk.
- All domain names must be unique. ASAP can optionally append the process ID of the calling process as the final level of the domain name if desired. However, such IDs are generic and not repeatable, so the current process will probably not be assigned the same ID the next time it is run. Thus an application-specific name is preferred. You can use the concat_ind parameter to specify whether or not ASAP should automatically append the process ID to the domain name.

This parameter is required.

void **ptr_handle

A handle for the shared memory allocated to this domain. This handle must be maintained and passed to all subsequent calls to the ASAP API when you update this domain. This parameter is required.

short *ptr_error_detail

A detailed error code, if any. This parameter is optional. The default value is NULL, which means no detailed error code is returned.

const char *ptr_asap_id

The null-terminated, up to ASAP_MAX_ID_LENGTH character ID of the ASAP environment on the NonStop server into which this domain will register. This parameter is optional. The default value is NULL, which causes ASAP to use the default ASAP ID configured on the NonStop server. If specified, the ID must begin with a dollar sign ($) and must be at least 2 bytes long ($ + at least 1 character). In addition, the first character following $ must be alphabetic, and any remaining characters must be alphanumeric.

const bool rank_enabled_ind

Flag to enable or disable ranking when the domain is registered. This parameter is optional. The default value is true, which means ranking is enabled by default.

const bool concat_ind

Flag indicating whether ASAP should automatically append the current process ID (PID), in text form, to the end of the domain name. This parameter is optional. The default value is true, which means the PID is appended as the last level of the domain name.

const int io_timeout

Registration timeout in milliseconds. A value of 0 indicates no timeout (wait indefinitely). This parameter is optional. The default value is 2,000 (2 seconds).

const char *ptr_version

Null-terminated, printable ASCII-text version number for the application, which can be up to ASAP_MAX_APP_VERSION_LENGTH bytes in length, not including the null terminator. This parameter is optional. The default value is NULL, which means there is no application version.

Return Values

ASAP_ERROR_NONE

No error. Registration was successful.

ASAP_ERROR_MEMORY

Could not allocate memory to perform the operation.

ASAP_ERROR_INVALID_PARAM

A parameter was invalid. If specified, *ptr_error_detail contains the incorrect parameter number (first param = 1).

ASAP_ERROR_INVALID_DOMAIN_NAME

The specified domain name was invalid. Possibly the name is too long (exceeds ASAP_MAX_DOMAIN_NAME_LENGTH), contains too many levels (exceeds ASAP_MAX_DOMAIN_NAME_LEVELS), contains an invalid character, and so on.

ASAP_ERROR_INVALID_ID

The specified ASAP ID exceeded ASAP_MAX_ID_LENGTH characters in length, did not begin with a dollar sign ($), did not contain any characters after $, the first character after $ was not alphabetic, or a subsequent character was not alphanumeric.

ASAP_ERROR_INVALID_APP_VERSION

The specified application version either exceeded ASAP_MAX_APP_VERSION_LENGTH characters or contained non-ASCII printable characters.

ASAP_ERROR_NO_SERVER

The ASAP Hybrid for Linux server is not running or could not service the request.

ASAP_ERROR_LIBRARY_PIPE

The pipe file for the library could not be created or could not be opened.

ASAP_ERROR_DUPLICATE_DOMAIN

Duplicate domain name.

ASAP_ERROR_TOO_MANY_DOMAINS

Maximum number of domains already registered.

ASAP_ERROR_SHARED_SEGMENT

Could not allocate or attach the shared segment.

ASAP_ERROR_SERVER_RESOURCE

Could not complete the operation because of lack of server resources.

ASAP_ERROR_REMOVED

The supplied domain exists already but has been removed. Once cleanup occurs, the domain can be registered again.

ASAP_ERROR_MUTEX

The mutex for the domain could not be allocated. Thread-safe API calls for this domain (that is, asap_update_ts, asap_remove_ts, and so on) are not available. All such calls return the error ASAP_ERROR_MUTEX and do not perform any operation. However, the registration was successful otherwise, and the client application can use the standard ASAP API calls.

Differences from the NonStop ASAPX API

- The domain name parameter is supplied as a standard C null-terminated string, and the domain name length parameter (domain^name^len) has been eliminated.
- The segment^id parameter has been eliminated.
- The segment^base parameter has been eliminated.
- The ASAP ID parameter is supplied as a standard C null-terminated string, and the ID length parameter (id^len) has been eliminated.
- The flags parameter has been replaced by two separate Boolean parameters: rank_enabled_ind, which is used to specify whether ranking is initially enabled for the domain; and concat_ind, which is used to indicate whether the PID for the calling process should be appended to the domain name. There is no longer an option to specify whether or not domains can perform replace update operations on nonconstant data items. These operations are always permitted.
- The timeout value is supplied in milliseconds instead of in hundredths of seconds.
- The application version parameter is supplied as a standard C null-terminated string.

- A different range of values and possible errors is returned.

Example

```
void *Handle;                                          // Handle from
                                                       // asap_register
short ErrorDetail;                                     // Error detail from
                                                       // call
short ReturnValue;                                     // Return value from
                                                       // call

ReturnValue = ::asap_register("MyApp\\MyDomain",       // Supply the domain
                                                       // name
                              &Handle,                 // Returned handle
                              &ErrorDetail,            // Returned error
                                                       // information
                              "$zoo",                  // ASAP ID is $zoo
                              true,                    // Enable ranking
                              true,                    // Concat PID to domain
                                                       // name
                              3000,                    // Wait up to 3 seconds
                              "01");                   // Application version
                                                       // 01
```

asap_remove and asap_remove_ts

Removes a domain from the list of domains monitored by ASAP. This procedure must be called whenever ASAP is to stop monitoring a given domain. This procedure affects active monitoring only. It does not remove historical data from the ASAP database.

When a domain is removed, ASAP writes a final record for the domain to the ASAP database and then cleans up all resources associated with that domain. The domain is not monitored from that point forward unless it is reregistered by the application. For online applications, asap_remove should generally need to be called only when the application is shut down. The exception is for domains that are more transient in nature, which might be registered, update values, and then be removed when a given unit of work has been completed.

Note. A record is written to the ASAP database for each registered domain every sample interval. If a domain has not been updated in an interval, it is marked as inactive by ASAP, and alerts might optionally be generated. To avoid spurious warnings of this type, applications should remove domains that no longer need to be monitored. On the other hand, if a domain is simply entering a dormant state for some period of time, the asap_control API procedure can be used to temporarily disable ranking for that domain.

Declaration

```
short asap_remove (
   void             *handle,                    // IN REQUIRED
                                                // Shared memory handle for
                                                // this domain
   short            *ptr_error_detail = NULL,   // OUT OPTIONAL
                                                // Detailed error number, if
                                                // any
   const bool        deallocate_ind = true      // IN OPTIONAL
                                                // Deallocation indicator
   );

short asap_remove_ts (
   void             *handle,                    // IN REQUIRED
                                                // Shared memory handle for
                                                // this domain
   short            *ptr_error_detail = NULL,   // OUT OPTIONAL
                                                // Detailed error number, if
                                                // any
   const bool        deallocate_ind = true      // IN OPTIONAL
                                                // Deallocation indicator
   );
```

Parameter Descriptions

void *handle

The handle returned from the call to asap_register. This parameter is required.

short *ptr_error_detail

A detailed error code, if any. This parameter is optional. The default value is NULL, which means no detailed error code is returned.

const bool deallocate_ind

Flag to indicate whether resources should be deallocated for this domain. If the domain is being removed as part of application or process shut down, resources should be deallocated. If the application is removing a domain but planning to reregister it in the near future, resources can remain. This parameter is optional. The default value is true, which means that resources will be deallocated.

Return Values

ASAP_ERROR_NONE

No error. Removal was successful.

ASAP_ERROR_INVALID_PARAM

A parameter was invalid. If specified, *ptr_error_detail contains the incorrect parameter number (first param = 1).

ASAP_ERROR_SHARED_SEGMENT

Could not detach the shared segment.

ASAP_ERROR_MUTEX (applies to asap_remove_ts only)

No mutex was allocated for this domain, or there was an error accessing the mutex. The domain is not removed. To remove it, the application must use the standard asap_remove call.

Differences from the NonStop ASAPX API

- The segment^id parameter has been eliminated.
- The flags parameter has been replaced by the Boolean parameter deallocate_ind, but this parameter still has the same purpose: to indicate whether or not resources should be deallocated for the domain.
- A different range of values and possible errors is returned.

Example

```
short ErrorDetail;                              // Error detail from
                                                // call

short ReturnValue;                              // Return value from
                                                // call


ReturnValue = ::asap_remove(Handle,             // Handle from
                                                // asap_register
                            &ErrorDetail,       // Returned error
                                                // information
                            true);              // Deallocate domain
```

asap_update and asap_update_ts

Update a single data item for a domain, using the supplied value and math (operation type).

Data items are defined in the EDL for the application. This EDL must be loaded into the running ASAP environment.

Declaration

```
short asap_update (
   void             *handle,                    // IN REQUIRED
                                                // Shared memory handle for
                                                // this domain
   const short       dataitem,                  // IN REQUIRED
                                                // Index of the data item
                                                // being updated
   const long long   value = 1,                 // IN OPTIONAL
                                                // Value used to update the
                                                // data item
   const short       math = ASAP_MATH_ADD,      // IN OPTIONAL
                                                // Math type; see ASAP_MATH
                                                // constants
   short            *ptr_error_detail = NULL    // OUT OPTIONAL
                                                // Detailed error number, if
                                                // any
   );

short asap_update_ts (
   void             *handle,                    // IN REQUIRED
                                                // Shared memory handle for
                                                // this domain
   const short       dataitem,                  // IN REQUIRED
                                                // Index of the data item
                                                // being updated
   const long long   value = 1,                 // IN REQUIRED
                                                // Value used to update the
                                                // data item
   const short       math = ASAP_MATH_ADD,      // IN OPTIONAL
                                                // Math type; see ASAP_MATH
                                                // constants
   short            *ptr_error_detail = NULL    // OUT OPTIONAL
                                                // Detailed error number, if
                                                // any
   );
```

Parameter Descriptions

void *handle

The handle returned from the call to asap_register. This parameter is required.

const short dataitem

The numeric index between 0 and (ASAP_MAX_DATAITEMS - 1) of the item to update. This parameter is one of the in-memory data items as defined in EDL for this ASAP domain. This parameter is required.

const long long value

The update value, in 64-bit binary form, representing a positive integer value or an 8-byte ASCII string. This parameter is optional. The default value is 1.

const short math

The type of math to be done when applying the new value to the in-memory data item. Possible values are given by the ASAP_MATH constants defined in the asapx.h header file. This parameter is optional. The default value is ASAP_MATH_ADD.

short *ptr_error_detail

A detailed error code, if any. This parameter is optional. The default value is NULL, which means no detailed error code is returned.

Return Values

ASAP_ERROR_NONE

No error. Update was successful.

ASAP_ERROR_INVALID_PARAM

A parameter was invalid. If specified, *ptr_error_detail contains the incorrect parameter number (first param = 1).

ASAP_ERROR_REMOVED

The domain has been removed. Once cleanup occurs, the domain can be reregistered.

ASAP_ERROR_HOST_DOMAIN

There is a problem with this domain on the NonStop server. Updates continue to occur locally on the Linux system and are sent to the server. However, the server might not be able to update or display this data. *ptr_error_detail contains the NonStop server error number.

ASAP_ERROR_MUTEX (applies to asap_update_ts only)

No mutex was allocated for this domain, or there was an error accessing the mutex. The update is not performed. To update data, the application must use the standard asap_update call.

Differences From the NonStop ASAPX API

A different range of values and possible errors is returned.

Example

```
short ErrorDetail;                                  // Error detail from
                                                    // call
short ReturnValue;                                  // Return value from
                                                    // call

ReturnValue = ::asap_update(Handle,                 // Handle from
                                                    // asap_register
                            11,                     // Data item 11
                            1234LL,                 // Value = 1234
                            ASAP_MATH_ADD,          // Add it to the data
                                                    // item
                            &ErrorDetail);          // Returned error
                                                    // information
```

asap_updatelist and asap_updatelist_ts

Update a list of data items (up to ASAP_MAX_DATAITEMS - 1) for a single domain.

Data items are defined in the EDL for the application. This EDL must be loaded into the running ASAP environment.

Declaration

```
short asap_updatelist (
   void             *handle,                    // IN REQUIRED
                                                // Shared memory handle for
                                                // this domain
   const short       count,                     // IN REQUIRED
                                                // Number of items in the
                                                // list
   struct asap_list *ptr_asap_list,             // IN REQUIRED
                                                // Pointer to list of items
   short            *ptr_error_detail = NULL    // OUT OPTIONAL
                                                // Detailed error number, if
                                                // any
   );

short asap_updatelist_ts (
   void             *handle,                    // IN REQUIRED
                                                // Shared memory handle for
                                                // this domain
   const short       count,                     // IN REQUIRED
                                                // Number of items in the
                                                // list
   struct asap_list *ptr_asap_list,             // IN REQUIRED
                                                // Pointer to list of items
   short            *ptr_error_detail = NULL    // OUT OPTIONAL
                                                // Detailed error number, if
                                                // any
   );
Parameter Descriptions
```

void *handle

The handle returned from the call to asap_register. This parameter is required.

const short count

The number of asap_list elements to be processed. This parameter is required.

struct asap_list *ptr_asap_list

An array indicating the data item, value, and math parameters for each item to be updated. These values are defined in a structure of type asap_list, which contains a maximum of ASAP_MAX_DATAITEMS elements. This parameter is required.

short *ptr_error_detail

A detailed error code, if any. This parameter is optional. The default value is NULL, which means no detailed error code is returned.

Return Values

ASAP_ERROR_NONE

No error. Update was successful.

ASAP_ERROR_INVALID_PARAM

A parameter was invalid. If specified, *ptr_error_detail contains the incorrect parameter number (first param = 1). If one of the asap_triplet items in *ptr_asap_list is incorrect, the value in *ptr_error_detail is: (10 * (index in list + 1)) + one of the following: 0 if the data_item is incorrect, 1 if the val is incorrect, or 2 if the math is incorrect. For example, if the first asap_triplet item in the list (index = 0) contains the values data_item = 3, val = 1, math = 5 (this is an invalid math value), the *ptr_error_detail value returned would be (10 * (0 + 1)) + 2 = 12. Looked at another way, the error values are defined as:

10 = first item in list, data_item invalid
11 = first item in list, val invalid
12 = first item in list, math invalid
20 = second item in list, data_item invalid
….

ASAP_ERROR_REMOVED

The domain has been removed. Once cleanup occurs, the domain can be reregistered.

ASAP_ERROR_HOST_DOMAIN

There is a problem with this domain on the NonStop server. Updates continue to occur locally on the Linux system and are sent to the server. However, the server might not be able to update or display this data. *ptr_error_detail contains the NonStop server error number.

ASAP_ERROR_MUTEX (applies to asap_updatelist_ts only)

No mutex was allocated for this domain, or there was an error accessing the mutex. The update is not performed. To update data, the application must use the standard asap_updatelist call.

Differences from the NonStop ASAPX API

A different range of values and possible errors is returned.

Example

```
short ErrorDetail;                                  // Error detail from
                                                    // call
short ReturnValue;                                  // Return value from
                                                    // call
struct asap_list List;                              // asap_list struct for
                                                    // call

List.idx[0].data_item = 5;                          // First data item is 5
List.idx[0].val       = 1234LL;                     // Value = 1234
List.idx[0].math      = ASAP_MATH_ADD;              // Add it to the data
                                                    // item
List.idx[1].data_item = 9;                          // Second data item is
                                                    // 9
List.idx[1].val       = 2LL;                        // Value = 2
List.idx[1].math      = ASAP_MATH_REPLACE_NUMBER;   // Replace the value in
                                                    // data item
ReturnValue = ::asap_updatelist(Handle,             // Handle from
                                                    // asap_register
                                2,                  // 2 items in the list
                                &List,              // Address of the
                                                    // asap_list struct
                                &ErrorDetail);      // Returned error
                                                    // information
```

asap_control and asap_control_ts

Activate or deactivate ranking for the domain. Deactivation is useful in the case of a domain that enters a dormant state for some time. ASAP sets the state of domain to exists, and no further alerts are issued for that domain until ranking is activated.

An application should call asap_control for a domain if the domain is not updating its data for a time but will become active again in the future. If a domain will never update its data again, the domain should be removed using asap_remove.

Declaration

```
short asap_control (
   void             *handle,                   // IN REQUIRED
                                               // Shared memory handle for
                                               // this domain
   const bool        rank_enabled_ind,         // IN REQUIRED
                                               // true = rank enable, false
                                               // = disable
   short            *ptr_error_detail = NULL   // OUT OPTIONAL
                                               // Detailed error number, if
                                               // any
   );

short asap_control_ts (
   void             *handle,                   // IN REQUIRED
                                               // Shared memory handle for
                                               // this domain
   const bool        rank_enabled_ind,         // IN REQUIRED
                                               // true = rank enable, false
                                               // = disable
   short            *ptr_error_detail = NULL   // OUT OPTIONAL
                                               // Detailed error number, if
                                               // any
```

```
);
```

Parameter Descriptions

void *handle

The handle returned from the call to asap_register. This parameter is required.

const bool rank_enabled_ind

Flag to enable or disable ranking; true = ranking enabled, false = ranking disabled. This parameter is required.

short *ptr_error_detail

A detailed error code, if any. This parameter is optional. The default value is NULL, which means no detailed error code is returned.

Return Values

ASAP_ERROR_NONE

No error. Operation was successful.

ASAP_ERROR_INVALID_PARAM

A parameter was invalid. If specified, *ptr_error_detail contains the incorrect parameter number (first param = 1).

ASAP_ERROR_REMOVED

The domain has been removed. Once cleanup occurs, the domain can be reregistered.

ASAP_ERROR_HOST_DOMAIN

There is a problem with this domain on the NonStop server. Updates continue to occur locally on the Linux system and are sent to the server. However, the server might not be able to update or display this data. *ptr_error_detail contains the NonStop server error number.

ASAP_ERROR_MUTEX (applies to asap_control_ts only)

No mutex was allocated for this domain, or there was an error accessing the mutex. The operation is not performed. To activate or deactivate ranking, the application must use the standard asap_control call.

Differences from the NonStop ASAPX API

- The flags parameter has been replaced by the Boolean parameter rank_enabled_ind, but this parameter still has the same purpose: to indicate whether ranking should be enabled or disabled.
- A different range of values and possible errors is returned.

Example

```
short ErrorDetail;                                  // Error detail from
                                                    // call
short ReturnValue;                                  // Return value from
                                                    // call

ReturnValue = ::asap_control(Handle,                // Handle from
                                                    // asap_register
                             false,                 // Disable ranking
                             &ErrorDetail);         // Returned error
                                                    // information
```

asap_opstate and asap_opstate_ts

Set the operational state for the domain, independent of the state of any of the domain's data items. Calling applications can use these procedures to explicitly specify the state of a given domain and associate a textual description with that state.

If asap_opstate is called for a given domain, the state specified overrides any state calculated by ASAP while analyzing the domain's metric values.

Declaration

```
short asap_opstate (
   void             *handle,                    // IN REQUIRED
                                                // Shared memory handle for
                                                // this domain
   const char       *ptr_status_text,           // IN REQUIRED
                                                // Operational status to
                                                // display
   const short       status_state,              // IN REQUIRED
                                                // Operational state; a valid
                                                // ASAP_STATE
   short            *ptr_error_detail = NULL    // OUT OPTIONAL
                                                // Detailed error number, if
                                                // any
   );

short asap_opstate_ts (
   void             *handle,                    // IN REQUIRED
                                                // Shared memory handle for
                                                // this domain
   const char       *ptr_status_text,           // IN REQUIRED
                                                // Operational status to
                                                // display
   const short       status_state,              // IN REQUIRED
                                                // Operational state; a valid
                                                // ASAP_STATE
   short            *ptr_error_detail = NULL    // OUT OPTIONAL
                                                // Detailed error number, if
                                                // any
  );
```

Parameter Descriptions

void *handle

The handle returned from the call to asap_register. This parameter is required.

const char *ptr_status_text

Null-terminated character string containing the status text to set. This parameter is displayed as the operational status of the domain, overriding any state derived from the domain's data item values. This string can be up to ASAP_MAX_STATUS_TEXT_LENGTH bytes in length, not including the null terminator, must not be blank, and can contain only ASCII-printable text. This parameter is required.

const short status_state

The ASAP OEM state to assign for this domain. This value can be from ASAP_STATE_EXISTS (object exists) through ASAP_STATE_DOWN (object is down). For ASAP_STATE values, see the asapx.h header file. This parameter is required.

short *ptr_error_detail

A detailed error code, if any. This parameter is optional. The default value is NULL, which means no detailed error code is returned.

Return Values

ASAP_ERROR_NONE

No error. Operation was successful.

ASAP_ERROR_INVALID_PARAM

A parameter was invalid. If specified, *ptr_error_detail contains the incorrect parameter number (first param = 1).

ASAP_ERROR_REMOVED

The domain has been removed. Once cleanup occurs, the domain can be reregistered.

ASAP_ERROR_HOST_DOMAIN

There is a problem with this domain on the NonStop server. Updates continue to occur locally on the Linux system and are sent to the server. However, the server might not be able to update or display this data. *ptr_error_detail contains the NonStop server error number.

ASAP_ERROR_MUTEX (applies to asap_opstate_ts only)

No mutex was allocated for this domain, or there was an error accessing the mutex. The operation is not performed. To set the operational state of the domain, the application must use the standard asap_opstate call.

Differences from the NonStop ASAPX API

- The operational text parameter is supplied as a standard C null-terminated string, and the operational text length parameter (optext^len) has been eliminated.
- A different range of values and possible errors is returned.

Example

```
short ErrorDetail;                                  // Error detail from
                                                    // call
short ReturnValue;                                  // Return value from
                                                    // call

ReturnValue = ::asap_opstate(Handle,                // Handle from
                                                    // asap_register
                             "Object down",         // Status text; 15
                                                    // bytes max
                             ASAP_STATE_CRITICAL,   // Critical state
                             &ErrorDetail);         // Returned error
                                                    // information
```

Sample Client Application

This small sample application for ASAP Hybrid for Linux takes a domain name that you specify, registers it, updates data items, and then removes the domain. It demonstrates the use of the asap_register, asap_update, asap_opstate, and asap_remove API procedures.

```
// ASAP Hybrid for Linux sample client application
//

// Include the asapx.h header file
#include "asapx.h"
#include <string.h>
#include <stdio.h>


int main(int argc, char *argv[])
{
   char  DomainName[ASAP_MAX_DOMAIN_NAME_LENGTH + 1];       // Domain name to
                                                            // register
   void *Handle;                                            // Handle from
                                                            // asap_register
   short ErrorDetail;                                       // Error detail
                                                            // from calls
   short ReturnValue;                                       // Return value
                                                            // from calls


   while (true)
   {
      // Get a domain name to register
      ::printf("Enter domain name, EXIT to stop: ");
      ::scanf("%s", DomainName);
      if (::strcmp(DomainName, "EXIT") == 0) break;

      // Register the domain
      ReturnValue = ::asap_register(DomainName,             // Supply the
                                                            // domain name
                                    &Handle,                // Returned
                                                            // handle
                                    &ErrorDetail,           // Returned error
                                                            // information
                                    NULL,                   // Default ASAP
                                                            // ID
                                    true,                   // Enable ranking
                                    true,                   // Concat PID to
                                                            // domain name
```

```
                                    3000,                   // Wait up to 3
                                                            // seconds
                                    "V1");                  // Application
                                                            // version V1

      ::printf("asap_register returned %d, error detail = %d\n", ReturnValue,
               ErrorDetail);

      // Check return value
      if (ReturnValue != ASAP_ERROR_NONE) continue;         // Registration
                                                            // error

      // Registered o.k.; update data items
      // Add 1000 to data item 2 for the domain
      ReturnValue = ::asap_update(Handle,                   // Handle from
                                                            // asap_register
                                  2,                        // Data item 2
                                  1000LL,                   // Value = 1000
                                  ASAP_MATH_ADD,            // Add it to the
                                                            // value
                                  &ErrorDetail);            // Returned error
                                                            // information
      ::printf("asap_update returned %d, error detail = %d\n", ReturnValue,
ErrorDetail);
      // Set data item 5 to 1234
      ReturnValue = ::asap_update(Handle,                   // Handle from
                                                            // asap_register
                                  5,                        // Data item 5
                                  1234LL,                   // Value = 1234
                                  ASAP_MATH_REPLACE_NUMBER, // Set the value
                                  &ErrorDetail);            // Returned error
                                                            // information
      ::printf("asap_update returned %d, error detail = %d\n", ReturnValue,
ErrorDetail);

      // Data items updated; now set status & status text (note that you do
      // not have to update status and state information if you update data
      // items, nor do you have to update data items if you update status and
      // state information. They're
      // completely independent of each other; you can use either or both).
      ReturnValue = ::asap_opstate(Handle,                  // Handle from
                                                            // asap_register
                                   "Sample text",           // Status text;
                                                            // 15 bytes max
                                   ASAP_STATE_CRITICAL,     // Critical state
                                   &ErrorDetail);           // Returned error
                                                            // information
      ::printf("asap_opstate returned %d, error detail = %d\n", ReturnValue,
               ErrorDetail);

      // At this point, if HTMLStatus is TRUE in the asap.conf configuration
      // file, you can look at the asapdomains.html file in the ASAP Hybrid
      // for Linux server directory, and this domain and data will appear.
      // Note: You might have to wait a few seconds for the next server
      // sample interval.
      // Pause to give yourself a chance to look at the data. If you wait for
      // multiple samples, the New flag for the domain changes from True to
      // False because the domain is no longer new.  In addition, the Upd
      // flag changes from True to False because the domain has not been
      // updated in any subsequent sample (because the application will be
      // waiting at this prompt).
      ::printf("Enter C to continue...");
      while (::getchar() != 'C');

      // Remove the domain you just registered
      ReturnValue = ::asap_remove(Handle,                   // Handle from
```

```
                                                        // asap_register
                                    &ErrorDetail,       // Returned error
                                                        // information
                                    true);              // Deallocate
                                                        // domain
      ::printf("asap_remove returned %d, error detail = %d\n", ReturnValue,
ErrorDetail);

   }  // while (true)

}  // int main(int argc, char *argv[])
```

A

ASAP Hybrid EMS Event Messages

This appendix lists detailed information about the EMS event messages generated by ASAP Hybrid.

1901

```
1901:  ASAP INFO HYBRID version date started name cpu
```

An ASAP Hybrid component has successfully started in a processor.

version

is the version of the ASAP component; for example, V01.

date

is the release date of this version.

name

is the process name of the starting component.

cpu

is the processor where the component started.

Cause. Someone started ASAP.

Effect. ASAP starts.

Recovery. Not applicable.

1902

```
1902:  ASAP ERR HYBRID name Error Creating Process pname,
Error:  err errdetail
```

The Proxy SGP could not create the process identified by pname.

name

is the name of the Proxy SGP process.

pname

is the name of the process that failed to start.

err

is the error returned by the process_create_ procedure.

errdetail

is the detailed error returned by the process_create_ procedure.

Cause. The Proxy SGP encountered a process creation error while attempting to start a Hybrid Gateway process.

Effect. The Hybrid Gateway process is not started.

Recovery. Correct the process creation errors and restart ASAP.

1903

```
1903:  ASAP ERR HYBRID name Server I/O Error pname, FE:  err
```

The Proxy SGP could not communicate with the process identified by pname.

name

is the name of the Proxy SGP process.

pname

is the name of the process that the SGP was attempting to communicate with.

err

is the file error encountered by the file-system operation.

Cause. The Proxy SGP encountered a file error while attempting to communicate with a Hybrid Gateway process.

Effect. The Proxy SGP cannot communicate with the Gateway process.

Recovery. Correct the file-system error and retry the operation.

1904

```
1904:  ASAP ERR HYBRID name cname EditReadInit:  err
```

The ASAP Hybrid component could not read its configuration due to a problem encountered by the editreadinit procedure.

name

is the process name of the Hybrid component.

cname

is the name of the configuration file that ASAP was attempting to process.

err

is the error returned by the editreadinit procedure.

Cause. The Hybrid component encountered an error with the editreadinit procedure.

Effect. The Hybrid component cannot read its configuration file.

Recovery. Correct the editreadinit error and restart ASAP.

1905

```
1905:  ASAP ERR HYBRID name cname, FE:  err
```

The ASAP Hybrid component encountered a file-system error while attempting to process its configuration file.

name

is the process name of the Hybrid component.

cname

is the name of the configuration file that ASAP was attempting to process.

err

is the file-system error number.

Cause. The Hybrid component encountered a file-system error.

Effect. The Hybrid component cannot successfully process its configuration file.

Recovery. Correct the file-system error and restart ASAP.

1906

```
1906:  ASAP ERR HYBRID name Unknown Config Opt:  text
```

The ASAP Hybrid component found an error in its configuration file.

name

is the process name of the Hybrid component.

text

is the configuration option that was found to be in error.

Cause. The Hybrid component encountered an error within its configuration file.

Effect. The Hybrid component could not process all its configuration options.

Recovery. Correct the error in the option text and restart ASAP.

1907

```
1907:  ASAP ERR HYBRID name Unknown Param:  text
```

The ASAP Hybrid component found an error in one of its startup parameters.

name

is the process name of the Hybrid component.

text

is the parameter that was found to be in error.

Cause. The Hybrid component encountered an error with a startup parameter.

Effect. The Hybrid component could not process all its startup parameters.

Recovery. Correct the error in the parameter text and restart ASAP.

1908

```
1908:  ASAP ERR HYBRID name $Receive Error, FE:  err
```

An ASAP Hybrid component encountered an error while processing $Receive.

name

is the process name of the Hybrid component.

err

is the file error encountered by the file-system operation.

Cause. An ASAP Hybrid component encountered a file error on $Receive.

Effect. The $Receive operation is not completed. Failure to open $Receive causes the process to abnormally terminate.

Recovery. Correct the file-system error and retry the operation.

1909

```
1909:  ASAP ERR HYBRID name domain operation Err:  err
errdetail
```

A Hybrid Gateway process encountered an error while attempting to register, update, or remove a domain within ASAP.

name

is the name of the Hybrid Gateway process.

domain

is the domain name the operation failed for.

operation

is the type of operation that failed, Register, Update, or Remove.

err

is the ASAP error encountered by the operation.

errdetail

is the ASAP error detail information encountered by the operation.

Cause. A Hybrid Gateway could not register, update, or remove a domain because of an ASAP error.

Effect. The operation is not completed.

Recovery. Correct the ASAP error and retry the operation. For a list of possible ASAP errors returned for the type of operation, see the *ASAP Extension Manual*. For example, see the ASAP_REGISTER_ procedure for a description of registration errors and the ASAP_UPDATE_ procedure for a list of update errors.

1910

```
1910:  ASAP ERR HYBRID name Datagram Error:  text err
```

A Hybrid Gateway process encountered an error while processing a UDP datagram from a remote system.

name

is the name of the Hybrid Gateway process.

text

is a brief text description of the error found in the datagram.

err

is the value found to be in error.

Cause. The datagram has probably been corrupted.

Effect. The Gateway stops processing the datagram.

Recovery. Attempt to identify and correct network problems that could corrupt UDP data.

1911

```
1911:  ASAP ERR HYBRID name Unknown Datagram Received from
ipaddr
```

A Hybrid Gateway process received a foreign or corrupt datagram.

name

is the name of the Hybrid Gateway process.

ipaddr

is the IP address and port number of the sending system.

Cause. A datagram was received that the Hybrid Gateway was not expecting.

Effect. The Gateway does not process the datagram.

Recovery. Ensure the Gateway is configured with a unique IP address and port number.

1912

```
1912:  ASAP INFO HYBRID name Sender ipaddr is UpRev, Version
version
```

A Hybrid Gateway process received a message from a more recent version of ASAP Hybrid on a remote system.

name

is the name of the Hybrid Gateway process.

ipaddr

is the IP address and port number of the sending system.

version

is the sender's version number.

Cause. A newer version of ASAP Hybrid has been installed on a remote system.

Effect. Some of the features in the newer version might not be supported.

Recovery. Install the new version of ASAP Hybrid on the NonStop server.

1913

```
1913:  ASAP ERR HYBRID name Error operation Socket, errno:
errno
```

A Hybrid Gateway process encountered a TCP/IP socket error.

name

is the name of the Hybrid Gateway process.

operation

is the operation that encountered the error.

errno

is the socket error number.

Cause. TCP/IP returned an error when creating, configuring, setting, binding, getting info about, receiving from, or sending to a socket.

Effect. The socket operation does not complete.

Recovery. Correct the socket error and retry the operation.

1914

```
1914:  ASAP ERR HYBRID name Unknown Request on $Receive: buf
```

A Hybrid component received a server request it did not understand.

name

is the name of the Hybrid Gateway process.

buf

is the first word of the buffer received.

Cause. An ASAP Hybrid component received a request buffer that it could not decode.

Effect. ASAP Hybrid ignores the message.

Recovery. Identify and stop the message sender.

Index

A

C

D

G

I

L

R

S

T

V

Content Feedback

First Name: _____ Last Name: _____
Phone: _____ e-mail address: _____
Company: _____

(All contact information fields are required.)

If you're reporting an error or omission, is your issue:

- ☐ **Minor**: I can continue to work, but eventual resolution is requested.
- ☐ **Major**: I can continue to work, but prompt resolution is requested.
- ☐ **Critical**: I cannot continue to work without immediate response.

Comments (give sufficient detail to help us locate the text):

_____
_____
_____
_____
_____

Thank you for taking the time to provide us with your comments.

You can submit this form online, e-mail it as an attachment to pubs.comments@hp.com, fax it to 408-285-5520, or mail it to:

Hewlett-Packard Company
NonStop Enterprise Division
19333 Vallco Parkway, MS 4421
Cupertino, CA 95014-2599
Attn.: Product Manager, Software Publications

4. The ASAP Hybrid for Linux API

A. ASAP Hybrid EMS Event Messages

Index

Examples

Figures

Tables