Pharos TP
Administration Guide

Pharos v4.1.0

Copyright Notice

Restricted Rights Legend

Trademarks

안내서 정보

안내서 제목: Pharos TP Administration Guide
발행일: 2015-03-02
소프트웨어 버전: Pharos TP v4.1.0
안내서 버전: v4.1.0

목 차

그림 목차

표 목차

안내서에 대하여

안내서의 대상

본 안내서는 Pharos TP®(이하 Pharos TP)를 사용하여 어플리케이션 성능을 모니터링 하고자 하는 관리자를 위한 기본 운영 안내서이다. Pharos TP 에이전트를 사용해서 어플리케이션 성능을 모니터링 하기 위해 필요한 환경 설정을 위한 안내서이다. 구축의 이해를 돕기 위해 다양한 예제를 수록해서 사용자의 이해를 돕고자 한다.

안내서의 전제 조건

본 안내서는 관리자가 Pharos TP 에이전트에 대한 전반적인 이해와 Pharos TP 에이전트가 제공하는 각종 기능 및 특성에 대한 습득을 위한 운영 안내서이다.

본 안내서를 원활하게 이해하기 위해서는 다음과 같은 사항을 미리 알고 있어야 한다.

- 운영체제 별 기본 명령어에 대한 기본적인 이해
- Database에 대한 기본적인 이해
- TP Monitor에 대한 기본적인 이해
- TP Service 프로그래밍의 이해
- C 프로그래밍의 이해

관련 안내서

본 안내서는 Pharos TP 에이전트의 기본적인 운영 방법을 설명하며, Pharos TP 설치 및 사용자 안내서는 다루지 않는다.

- "Pharos TP Installation Guide"

Pharos TP의 설치개요 및 수집서버와 에이전트 설치와 제거 방법에 대해 기술한다.

- "Pharos TP User Guide"

Pharos TP의 각 기능설명 및 분석방법에 기술한다.

- "Pharos Server Administration Guide"

수집서버에 대한 환경설정 파일과 시스템 관리방법에 대해 기술한다.

안내서 구성

Pharos TP Administration Guide는 총 8개의 장과 Appendix로 구성되어 있다.
각 장의 주요 내용은 다음과 같다.

- 제1장: Pharos TP소개
  Pharos TP 에이전트의 내부구조, 시스템 구성, 관리, 디렉터리 환경을 기술한다.

- 제2장: 환경변수 설정
  Pharos TP 에이전트에서 사용하는 환경변수의 종류와 역할을 설명하고, 환경변수를 설정하는 방법을 기술한다.

- 제3장: 환경파일 설정
  Pharos TP 에이전트의 환경파일의 종류 및 역할을 설명하고 각 환경파일 파라메타를 설정하는 방법을 기술한다.

- 제4장: 사용자 함수 성능정보 추출 방법
  Pharos TP 에이전트에서 사용하는 사용자 함수들에 대해서 Hooking 모듈을 생성하는 방법을 설명하고, 적용하는 방법을 기술한다.

- 제5장: 데몬 프로세스 성능정보 추출 방법
  Pharos TP 에이전트에서 데몬 프로세스를 모니터링 하기 위해서 데몬 프로세스에서 호출하는 함수들에 대해서 Hooking 모듈을 생성하는 방법을 설명하고, 적용하는 방법을 기술한다.

- 제6장: Pharos TP 실행 및 종료
  Pharos TP 에이전트를 실행하고 종료하는 방법을 기술한다.

- 제7장: Pharos TP 관리도구
  Pharos TP 에이전트 관리도구인 pharosadm의 사용방법을 기술한다.

- 제8장: 부록
  Pharos TP 에이전트를 설치 한 정상적으로 설치되었는지를 확인하기 위한 Check List를 기술한다.

안내서 규약

표 기	의미
<AaBbCc123>	프로그램 소스 코드의 파일명
<Ctrl>+C	Ctrl과 C를 동시에 누름
[Button]	GUI의 버튼 또는 메뉴 이름
진하게	강조
" "(따옴표)	다른 관련 안내서 또는 안내서 내의 다른 장 및 절 언급
'입력항목'	화면 UI에서 입력 항목에 대한 설명
하이퍼링크	메일계정, 웹 사이트
>	메뉴의 진행 순서
+----	하위 디렉터리 또는 파일 있음
|----	하위 디렉터리 또는 파일 없음
참고	참고 또는 주의사항
[그림] 1-1	그림 이름
[표] 1-1	표 이름
AaBbCc123	명령어, 명령어 수행 후 화면에 출력된 결과물, 예제코드
[ ]	옵션 인수 값
|	선택 인수 값
「」	디렉터리 경로 또는 해당 파일 경로

시스템 환경

항목	요구사항	
운영체제	IBM AIX 5.x, IBM AIX 6.x, IBM AIX 7.x	
	HP-UX 11.xx	
	Linux XX	
	Solaris 7~9 (SunOS 5.7~5.9)	
Hardware	RAM	2048MB 이상 메모리 공간 사용
	Disk Space	최소 1GB
	OS	32Bit, 64Bit

지원 TP Monitor

제품명	요구사항
Tmax	Tmax3.0, Tmax 4.0, Tmax 5.0
Tuxedo	Tuxedo 8~12gR1

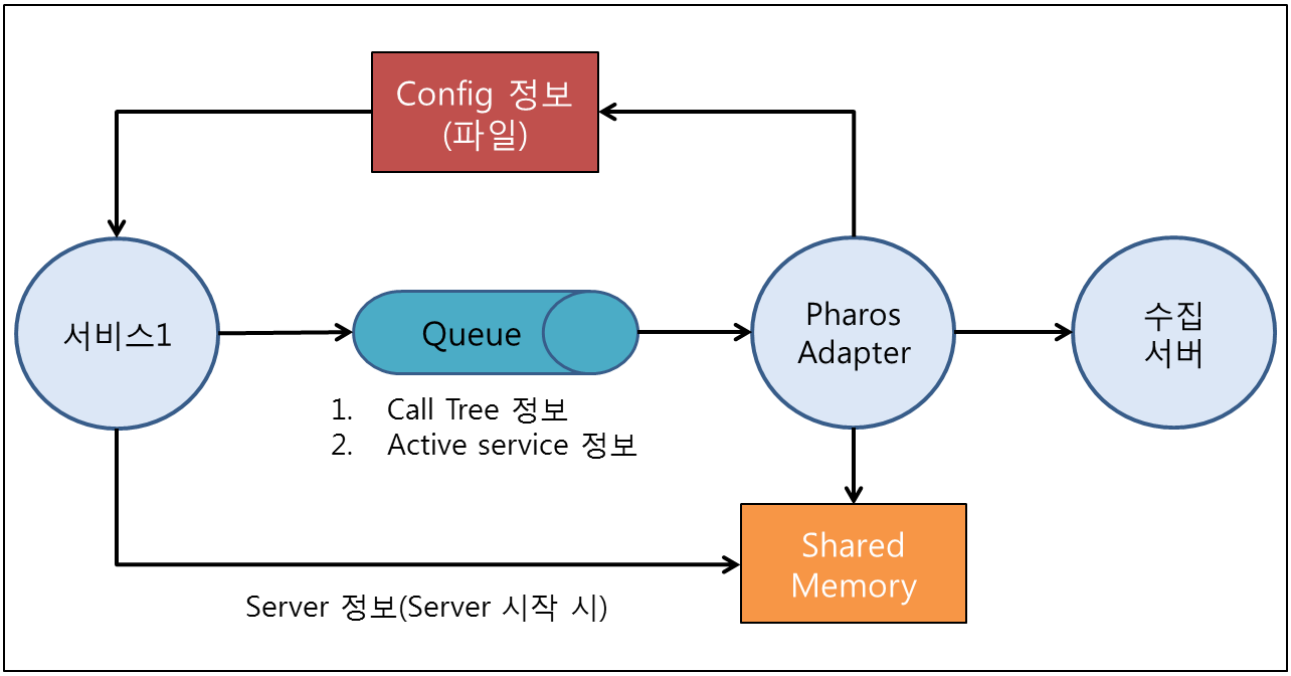
01

Pharos TP 소개

1. Pharos TP 소개

1.1. Pharos TP 개요

Pharos TP 에이전트는 Middleware 시스템 하에서 실행되는 프로세스에서 서비스가 호출되는 시점부터 종료될 때까지 실시간으로 서비스 처리 상태에 대한 Call-Tree 및 데이터베이스 처리 정보를 수집하여 서비스가 현재 어떠한 상태에 놓여 있는지를 파악할 수 있도록 제공하는 시스템이다. 또한 서비스 처리 시 CPU 사용량, 메모리 Leak, 경과시간을 추적하여 서비스가 안정적으로 실행되는지의 정보를 제공하는 시스템이다.



[그림] 1-1 Pharos TP 에이전트 구성도

● Call-Tree 생성

클라이언트로부터 호출된 서비스의 시작부터 종료시점까지 ATMI API나 데이터베이스 SQL에 대한 Call-Tree를 생성한다.

● 메모리 Leak 검출

서비스가 처리되면서 메모리를 할당하고 반환하지 않는지를 추적한다.

● 시스템 리소스 사용량

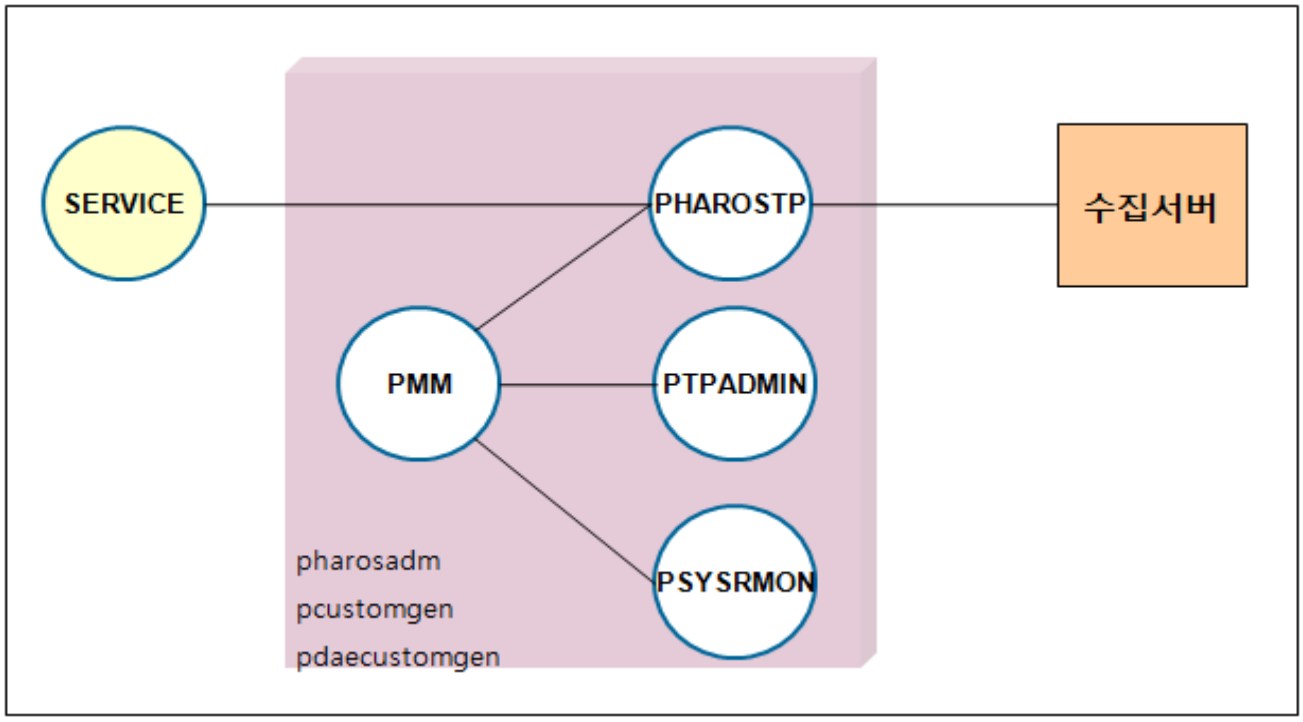
서비스에서 사용한 시스템 리소스 사용량을 추적한다.

● 데이터베이스 처리정보 검출

서비스에서 호출한 SQL 문장 및 SQL에서 사용한 bind 변수 데이터를 추출한다.

1.2. 내부 구조

Pharos TP 에이전트는 내부적으로 4개의 운용 프로세스(PMM, PHAROSTP, PTPADMIN, PSYSRMON)와 3개의 관리 명령어(PHAROSADM, PCUSTOMGEN, PDAECUSTOMGEN)로 구성되어 있다.



[그림] 1-2 Pharos TP 에이전트 프로세스 구조

● 운용 프로세스

- PMM(Pharos 매니저): Pharos TP 프로세스(pharostp, ptpadmin, psysrmon)를 관리하는 프로세스로, 프로세스가 비정상적으로 종료된 경우에 재실행 등을 처리한다.
- PHAROSTP(Adapter): 서버 프로세스의 성능정보를 수집서버에 전달하거나 SQL Hashing 등 Pharos TP의 실질적인 처리를 담당하는 프로세스이다.
- PTPADMIN(TP Monitor Admin): TP Monitor의 정보를 실시간으로 추출하는 프로세스이다.
- PSYSRMON(System Resource Monitoring): 서버 프로세스가 실행되는 node의 시스템 정보를 추출하는 프로세스이다.

● 관리 명령어

- PHAROSADM(Pharos Admin): Pharos TP 에이전트의 환경파일 정보를 조회하거나 실시간으로 변경하는 명령어이다.
- PCUSTOMGEN(사용자 모듈 Hooking): 공유 라이브러리로 개발된 사용자 모듈 중 Hooking을 원하는 모듈에 대해서 Hooking이 가능하도록 소스를 생성하는 명령어이다.
- PDAECUSTOMGEN(Daemon 모듈 Hooking): 데몬 프로세스에서 사용하는 시스템 함수나 공유 라이브러리로 개발된 사용자 모듈 중 Hooking을 원하는 모듈에 대해서 Hooking이 가능하도록 소스를 생성하는 명령어이다.

1.3. 시스템 관리

Pharos TP 에이전트 관리는 크게 정적 관리와 동적 관리로 나뉜다.

● 정적 관리

시스템이 기동되기 전에 이루어질 수 있는 관리, 또는 동작 중인 시스템에 영향을 미치지 않는 관리 행위를 말하며, 시스템 환경설정**(configuration)**작업을 의미한다

● 동적 관리

시스템이 동작하고 있는 상황에서 시스템의 상태나 성능에 영향을 줄 수 있는 관리 행위를 말하며, Pharos TP 에이전트 기동/종료 **(ptpboot/ptpdown)**, 시스템 동작 관리**(pharosadm)** 및 시스템 환경변수 변경 작업을 의미한다.

1.4. 디렉터리 구성

일반적으로 Pharos TP 에이전트가 설치될 파일 시스템의 디렉터리 구성은 다음과 같다.

```
+---- bin
+---- config
+---- data
+---- lib
+---- log
+---- path
+---- license
+---- usrinc
```

Pharos TP 에이전트에서 사용하는 고유의 디렉터리는 다음과 같다.

폴더	설명
bin	Pharos TP 에이전트의 명령어/유틸리티를 포함하는 디렉터리이다
config	환경설정 파일을 저장하는 디렉터리이다. 시스템 변수 : PHAROS_CONFIG
custom	어댑터 프로세스 중 psysrmon 바이너리를 생성하기 위한 파일을 저장하는 디렉터리이다.
data	수집서버 장애 시 데이터 백업을 위한 디렉터리이다
lib	Pharos TP 에이전트 라이브러리 파일을 저장하는 디렉터리이다.

log	Pharos TP Log 파일을 저장하는 디렉터리이다. 시스템 변수 : PHAROS_LOGDIR
path	프로세스 간 내부 통신을 위한 네임드 파이프(named pipe)가 생성될 디렉터리이다.
license	Pharos TP 에이전트에 대한 License 파일을 저장하는 디렉터리이다.
usrinc	사용자 프로그램 변경 시 사용할 헤더 파일을 저장하는 디렉터리이다.

[표] 1-1 Pharos TP 에이전트 디렉터리 설명

1.5. 주요 파일 설명

Pharos TP 에이전트에서 설치될 파일은 운용 프로세스인 pmm, pharostp, ptpadmin, psysrmon과 관리 명령어인 pharosadm, pcustomgen, pdaecustomgen, pcorelogsend, ptpboot, ptpdown이 있다. 기타 환경파일 pharostp.cfg가 있다.

1.5.1. pmm

Pharos TP Manager는 Pharos TP 에이전트 프로세스(pharostp, ptpadmin, psysrmon)를 관리하는 프로세스로 Adapter나 TP Admin 프로세스가 비정상적으로 종료된 경우에 재실행 등을 처리하는 프로세스이다. 단, 사용자가 정상적으로 종료시킨 경우에는 재실행하지 않는다. Pharos TP 에이전트 프로세스는 먼저 Pharos Manager 프로세스와 내부적으로 연결해야 하며, Pharos Manager를 종료하면 자동적으로 Pharos TP 에이전트 프로세스는 자동으로 종료된다. 이 후부터 "pmm" 실행 파일을 Pharos 매니저라 정의한다.

1.5.2. pharostp

Pharos TP Adapter 프로세스는 Middleware 서버 프로세스의 성능정보를 수집서버에 전달하거나 SQL Hashing 등 Pharos TP의 실질적인 처리를 담당하는 실행 파일이다. 이 후부터 "pharostp" 실행 파일을 Pharos 엔진이라 정의한다.

1.5.3. ptpadmin

성능정보 추출 대상인 TP Monitor의 admin 정보를 실시간으로 추출하는 프로세스이다. admin 정보로는 그룹정보, 서버정보, 서비스정보, 큐 정보, 클라이언트 정보 등을 일정간격으로 추출한다.

1.5.4. psysrmon

성능정보 추출 대상인 업무 프로세스가 실행된 시스템의 리소스 정보(CPU, 메모리)를 일정간격으로 추출하는 프로세스이다.

1.5.5. pharosadm

Pharos TP 에이전트 실행 환경을 간편하게 조회하고 변경할 수 있는 명령어로 자세한 실행 방법은 "6.

Pharos TP 실행 및 종료" 설명을 참조한다. 이후부터 "pharosadm" 실행 파일을 Pharos admin이라 정의한다.

1.5.6. pcustomgen

공유라이브러리로 개발된 사용자 모듈 중 Hooking을 원하는 모듈에 대해서 Hooking 가능하도록 소스를 생성하는 파일로 자세한 실행 방법은 "4. 사용자 함수 성능정보 추출방법"의 설명을 참조한다.

1.5.7. pdaecustomgen

데몬 프로세스에서 사용하는 시스템 함수나 공유라이브러리로 개발된 사용자 모듈 중 Hooking을 원하는 모듈에 대해서 Hooking 가능하도록 소스를 생성하는 파일로 자세한 실행 방법은 "5. 데몬 프로세스 성능정보 추출 방법"의 설명을 참조한다.

1.5.8. pcorelogsend

서버 프로세스가 비정상적으로 종료된 경우 이를 분석하여, 분석된 정보를 수집서버에 전달하는 명령이다.

1.5.9. ptpboot

Pharos TP 에이전트를 실행하는 shell 파일이다.

1.5.10. ptpdown

Pharos TP 에이전트를 종료하는 shell 파일이다.

1.5.11. pharostp.cfg

Pharos TP 에이전트 실행 환경을 정의하는 설정 파일이다. 환경 파일의 세부 항목에 대한 자세한 설명은 "2. 환경변수 설정" 설명을 참조한다.

1.5.12. tmaxadminfosvc

TUXEDO 시스템은 ptpadmin 프로세스에서 shell 방식으로 직접 shell를 실행하여 admin 정보를 조회하는 반면, TMAX 시스템의 admin 정보를 조회하는 방법은 TMAX 시스템에서 제공하는 API(tmadmin)를 이용한다. 그러나 tmadmin API는 서버 모드에서만 사용할 수 있는 관계로 admin 정보를 조회할 수 있는 별도의 서비스가 있어야 한다. tmaxadminfosvc 서버 프로세스는 Tmax 시스템에서 Tmax의 admin 정보를 조회하는 서버이다.

참고

tmaxadminfosvc 서버 프로세스 (동 서비스는 반드시 Tmax config에 등록되어야 한다.)

```
*SERVER
tmaxadminfosvc     SVGNAME = svg1, MIN=1, MAX=1,
                        CLOPT="-o $(SVR).$(CDATE).log -e $(SVR).$(CDATE).log"
*SERVICE
TMAXADMINFOSVC     SVRNAME = Tmaxadminfosvc
```

02

환경변수 설정

2. 환경변수 설정

Pharos TP 에이전트에서 사용하는 환경변수는 크게 2가지로 나누어지는데, 첫 번째는 Pharos TP 에이전트에서 사용하는 환경변수이고, 두 번째는 TP Monitor 서버 프로세스에서 사용하는 환경변수이다. Pharos TP 에이전트에서 사용하는 환경변수의 대부분은 Pharos TP 에이전트에 접근하기 위한 변수이다. 또한 환경변수는 해당 시스템에서 수행되는 모든 TP Monitor 서버 프로세스에 동일하게 적용된다. Pharos TP 에이전트가 설치된 서버는 해당 shell의 프로파일에 환경변수를 반드시 설정해야 한다.

2.1. 환경변수

2.1.1. Pharos TP 에이전트 환경변수

다음은 Pharos TP 에이전트에 설정할 환경변수 목록이다.

환경변수	내용
PHOMEDIR	Pharos TP 에이전트의 홈 디렉터리이다.
PHAROS_TPHOME	Pharos TP 프로젝트 홈 디렉터리이다.
PHAROS_CONFIG	Pharos TP 에이전트의 환경 파일을 지정한다.
PHAROS_LOGDIR	Pharos TP 에이전트의 로그 파일 위치를 지정한다.
LOGLEVEL	Pharos TP 에이전트의 로그레벨을 지정한다. 로그레벨은 디버그 모드로 실행되는 환경에서만 적용된다. - E : Error / W : Warning / I : Info / D : Debug / T : Trace / H : Hexdump ● 설정예 LOGLEVEL=EWIDT
OS_VENDER	Pharos TP 에이전트가 실행되는 시스템 벤더를 지정한다 - HP-UX: hp64 / IBM AIX: ibm64 /SUN: sun64 ● 설정예 OS_VENDER=ibm64
ORACLE_VERSION	서버 프로세스에서 접속하는 데이터베이스 버전 번호를 등록한다. oracle이라고만 해도 무방하다. - Oracle 32 Bit : ORA32 / Oracle 64 Bit : ORA64 ● 설정예 ORACLE_VERSION=ORA64
TP_TYPE	TP Monitor 종류를 등록한다. - Tuxedo : tuxedo / Tmax : tmax ● 설정예 TP_TYPE=tuxedo

DB_VENDER	Tmax나 Tuxedo 서버 프로그램에서 사용하는 데이터베이스 벤더를 지정한다. - Oracle : oracle, Altibase: altibase ● 설정예 DB_VENDER=oracle
TP_SERVER_DOWN	서버 프로세스가 실행되면서 Pharos TP 에이전트에 접속하지 못한 경우에 서버 프로세스를 종료할 것인지 여부를 등록한다. ● 설정예 true : 접속 오류 시, 서버 다운 false : 접속 오류 시, 서버 다운하지 않음
LIBPATH	Pharos TP 에이전트 공유라이브러리 경로를 지정한다.
LD_LIBRARY_PATH	Pharos TP 에이전트 공유라이브러리 경로를 지정한다.

[표] 2-1 Pharos TP 에이전트 환경변수

2.1.2. TP Monitor 서버 프로세스 환경변수

다음은 서버 프로세스가 실행되는 계정에서 설정할 환경변수 목록이다.

환경변수	내용
PHOMEDIR	Pharos TP 에이전트의 홈 디렉터리이다.
PHAROS_TPHOME	Pharos TP 프로젝트 홈 디렉터리이다.
PHAROS_CONFIG	Pharos TP 에이전트의 환경 파일을 지정한다.
PHAROS_LOGDIR	Pharos TP 에이전트의 로그 파일 위치를 지정한다.
LOGLEVEL	Pharos TP 에이전트의 로그레벨을 지정한다. 로그레벨은 디버그 모드로 실행되는 환경에서만 적용된다. ● 설정예 - E : Error / W : Warning / I : Info / D : Debug / T : Trace / H : Hexdump
OS_VENDER	Pharos TP 에이전트가 실행되는 시스템 벤더를 지정한다 - HP-UX: hp64 / IBM AIX: ibm64 /SUN: sun64 ● 설정예 OS_VENDER=ibm64
ORACLE_VERSION	서버 프로세스에서 접속하는 데이터베이스 버전 번호를 등록한다. oracle이라고만 해도 무방하다. - Oracle 32 Bit : ORA32 / Oracle 64 Bit : ORA64 ● 설정예 ORACLE_VERSION=ORA64
DB_VENDER	Tmax나 Tuxedo 서버 프로그램에서 사용하는 데이터베이스 벤더를 지정한다. - Oracle : oracle, Altibase: altibase ● 설정예 DB_VENDER=oracle

LIBPATH	Pharos TP 에이전트 공유라이브러리 경로를 지정한다.
LD_LIBRARY_PATH	Pharos TP 에이전트 공유라이브러리 경로를 지정한다.
TP_TYPE	TP Monitor 종류를 등록한다. - Tuxedo : tuxedo / Tmax : tmax ● 설정예 TP_TYPE=tuxedo
PRELOAD 변수	공유 라이브러리 Hooking을 위한 라이브러리 명을 등록한다. ● [Tuxedo] 설정예 IBM 32 : LDR_PRELOAD32=$PHOMEDIR/lib/libpharostpsh.so IBM 64 : LDR_PRELOAD64=$PHOMEDIR/lib/libpharostpsh.so SUN 64 : LD_PRELOAD_64=$PHOMEDIR/lib/libpharostpsh.so 기타 OS : LDR_PRELOAD=$PHOMEDIR/lib/libpharostpsh.so ● [Tmax] 설정예 IBM 32 : LDR_PRELOAD32=$PHOMEDIR/lib/libpharosload.so IBM 64 : LDR_PRELOAD64=$PHOMEDIR/lib/libpharosload.so SUN 64 : LD_PRELOAD_64=$PHOMEDIR/lib/libpharosload.so 기타 OS : LDR_PRELOAD=$PHOMEDIR/lib/libpharosload.so
TP_SERVER_DOWN	서버 프로세스가 실행되면서 Pharos TP 에이전트에 접속하지 못한 경우에 서버 프로세스를 종료할 것인지 여부를 등록한다. ● 설정예 – true : 접속 오류 시 서버 다운 / false : 접속 오류 시 서버 다운하지 않음

[표] 2-2 Pharos TP 서버 프로세스 환경변수

2.2. 환경변수 설정 방법

서버에 환경변수를 설정할 때는 사용하는 shell에 따라 설정할 대상이 다르다. korn shell 및 bash shell의 경우는 각각 <.profile> 및 <.bash_profile>에, c shell의 경우 <.cshrc>에 설정한다.

참고

환경변수를 설정할 때 korn shell과 bash shell의 경우 환경변수명과 등호(=) 사이에는 공백이 없어야 한다

2.2.1. Pharos TP 에이전트 환경변수 설정

<.profile> / <.bash_profile>

```
export PHOMEDIR=/home/pharostp/apm
export PHAROS_TPHOME=$PHOMEDIR/pharostp
export PHAROS_CONFIG=pharostp.cfg
export PHAROS_LOGDIR=$PHOMEDIR/log
export LOGLEVEL=EWIDTH
export OS_VENDER=ibm64
export TP_TYPE=tuxedo
export DB_VENDER=oracle
export ORACLE_VERSION=oracle
export TP_SERVER_DOWN=false
export LIBPATH=$LIBPATH:$PHOMEDIR/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PHOMEDIR/lib
```

<.cshrc>

```
setenv PHOMEDIR /home/pharostp/apm
serenv PHAROS_TPHOME $PHOMEDIR/pharostp
setenv PHAROS_CONFIG pharostp.cfg
setenv PHAROS_LOGDIR $PHOMEDIR/log
setenv LOGLEVEL EWIDTH
setenv OS_VENDER ibm64
setenv TP_TYPE tuxedo
setenv DB_VENDER oracle
setenv ORACLE_VERSION oracle
setenv TP_SERVER_DOWN false
setenv LIBPATH $LIBPATH:$PHOMEDIR/lib
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:$PHOMEDIR/lib
```

[그림] 2-1 Pharos TP 관리자 - pharosadt.env 설정

2.2.2. Tuxedo 서버 프로세스 환경변수 설정

아래의 환경변수 설정 방법은 Tuxedo 시스템인 경우에 사용하는 방법이다.

<.profile> / <.bash_profile>

```
#=================================================================#
# PHAROSTP TPM Environment ( TP-M environment set )
#=================================================================#
export PHOMEDIR=/home/pharostp/apm
export PHAROS_TPHOME=$PHOMEDIR/pharostp
export PHAROS_CONFIG=pharostp.cfg
export PHAROS_LOGDIR=$PHOMEDIR/log
export LOGLEVEL=EWIDT
export OS_VENDER=ibm64
export TP_TYPE=tuxedo
export DB_VENDER=oracle
export ORACLE_VERSION=oracle
export TP_SERVER_DOWN=false
export LIBPATH=$LIBPATH:$PHOMEDIR/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PHOMEDIR/lib

#=================================================================#
# export preload (Tuxedo only)
#=================================================================#
if [ $OS_VENDER = ibm64 ]; then
    export LDR_PRELOAD64=$PHOMEDIR/lib/libpharostpsh.so
elif [ $OS_VENDER = ibm32 ]; then
    export LDR_PRELOAD=$PHOMEDIR/lib/libpharostpsh.so
elif [ $OS_VENDER = sun64 ]; then
    export LD_PRELOAD_64=$PHOMEDIR/lib/libpharosload.so
else
    export LD_PRELOAD=$PHOMEDIR/lib/libpharostpsh.so
fi
```

<.cshrc>

```
#==================================================================#
# PHAROSTP TPM Environment ( TP-M environment set )
#==================================================================#
setenv PHOMEDIR /home/pharosTP/apm
setenv PHAROS_TPHOME $PHOMEDIR/pharostp
setenv PHAROS_CONFIG pharostp.cfg
setenv PHAROS_LOGDIR $PHOMEDIR/log
setenv LOGLEVEL EWIDT
setenv OS_VENDER ibm64
setenv TP_TYPE tuxedo
setenv DB_VENDER oracle
setenv ORACLE_VERSION oracle
setenv TP_SERVER_DOWN false
setenv LIBPATH $LIBPATH:$PHOMEDIR/lib
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:$PHOMEDIR/lib

#==================================================================#
# export preload (Tuxedo only)
#==================================================================#
if [ $OS_VENDER = ibm64 ]; then
      setenv LDR_PRELOAD64 $PHOMEDIR/lib/libpharostpsh.so
elif [ $OS_VENDER ibm32 ]; then
      setenv LDR_PRELOAD $PHOMEDIR/lib/libpharostpsh.so
elif [ $OS_VENDER = sun64 ]; then
      setenv LD_PRELOAD_64 $PHOMEDIR/lib/libpharostpsh.so
else
      setenv LD_PRELOAD $PHOMEDIR/lib/libpharostpsh.so
fi
```

[그림] 2-2 Tuxedo 관리자 - 환경변수 설정

주의

Pharos TP 에이전트가 설치된 계정과 Tuxedo 계정이 다른 경우 LD_PRELOAD에 등록하는 라이브러리에 대한 실행 권한을 부여해야 한다.

2.2.3. Tmax 서버 프로세스 환경변수 설정

아래의 환경변수 설정 방법은 Tmax 시스템인 경우에 사용하는 방법이다.

<.profile> / <.bash_profile>

```
#==================================================================#
# PHAROSTP TPM Environment ( TP-M environment set )
#==================================================================#
export PHOMEDIR=/home/pharostp/apm
export PHAROS_TPHOME=$PHOMEDIR/pharostp
export PHAROS_CONFIG=pharostp.cfg
export PHAROS_LOGDIR=$PHOMEDIR/log
export LOGLEVEL=EWIDT
export OS_VENDER=ibm64
export TP_TYPE=tmax
export DB_VENDER=oracle
export ORACLE_VERSION=oracle
export TP_SERVER_DOWN=false
export LIBPATH=$LIBPATH:$PHOMEDIR/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PHOMEDIR/lib

#==================================================================#
# export preload
#==================================================================#
# Tmax: Vserion 4.0.1  이후
if [ $OS_VENDER = ibm64 ]; then
      export LDR_PRELOAD64=$PHOMEDIR/lib/libpharosload.so
elif [ $OS_VENDER = ibm32 ]; then
      export LDR_PRELOAD=$PHOMEDIR/lib/libpharosload.so
elif [ $OS_VENDER = sun64 ]; then
      export LD_PRELOAD_64=$PHOMEDIR/lib/libpharosload.so
else
      export LD_PRELOAD=$PHOMEDIR/lib/libpharosload.so
fi
```

<.cshrc>

```
#==================================================================#
# PHAROSTP TPM Environment ( TP-M environment set )
#==================================================================#
setenv PHOMEDIR /home/pharostp/apm
setenv PHAROS_TPHOME $PHOMEDIR/pharostp
setenv PHAROS_CONFIG pharostp.cfg
setenv PHAROS_LOGDIR $PHOMEDIR/log
setenv LOGLEVEL EWIDT
setenv OS_VENDER ibm64
setenv TP_TYPE tmax
setenv DB_VENDER oracle
setenv ORACLE_VERSION oracle
setenv TP_SERVER_DOWN false
setenv LIBPATH $LIBPATH:$PHOMEDIR/lib
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:$PHOMEDIR/lib

#==================================================================#
# export preload
#==================================================================#
# Tmax: Vserion 4.0.1 이후
if [ $OS_VENDER = ibm64 ]; then
    setenv LDR_PRELOAD64 $PHOMEDIR/lib/libpharosload.so
elif [ $OS_VENDER = ibm32 ]; then
    setenv LDR_PRELOAD $PHOMEDIR/lib/libpharosload.so
elif [ $OS_VENDER = sun64 ]; then
    setenv LD_PRELOAD_64 $PHOMEDIR/lib/libpharosload.so
else
    setenv LD_PRELOAD $PHOMEDIR/lib/libpharosload.so
fi
```

[그림] 2-3 Tmax 관리자 - 환경변수 설정

2.3. 데이터베이스 환경변수 설정

Pharos TP 에이전트에서 데이터베이스 SQL문(SQL plan)을 실행하기 때문에 에이전트를 실행하는 계정은 반드시 데이터베이스 관련 환경변수를 설정해야 한다.

다음은 에이전트가 실행되는 계정에서 데이터베이스에 관련한 설정할 환경변수 목록이다.

```
#=================================================================#
# Oracle Environment
#=================================================================#
export ORACLE_BASE=/oracle/app/oracle
export ORACLE_HOME=$ORACLE_BASE/product/10.2.0
export ORACLE_SID=KFCORAP1
export NLS_LANG=American_America.K016KSC5601
export PATH=$PATH:$ORACLE_HOME/bin:$ORACLE_BASE/opatch/OPatch
export LIBPATH=$LIBPATH:$ORACLE_HOME/lib:$ORACLE_HOME/lib
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
```

[그림] 2-4 Pharos TP 관리자 - Oracle 환경설정

2.4. Tuxedo 환경변수 설정

Pharos TP 에이전트에서 TP Monitor admin정보를 추출하기 위하여 TP Monitor API를 사용하기 때문에 에이전트를 실행하는 계정은 반드시 TP Monitor 관련 환경변수를 설정해야 한다.

다음은 에이전트가 실행되는 계정에서 Tuxedo 관련 설정할 환경변수 목록이다.

```
#=================================================================#
# Tuxedo Environment
#=================================================================#
export TUXDIR=/usr/ssw/tuxedo
export TUXCONFIG=$TUXDIR/config/tuxconfig;
export BDMCONFIG=$TUXDIR/config/dmconfig;
export TLOGDEVICE=$TUXDIR/log/tlog/TLOG;
export ULOGPFX=$TUXDIR/log/ulog/ULOG;
export ALOGPFX=$TUXDIR/log/ulog/ACCESS;
export FLDTBLDIR32=$TUXDIR/udataobj
export FIELDTBLS32=tpadm,Usysfl32,nhic.fld,nhic_ia.fld,nhic_ib.fld
export LIBPATH=$LIBPATH:$TUXDIR/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$TUXDIR/lib
```

[그림] 2-5 Pharos TP 관리자 – Tuxedo 환경설정

2.5. Tmax 환경변수 설정

Pharos TP 에이전트에서 TP Monitor Admin 정보를 추출하기 위하여 TP Monitor API를 사용하기 때문에 에이전트를 실행하는 계정은 반드시 TP Monitor 관련 환경변수를 설정해야 한다.

다음은 에이전트가 실행되는 계정에서 Tmax 관련 설정할 환경변수 목록이다.

```
#==================================================================#
# Tmax Environment
#==================================================================#
export TMAXDIR=/home/pharostp/Tmax64
export TMAX_HOST_ADDR=175.118.115.24
export TMAX_HOST_PORT=8888
export TMAX_CONNECT_TIMEOUT=3
export PATH=$PATH:/home/Tmax/Tmax5/bin
export LIBPATH=$LIBPATH:$TMAXDIR/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$TMAXDIR/lib
```

[그림] 2-6 Pharos TP 관리자 – Tmax 환경설정

03

환경파일 설정

3. 환경파일 설정

3.1. 개요

PHAROS TP 에이전트의 환경파일에는 성능정보 추출을 위한 환경설정과 에이전트에 대한 환경을 설정한다.

환경파일은 총 13개의 카테고리로 구성된다.

- **에이전트 정보**
- **TP Monitor 정보**
- **수집서버 정보**
- **모니터링 정보**
- **Tuxedo 어드민 모니터링 정보**
- **모듈 모니터링 정보**
- **SQL 모니터링 정보**
- **포지션 정보**
- **에러위치 정보**
- **로그 정보**
- **공유메모리 정보**
- **연계서비스 정보**
- **Advice 정보**

에이전트 정보에는 에이전트에 대한 고유한 이름과 번호 등을 등록하고, TP Monitor 정보는 대상 시스템에서 사용하는 TP Monitor에 대한 정보나 데이터베이스 접속 정보 등을 등록한다. 수집서버 정보는 성능정보를 처리하는 수집서버의 연결 정보를 등록한다. 모니터링 정보는 성능정보 추출에 필요한 레벨, 임계값, 각종 인터벌 정보, 거래추적 예외 서버나 서비스 등을 등록한다. SQL 모니터링 정보는 SQL 처리와 관련된 세션 ID, 바인드 변수, SQL plan, SQL 처리 임계값 등을 등록한다.

포지션 정보는 거래 요청전문 내에 포함된 글로벌 ID, 거래코드 등을 구할 수 있는 위치 정보를 등록하고, 에러위치 정보는 응답전문에서 에러코드나 에러메시지 정보를 구할 수 있는 위치 정보 등을 등록한다. 로그 정보는 TP Monitor 시스템 로그나 필터링 문자열, 코어 파일 위치 등을 등록한다. 공유메모리 정보는 성능정보 추출을 위해서 필요한 정보를 저장하기 위한 공유메모리 키나, 프로세스 수, Call-Tree 개수 등을 등록한다. 마지막으로 연계서비스는 하나의 트랜잭션으로 묶기 위해 서비스를 통하여 다른 시스템과 연계할 때 연계에 사용되는 서비스를 등록한다.

3.1.1. 환경파일의 형식

Pharos TP 에이전트 환경파일은 10개의 카테고리로 구성되어 있다. 이 중 에이전트 정보, TP Monitor 정보, 수집서버 정보, 모니터링 정보, SQL 모니터링 정보, 로그 정보, 공유메모리 정보는 필수로 등록해야 한다.

각 항목의 정의 형식은 다음과 같다.

```
[instance_info] ← 카테고리 명
항목1 = ....
항목2 = ....
항목3 = ....
```

[그림] 3-1 pharostp.cfg - 환경파일 형식

다음은 Pharos TP 환경파일을 설정한 예제이다.

```
[instance_info]
# adapter name
instance.res_name=aix6_TP
# adapter name make flag
# true : agent name = res_name + tp_username
instance.res_name_join=true
# adapter unique number
instance.res_id=100

[tpmon_info]
# 거래추적 대상 tp monitor name
tpmon.tp_name=TUXEDO
# 거래추적 대상 tp monitor version
tpmon.tp_ver=11.1.1.2.0
# 거래추적 대상 tp monitor 실행 계정 명
tpmon.tp_username=pharosTP
…

[server_info]
# Pharos Server IP-ADDR
server.address=175.118.115.24
# Pharos Server에 추적 정보를 전송할 포트번호 [default=8877]
server.send_port=44001
# Pharos Server로부터 요청받을 포트번호 [default=8887]
server.recv_port=44001
```

[그림] 3-2 pharostp.cfg - 환경파일 설정 예제

각 항목은 다음과 같은 형식으로 정의한다. 항목은 '항목 이름 = 값'의 형태로 정의된다.

```
항목 =  형태 (default값)
        범위 또는 크기
        내용
```

[그림] 3-3 pharostp.cfg – 환경파일 항목

각 항목은 다음의 4가지 형태 중 하나의 값으로 정의될 수 있으며, 항목 이름은 값으로 사용될 수 없다.

항목값	형태
numeric	숫자 범위를 정의
string	abc 형태의 문자열
literal	"abc" 형태의 문자열
true/false	TRUE 또는 FALSE 형태

[표] 3-1 Pharos TP – 환경 파일의 형식

3.1.2. 작성시 주의사항

환경파일의 각 항목은 다음과 같은 형식에 따라 작성되어야 한다.

1. 카테고리는 대괄호([*)로 시작하고 정의된 이름을 사용해야 한다.
2. 카테고리 이름이나 하위 개체의 이름은 반드시 첫 번째 칸에서 시작되어야 한다.
3. 항목과 항목 사이의 공백은 의미가 없다.
4. 항목을 정의하지 않으면 기본값으로 설정된다.

3.2. 환경파일 설정

환경파일에 필수적으로 정의되는 에이전트 정보, TP Monitor 정보, 수집서버 정보, 모니터링 정보, SQL 모니터링 정보, 로그 정보, 공유메모리 정보와 기타 포지션 정보, 에러위치 정보, 연계서비스 정보에 대해서 설명한다.

3.2.1. 에이전트 정보

하나의 독립적인 단위인 에이전트에 대한 환경설정을 한다. 에이전트 정보에는 기본적으로 다음과 같은 내용을 설정한다.

● 에이전트 이름을 무엇으로 설정할 것인가?

에이전트 정보의 환경설정 형식은 다음과 같다.

```
##############################################################################
# Instance information
##############################################################################
[instance_info]

# adapter name
instance.res_name=aix6_TP

# adapter name make flag
# true : agent name = res_name + tp_username
instance.res_name_join=false

# adapter unique number
instance.res_id=numeric
```

[그림] 3-4 pharostp.cfg – 에이전트 정보

필수항목

● instance.res_name = string
- 범위 : 12자 이내
- UI상에 표시될 에이전트의 이름을 등록한다. 에이전트의 이름은 멀티 노드 시스템 환경인 경우 유일한 이름을 등록해야 한다. 에이전트는 하나의 instance 개념이다. UI화면에 나오는 에이전트 이름과 동일하다.

● instance.res_name_join = true | false
- 기본값 : false
- 동적 변경 : 불가능
- 에이전트 하나로 두 개 이상의 TP 인스턴스를 관리하는 경우에 에이전트 명을 구분하기 위해 시스템 사용자 명을 에이전트에 추가할 것인지 여부를 등록한다. 시스템 사용자 명을 에이전트 명에 추가하는 경우에 에이전트 명이 12자리가 넘지 않도록 주의해야 한다.
- true : instance.res_name + tpmon.tp_username 로 에이전트 명을 구성
- false : instance.res_name 으로 에이전트 명을 사용

● instance.res_id = numeric
- 범위 : 1~31
- 에이전트를 구분하기 위한 유일한 번호를 등록한다.

3.2.2. TP Monitor 정보

성능정보 추출 대상 시스템에서 사용하는 TP Monitor에 대한 일반정보나 TP Monitor 엔진 프로세스의 이름을 등록한다. 또한 업무 서버 프로세스에서 사용하는 데이터베이스 접속 정보 등을 등록한다.

● 대상 시스템의 TP Monitor는 무엇인가?
● 대상 시스템에서 사용하는 데이터베이스는 무엇인가?

TP Monitor 정보의 환경설정 형식은 다음과 같다.

```
##############################################################################
# TP Monitor information
##############################################################################
[tpmon_info]

# 거래추적 대상 tp monitor name
tpmon.tp_name=TMAX

# 거래추적 대상 tp monitor version
tpmon.tp_ver=5.0

# 거래추적 대상 tp monitor 실행 계정 명
tpmon.tp_username=tmax

# 거래추적 대상 DB 종류
tpmon.db_name=oracle

# 거래추적 대상 DB 접속정보
tpmon.db_coninfo=pharosdb/pharosdb@FAUST11G

# 거래추적 대상 DB 종류
#tpmon.engine_proc=BBL, WSL, WSH, DMADM, GWADM, GWTDOMAIN
tpmon.engine_proc=tmm,clh,cll,tms_ora
```

[그림] 3-5 pharostp.cfg – TP Monitor 정보

필수항목

● tpmon.tp_name = Tuxedo | Tmax
  - 범위 : 9자 이내
  - 거래추적 대상 TP Monitor 이름을 등록한다.

● tpmon.tp_ver = string

- 범위 : 9자 이내
- 거래추적 대상 TP Monitor 버전을 등록한다.

● tpmon.tp_username = string
- 범위 : 18자 이내
- 거래추적 대상 TP monitor 실행 계정 명을 기록한다.

● tpmon. engine_proc = string
- 범위 : 100개 프로세스 이내
- TP Monitor의 가용성을 체크하기 위하여 사용하는 항목으로 TP Monitor의 Boot/Down을 판단할 수 있는 TP Monitor Engine 프로세스 명을 등록한다. TP Monitor Engine 프로세스가 두 개 이상인 경우 Comma로 분리한다.

참고

엔진 프로세스 중 하나라도 Down되면 전체 TP Monitor가 Down된 것으로 판단한다.

선택항목

● tpmon.db_name =string
- 범위 : 10자 이내
- 서버 프로세스에서 사용하는 데이터베이스 명을 등록한다.

● tpmon. db_coninfo = string
- 범위 : 60자 이내
- 데이터베이스 접속 정보를 등록한다. SQL 문장에 대한 SQL plan을 사용하는 경우에는 반드시 접속 정보를 등록해야 한다.

3.2.3. 수집서버 정보

성능정보를 전송할 수집서버의 정보를 등록한다. 에이전트는 해당 정보를 이용하여 수집서버와 연결하고, 성능정보를 전송한다. 다음과 같은 내용을 설정한다.

● 수집서버와 연결할 세션 개수는 몇 개로 할 것인가?

수집서버 정보의 환경설정 형식은 다음과 같다.

```
#############################################################################
# Server connection information setting
#############################################################################
[server_info]

# 수집서버 IP-ADDR
server.address=175.118.115.24

# 수집서버에 추적 정보를 전송할 포트번호 [default=8877]
server.send_port=54002

# 수집서버에 추적 정보를 전송할 세션 수 [min=1, max=10]
server.send_session_num=1

# 수집서버로부터 요청받을 포트번호 [default=8887]
server.recv_port=54002

# 수집서버로부터 요청받을 세션 수  [min=1, max=10]
server.recv_session_num=1
```

[그림] 3-6 pharostp.cfg – 수집서버 정보

필수항목

● server.address = string
- 범위 : 24자 이내
- 수집 서버의 IP Address 를 등록한다.

● server.send_port = numeric
- 범위 : 1000 ~ MAX_INT
- 기본값 : 8877
- 에이전트에서 수집서버로 데이터 전송을 위한 포트번호를 등록한다.

● server.send_session_num = numeric
- 범위 : 1 ~ 10
- 기본값 : 1
- 에이전트에서 수집서버로 데이터 전송을 위하여 연결할 세션 개수를 등록한다.

● server.recv_port = numeric
- 범위 : 1000 ~ MAX_INT
- 기본값 : 8887

- 수집 서버에서 에이전트로 요청을 보내기 위해 사용할 포트번호를 등록한다. 해당 포트번호는 수집 서버의「pharos_env.sh」설정인「PORT_AGENT」에 등록한 포트번호와 동일한 포트번호를 등록해야 한다.

● server.recv_session_num = numeric
- 범위 : 1 ~ 10
- 기본값 : 1
- 수집 서버에서 에이전트로 요청을 보내기 위하여 연결할 세션 개수를 등록한다.

3.2.4. 모니터링 정보

모니터링 정보는 성능정보 추출에서 가장 중요한 정보로 서버 프로세스 Hooking 모듈에서 정보추출 유무나 임계값 등을 판단할 수 있는 정보를 등록한다. 등록하는 정보 중 일부 항목에 대해서 동적 변경이 가능하다.

● 거래추적 정보를 수집할 것인가?
● 서비스 임계치를 적용할 것인가?
● Active 서비스 임계치를 얼마로 할 것인가?

모니터링 정보의 환경설정 형식은 다음과 같다.

```
################################################################################
# Monitoring information setting
################################################################################
[monitoring]

# APM type [tp|db|mci]
mon.type=tp

# APM내부 프로세스간 데이터 송수신 Message queue 키 값 [default=39829]
mon.msgkey=39969

# 거래추적 LEVEL [0=미추적 | 1=서비스 | 2=모듈 | 3=FUNCTION | 4=ATMI | 5=SQL]
# 거래추적 LEVEL 값은 추적하고자 하는 LEVEL보다 1큰 값을 등록해야 함
mon.trace_level=5

# 서비스 임계치(millisecond) : 임계치 이하의 데이터는 Pharos Server전송하지 않음 [default=0]
mon.svc_cp=0

# Active 서비스 리스트에 대한 임계치(millisecond) : 임계치 이상의 데이터만 전송  [default=10000]
mon.act_svc_cp=3000

# Active 서비스 리스트 전송 interval(millisecond) [default=5000]
mon.act_svc_int=5000

# Active 서비스 타임아웃(millisecond) : [default=200000]
mon.svc_timeout=500000

# 시스템의 CPU & MEMORY 정보 전송 interval(millisecond) [default=5000]
mon.sys_info_int=5000

# 거래추적 데이터에 대한 집계여부 [true|false]
mon.sum_flag=true

# 집계 데이터 전송 interval(millisecond) [default=60000]
mon.sum_send_int=60000

# 서비스 usage(CPU & Memory Leek) 생성 여부 [true|false]
mon.svc_sys_usage=true
```

```
# 프로세스별 CPU & Memory 정보 생성 여부 [true|false]
# "tpmon.tp_username" 항목에 등록된 계정의 프로세스에 한함
mon.proc_info_flag=true

# 프로세스별 CPU & Memory 정보 전송 interval(millisecond) [default=5000]
mon.proc_info_int=5000

# Request 정보 전송 interval(밀리초) [default=1000]
mon.req_send_int=1000

# TP Monitor Admin 정보 추출 여부 [true|false]
mon.adm_info_flag=true

# TP Monitor Config성 Admin 정보 전송 interval(millisecond) [default=600000]
mon.adm_cfg_info_int=600000

# TP Monitor Real Time Admin 정보 전송 interval(millisecond) [default=5000]
mon.adm_rt_info_int=5000

# 임계치 이하 데이터 버퍼링 건수 지정 [min=1, max=100]
mon.merge_cnt=10

# 임계치 이하 데이터 중 에러 데이터 버퍼링 여부 [true|false]
mon.err_buf_flag=true

# 거래추적에서 제외할 서버 프로세스 명 : comma로 구분
#mon.skip_svr=BBL, DBBL, LMS, WSL, WSH, DMADM, GWADM, GWTDOMAIN, tmboot, tmshutdown,
tmadmin, JSL, JREPSVR, tmunloadcf, tlisten, tuxadm, apaddusr, tmipcrm, tpkill, tms_ora
mon.skip_svr=tmm, clh, cll, tms_ora

# 거래추적에서 제외할 서비스 명 : comma로 구분
mon.skip_svc=

# 아래 거래코드만 거래추적할 것인지 아니면 제외할 것인지 여부 [positive|negative|false]
mon.txcode_flag=false
# 거래추적관련 거래코드 파일명 (default: config 파일 directory path)
mon.txcode_path=
```

```
# APM 헤더 처리 방법 [0=미사용|1=APM 헤더]
mon.apm_header_flag=0

# APM 헤더 flag가 '1'인 경우 skip 서비스 목록 : comma로 구분
mon.apm_skip_svc=

# flag for transaction tracking by hashcode : [0=Not used | 1=Message | 2=GUID]
mon.hashcode_type=0

# 비동기 호출을 하나의 call tree로 처리할 것인지 여부 [true|false]
mon.async_to_tx=false

# flag for async to sync transformation : [true|false]
# mon.async_timeout : millisecond
mon.async_to_sync=false
mon.async_start_svc=
mon.async_end_svc=
mon.async_timeout=0

# TP DAEMON process config 파일명
mon.proc_config=

# skip count for memory leak check : [default=10]
mon.leak_skip_cnt=10

# flag for polling data send : [true|false]
mon.polling_flag=true
# interval for polling data send(millisecond) : [default=30000]
mon.polling_int=30000
```

[그림] 3-7 pharostp.cfg – 모니터링 정보

필수항목

● mon.type = tp
  - 범위 : 8자 이내
  - 에이전트 타입을 등록한다.
    tp : TP Monitor 에이전트
    db : DB 모니터링 에이전트
    mci : MCI 시스템 모니터링 에이전트
    link : 인터페이스 시스템 모니터링 에이전트

● mon.msgkey = numeric
- 범위 : 1 ~ MAX_INT
- 기본값 : 39829
- 서버 프로세스의 성능정보를 에이전트에 전송하기 위해 APM내부 프로세스간 성능 정보를 송신 하기 위하여 사용할 메시지 큐의 키를 등록한다.

참고

시스템 내부의 기존에 사용되는 message queue 키 값과 중복되면 안된다. 메시지 큐는 동 항목에 정의한 key를 기준으로 12개가 생성된다. 그러므로 key 값을 정의할 때 콘솔 창에서 ipcs –q로 확인되는 키 값과 중복되지 않는 값으로 연속해서 12개를 정의할 수 있는 시작 key를 정의해야 한다

● mon.apm_header_flag = 0 | 1
- 기본값 : 0
- 동적 변경 : 불가
- 거래추적을 위하여 서비스간 호출 시 GUID 정보 전달을 위해서 APM 헤더를 생성하여 사용할 것인지 여부들 등록한다. 거래추적을 하지 않고 TP 모니터 서비스 별로 모니터링 하는 경우에는 0번으로 등록한다.

값	의미
0	거래추적을 하지 않고 단일 시스템만 모니터링 함
1	전문에 GUID가 없이 서비스간 호출 시 하나의 거래로 인식하여 거래추적 하고자 하는 경우 등록 한 시스템 내에서 서비스 간에 호출도 서로 다른 시스템간 호출과 동일하게 인식

[표] 3-2 Pharos TP apm_header_flag

주의

1번으로 사용하는 경우에는 일부 시스템에는 적용하고 일부 시스템에는 적용하지 않은 방식으로 사용하지 못하고 전체 시스템에 적용하든지 아니면 전체 시스템에 적용하지 않던지 해야 한다. 일부 시스템만 적용하는 경우에는 「apm_skip_svc」 항목에 APM 헤더가 전달되지 않도록 관련 서비스 명을 모두 등록해야 한다.

● mon.trace_level = numeric
- 범위 : 0 ~ 5
- 동적 변경 : 가능
- 성능 정보 추출 레벨을 등록한다. 대상이 서비스 단위인지 아니면 SQL 단위까지 성능정보를 추출 할 것인지 등 대상의 상세 레벨을 등록한다.

레벨	의미

0	성능정보를 추출 하지 않음(Hooking을 하지 않음)
1	서비스 단위 까지만 성능정보 추출 서비스 단위인 경우 모듈, SQL, ATMI 단위로는 정보를 추출하지 않음
2	모듈 단위까지 성능정보 추출 서비스 단위의 거래추적은 포함하고 FUNCTION, SQL, ATMI 단위로는 정보를 추출하지 않음
3	FUNCTION 단위까지 성능정보 추출 서비스 단위의 거래추적은 포함하고 SQL, ATMI 단위로는 정보를 추출하지 않음
4	ATMI(tpcall, tpacall, tpforward) 호출 단위까지 성능정보 추출 대상 서버 프로세스의 모든 정보를 추출함
5	SQL 문장 단위까지 성능정보 추출

[표] 3-3 Pharos TP mon.trace_level

선택항목

● mon.svc_cp = numeric
- 범위 : 0 ~ 100000 (밀리초)
- 기본값 : 0
- 동적 변경 : 가능
- 대상 서비스의 임계치를 밀리초 단위로 등록한다. 서비스 처리시간이 임계치 보다 작으면 해당 정보는 수집서버에 통계정보만 전송하고, 그렇지 않으면 성능정보 전체를 전송한다. 서비스에 대해서 추출한 성능정보를 수집서버에 전송하면 에이전트와 수집서버간의 네트워크 과부하가 발생할 수 있으므로, 이를 해소하기 위하여 서비스 임계치 보다 처리시간이 오래 걸린 서비스에 대해서만 수집서버에 전송하여 네트워크 부하를 줄일 수 있다.

● mon.act_svc_cp = numeric
- 범위 : 0 ~ 100000 (밀리초)
- 기본값 : 10000 (10초)
- 동적 변경 : 가능
- 모든 Active 서비스 리스트를 수집서버에 전송하지 않고 임계치가 경과한 서비스만 수집서버에 전달하기 위한 임계치를 등록한다.
-

● mon.act_svc_int = numeric
- 범위 : 1000 ~ 100000 (밀리초)
- 기본값 : 5000 (5초)
- 동적 변경 : 가능
- 현재 수행중인 Active 서비스 리스트를 어느 정도 간격으로 수집서버에 전달할 것인지를 밀리초 단위로 등록한다.

● mon.svc_timeout = numeric
- 범위 : 1000 ~ max (밀리초)
- 기본값 : 200000 (200초)
- 동적 변경 : 불가능
- 특정 서비스가 비정상적으로 종료되어 이를 인지하지 못한 경우에 해당 서비스는 실제적으로 종료되었으나 계속해서 UI에서는 Active 상태에 있게 되는 문제가 있을 수 있다.
- Hooking 모듈에서 Signal를 처리하지 못하는 경우에 서비스가 비정상 종료하게 되면 이와 같은 문제가 발생할 수 있다.
- 이러한 문제점을 해결하기 위해서 동 항목에 등록된 시간이 초과되면 해당 서비스가 종료된 것으로 간주하여 해당 서비스를 Active 상태에서 제거하게 된다.

● mon.sys_info_int = numeric
- 범위 : 1000 ~ 500000 (밀리초)
- 기본값 : 5000 (5초)
- 동적 변경 : 가능
- 업무 서버의 시스템 정보(CPU, 메모리)를 어느 정도 간격으로 수집서버에 전달할 것인지를 밀리초 단위로 등록한다.

● mon.sum_flag = true | false
- 기본값 : false
- 동적 변경 : 가능
- 에이전트 내부적으로 집계를 처리 할 것인지 여부를 등록한다.
- 최초에는 에이전트에서 통계를 생성했기 때문에 의미가 있었지만 현재는 수집서버에서 통계를 생성하기 때문에 의미가 없는 항목이므로 false로 등록한다.
- true : 집계처리 수행
- false : 집계처리 하지 않음

● mon.sum_send_int = numeric
- 범위 : 60000 ~ 1000000
- 기본값 : 60000 (밀리초)
- 동적 변경 : 가능
- 에이전트 내부적으로 집계한 집계정보를 어느 정도 간격으로 수집 서버에 전달할 것인지를 밀리초 단위로 등록한다.
- sum_flag가 false이면 동 항목은 사용하지 않는다.

● mon.svc_sys_usage = true | false
- 기본값 : false
- 동적 변경 : 가능
- 서비스에 대한 성능 정보 중 서비스를 처리할 때 사용한 CPU 사용량 및 메모리 Leak 정보를 수집할 것인지 여부를 등록한다.

- true : 정보 수집
- false : 정보 수집하지 않음

● mon.proc_info_flag = true | false
- 기본값 : false
- 동적 변경 : 가능
- TP Monitor 하에서 실행되는 서버 프로세스에 대한 CPU 및 메모리 정보를 수집할 것인지 여부를 등록한다. 정보를 수집하는 경우 "tpmon.tp_username" 항목의 계정으로 실행된 프로세스에 대한 정보만 수집한다.
- true : 정보 수집
- false : 정보 수집하지 않음

● mon.proc_info_int = numeric
- 범위 : 5000 ~ 100000 (밀리초)
- 기본값 : 60000 (60초)
- 동적 변경 : 가능
- TP Monitor 하에서 실행되는 서버 프로세스에 대한 CPU 및 메모리 정보를 어느 정도 간격으로 수집 서버에 전달할 것인지를 밀리초 단위로 등록한다.

● mon.req_send_int = numeric
- 범위 : 100 ~ 100000 (밀리초)
- 기본값 : 1000 (1초)
- 동적 변경 : 불가능
- 입력된 거래 건수와 종료된 거래 건수 정보를 어느 정도 간격으로 수집 서버에 전달할 것인지를 밀리초 단위로 등록한다.

● mon.adm_info_flag = true | false
- 기본값 : false
- 동적 변경 : 가능
- TP Monitor Admin 정보를 수집할 것인지 여부를 등록한다. Admin 정보로는 서버 프로세스, 서비스 Queue 정보 등이 있다.
- true : 정보 수집
- false : 정보 수집하지 않음

● mon.adm_cfg_info_int = numeric
- 범위 : 300000 ~ 1000000 (밀리초)
- 기본값 : 600000 (600초)
- 동적 변경 : 가능
- TP Monitor Admin 정보 중 Config성 정보를 어느 정도 간격으로 수집서버에 전달할 것인지를 초단위로 등록한다. Config 정보로는 그룹정보, 서버정보, 서비스 정보 등을 수집한다.

● mon.adm_rt_info_int = numeric
- 범위 : 1000 ~ 100000 (밀리초)
- 기본값 : 5000 (5초)
- 동적 변경 : 가능
- TP Monitor Admin 정보 중 실시간성 정보를 어느 정도 간격으로 수집서버에 전달할 것인지를 밀리초 단위로 정의한다. 실시간 정보로는 현재는 큐 정보만 수집한다.

● mon.merge_cnt = numeric
- 범위 : 1 ~ 100
- 동적 변경 : 가능
- 임계치 이하 데이터에 대한 집계정보를 매번 수집서버에 전송하면 부하가 발생하므로, 이를 최소화하기 위하여 버퍼링하여 수집서버에 전송한다. 이를 위한 버퍼링 건수를 등록한다

● mon.err_buf_flag = true | false
- 기본값 : false
- 동적 변경 : 가능
- 임계치 이하 데이터 중 에러가 발생한 거래에 대해서 버퍼링 할 것인지 아니면 Call-Tree 정보를 전송할 것인지 여부를 등록한다.
- true : 버퍼링
- false : 버퍼링 하지 않음

● mon.skip_svr = string
- 범위 : 100개 서버 이내
- 성능정보 추출에서 제외할 서버 프로세스 명을 등록한다. 등록할 서버가 두 개 이상인 경우 Comma로 분리한다. 특수문자 '*' 문자를 이용하여 등록이 가능하다. 특수문자 '*' 문자를 이용하여 등록하는 경우에 '*' 문자를 처음이나 중간에 사용할 수는 없고 항상 뒤에 사용해야 한다.
  예) *SVR : 사용불가, SVR*P : 사용불가, SVR* : 사용가능

● mon.skip_svc = string
- 범위 : 100개 서비스 이내
- 성능정보 추출에서 제외할 서비스 명을 등록한다. 등록할 서비스가 두 개 이상인 경우 Comma로 분리한다. 특수문자 '*' 문자를 이용하여 등록이 가능하다. 특수문자 '*' 문자를 이용하여 등록하는 경우에 '*' 문자를 처음이나 중간에 사용할 수는 없고 항상 뒤에 사용해야 한다.
  예) *SVC : 사용불가, SVC*P : 사용불가, SVC* : 사용가능

● mon.txcode_flag = positive | negative | false (Version 3.5.2 이상)
- 기본값 : false
- 동적 변경 : 가능
- 등록된 거래코드에 대해서 성능정보를 추출 할 것인지 아니면 제외할 것인지, 또는 거래코드를

등록하지 않을 것인지 여부를 등록한다.
- positive : 등록된 거래코드만 성능정보를 추출함
- negative : 등록된 거래코드는 성능정보를 추출하지 않음
- false : 거래코드 미등록

참고

동 항목을 등록한 경우에는 반드시 전문 내에서 거래코드를 구할 수 있는 방법을 등록해야 한다. TP 버퍼 타입이 STRING이나 CARRAY인 경우 [position] 절의 txcode_pos, txcode_len 항목의 값을 반드시 등록해야 한다.

● mon.txcode_path = string (Version 3.5.2 이상)
- 범위 : 126자 이내
- 거래코드 리스트가 등록된 파일명을 등록한다.
- 거래코드 등록 방법은 "3.2.13. 거래코드 정보 등록" 절을 참조한다.

● mon.apm_skip_svc = string
- 범위 : 100개 서비스 이내
- 거래추적을 위해 APM 헤더를 사용할 때 하나의 하나의 TP 시스템 속하는 모든 서비스는 기본적으로 거래추적을 위하여 서비스간 호출 시 GUID 정보 전달을 위해서 APM 헤더를 전달하는데, 이때 거래추적에서 제외할 서비스가 있는 경우 APM 헤더를 전달하지 않기 위하여 예외 서비스를 등록하는 항목이다.
- 또한, Pharos TP가 설치되어 있는 않는 다른 TP 시스템의 서비스를 호출하는 경우에는 반드시 예외 서비스를 등록해야 한다. 왜냐하면 서비스 호출시 항상 APM 헤더를 추가하여 전달하기 때문에 호출 받는 서비스에 Pharos TP가 설치되어 있지 않으면 APM 헤더를 제거할 수 없어 에러가 발생하기 때문이다.
- 특수문자 '*' 문자를 이용하여 등록이 가능하다. 특수문자 '*' 문자를 이용하여 등록하는 경우에 '*' 문자를 처음이나 중간에 사용할 수는 없고 항상 뒤에 사용해야 한다.
  예) *SVC : 사용불가, SVC*P : 사용불가, SVC* : 사용가능

● mon.hashcode_type = numeric (Version 4.0.3 이상)
- 기본값 : 0
- 동적 변경 : 불가능
- 전문에 GUID가 있는 경우 APM 헤더를 사용하지 않고 거래를 추적할 수 있다. 이때 거래를 하나의 Call-Tree로 만들기 위해서는 Caller와 Callee간 연결번호가 있어야 하는데, 해당 번호를 무엇으로 할 것인지를 등록한다

타입	의미
0	- 해쉬코드를 연결번호로 사용하지 않음
1	- 서비스가 시작할 때 전달받은 메시지를 해쉬코드로 생성하여 연결번호(Callee)로

	사용 - 서비스 호출시에 호출할 때 사용하는 메시지를 해쉬코드로 생성하여 연결번호(Caller)로 사용
2	- 서비스가 시작할 때 전달받은 메시지 내에 있는 글로벌 ID를 해쉬코드로 생성하여 연결번호(Callee)로 사용 - 서비스 호출시에 호출할 때 사용하는 메시지 내에 있는 글로벌 ID를 해쉬코드로 생성하여 연결번호(Caller)로 사용
3	- 서비스가 시작할 때 전달받은 메시지와 서비스 명으로 해쉬코드를 생성하여 연결번호(Callee)로 사용 - 서비스 호출시에 호출할 때 사용하는 메시지와 서비스 명으로 해쉬코드를 생성하여 연결번호(Caller)로 사용

[표] 3-4 Pharos TP mon.hashcode_type

● mon.tracking_start_svc = string (Version 4.0.3 이상)

- 범위 : 100개 서비스 이내
- 거래추적에서 Call-Tree를 생성하기 위해서는 최초로 시작하는 서비스의 자신의 일련번호가 "0"번이 되어야 한다. Call-Tree 생성은 "0"번부터 차례로 호출 관계를 찾아가면서 전체를 하나의 Call-Tree로 만든다.
- TP to TP로 구성된 시스템에 대해서 APM 헤더를 사용하지 않고 해쉬코드 기반으로 거래추적을 하는 경우에 모든 서비스의 자신의 일련번호가 해쉬코드를 사용하지 때문에 "0"번이 없는 문제가 발생할 수 있다.
- "0"번이 없다 보니 거래 시작이 어느 서비스에서 시작되었는지 알 수가 없기 때문에 전체를 하나의 Call-Tree로 생성하지 못하고 또한 통계를 생성하지 못하는 문제가 발생한다.
- 이런 문제를 해결하지 위해서 동 항목에 서비스를 등록하면 해당 서비스가 최초로 호출되는 서비스로 인식하여 자신의 일련번호를 "0"번으로 설정한다. 그러나 무조건 동 항목에 등록되었다고 자신의 일련번호를 "0"번으로 하는 것이 아니라 전문 상에 있는 글로벌 ID 일련번호가 "<= 1" 조건이 경우에만 "0"번으로 처리한다.
- 이렇게 하는 이유는 등록된 서비스가 최초 서비스가 아닐 수 도 있기 때문이다.

주의

동 항목을 서비스를 등록하면 반드시 "position"절의 글로벌 ID 일련번호 위치 정보를 등록해야 한다. 동 항목에 등록 되었다고 무조건 시작 서비스로 인식하지 않고 글로벌 ID 일련번호에 따라서 시작 서비스로 인식하기 때문이다.

position.gid_seq_pos, postion.gid_seq_len 두 개의 항목에 글로벌 ID 위치 정보를 등록하면 된다.

● mon.async_to_tx = true|false

- 기본값 : false
- 동적 변경 : 가능
- 비동기 호출(tpacall)에 대해서도 하나의 거래로 인식하여 Call-Tree로 처리할 것인지 여부를 등록한다. 비동기 호출은 기본적으로 하나의 거래로 인식하지 않는다. 왜냐하면 비동기 호출은

거래가 언제 종료될지 모르고 또한 비동기 호출 내에서 또 다른 동기 호출이 있을 수 있어 이를 연계할 수 없기 때문이다.

- true : 하나의 거래로 인식
- false : 하나의 거래로 인식하지 않음

● mon.async_to_sync = true|false

- 기본값 : false
- 동적 변경 : 불가능
- 비동기 거래란 거래추적에서 처음으로 호출되는 서비스가 처리 중에 호출하는 서비스의 응답을 기다리지 않고 종료하는 거래를 말한다.
- 일반적으로 "mon.async_to_tx" 항목을 true로 설정하면 비동기 거래도 하나의 Call-Tree로 묶여져서 표현하기 때문에 전체 처리가 맞게 표현된다.
- 그러나 여러 서비스 중에서 하나라도 성능 정보를 추출하지 않으면 호출관계가 끊어지기 때문에 (보통 TP로 구현한 MCI인 경우 처음과 끝만 성능 정보를 추출하면 중간 단계가 없어서 연결 정보가 없어짐) 해당 거래의 전체 처리기간은 호출관계가 끊어지지 전까지의 시간으로 표현하여 실질적인 거래의 전체 처리시간이 왜곡되는 문제가 발생한다.
- 이러한 문제를 해결하기 위해서 비동기 거래를 동기 방식으로 처리할 수 있도록 등록하는 항목이다.
- 동 항목을 설정하면 도중에 연결관계가 끊어지더라도 하나의 거래로 인식한다.
- true : 비동기 거래를 동기 방식으로 처리
- false : 비동기 거래를 동기 방식으로 처리하지 않음

● mon.async_start_svc = string

- 범위 : 100개 서비스 이내
- 동적 변경 : 불가능
- "mon.async_to_sync" 항목이 true인 경우에 비동기 거래의 시작 서비스 명을 등록한다.
- 비동기 거래는 시작 서비스가 반드시 있어야 하기 때문에 "mon.async_to_sync" 항목을 true로 설정하는 경우에 반드시 등록해야 한다.

● mon.async_end_svc = string

- 범위 : 100개 서비스 이내
- 동적 변경 : 불가능
- 비동기 거래가 언제 끝나는지 판단하는 방법은 종료 서비스로 등록된 서비스가 종료되면 해당 거래가 종료된 것으로 판단한다.
- 그렇기 때문에 "mon.async_to_sync" 항목을 true로 설정하는 경우에 반드시 종료 서비스를 등록해야 한다.

● mon.async_timeout = numeric

- 기본값 : 0 (밀리초)
- 동적 변경 : 가능

- 비동기 거래의 종료 서비스가 호출되지 않거나 또는 종료 서비스가 갑자기 비정상적으로 종료된 경우에 해당 거래는 끝나지 안은 것으로 판단하여 계속해서 처리 중 상태로 표현한다.
- 이러한 경우에 지정된 시간이 지나면 타임아웃을 발생하여 해당 거래를 종료하고자 할 때 동 항목에 타임아웃 값을 밀리초 단위로 등록한다

● mon.proc_config = string
- 범위 : 126자 이내
- TP Monitor 환경에서 TP에 등록된 서버 프로세스 이 외에 데몬 형식으로 실행되는 프로세스에 대해 성능 정보를 수집하기 위해서 데몬 프로세스의 정보가 등록된 파일명을 등록한다.
- 데몬 프로세스 정보 등록 방법은 "3.2.14. 데몬 프로세스 정보 등록" 절을 참조한다.

● mon.leak_skip_cnt = numeric
- 범위 : 10 ~ MAX_INT (횟수)
- 기본값 : 10
- 동적 변경 : 불가능
- svc_sys_usage 항목이 true이면 서비스를 처리할 때 사용한 메모리 량을 계산하여 메모리 Leak이 발생했는지 체크하게 되는데 처음 프로세스가 실행 될 때부터 체크하게 되면 거의 대부분 메모리 Leak이 발생하게 된다.
- 이런 문제점을 해결하기 위해 처음 몇 번 동안에는 메모리 Leak을 조사하지 않도록 제외할 횟수를 등록하는 항목이다.

.

● mon.polling_flag = true|false (Version 4.0.3 이상)
- 기본값 : false
- 동적 변경 : 불가능
- 업무서버와 수집서버 사이에 방어벽이 있는 경우에 일정 시간 동안 데이터 통신이 없으면 자동으로 방어벽에서 세션을 끊기 때문에 수집서버와 통신할 때 장애가 발생한다. 이를 위해 Polling 데이터를 주기적으로 전송할 것인지 여부를 등록한다.
- true : 에이전트와 수집서버간 Polling 처리
- false : 에이전트와 수집서버간 Polling 하지 않음

● mon.polling_int = numeric (Version 4.0.3 이상)
- 범위 : 10000 ~ 600000
- 기본값 : 30000 (30초)
- 동적 변경 : 불가능
- polling_flag 항목이 true인 경우에 어느 정도 간격으로 Polling 데이터를 전송할 것인지 밀리초 단위로 등록한다.

3.2.5. TUXEDO 어드민 모니터링 정보

Tuxedo 시스템인 경우 서버 프로세스의 큐 정보는 tmadmin 명령어를 통하여 각 서버 프로세스 별 큐

정보를 조회한다. 그러나 Tuxedo 계정과 Pharos TP 계정이 다른 경우에 권한 문제로 인하여 tmadmin 명령어를 실행시킬 수 없을 수가 있다. 이 경우에 Tuxedo 계정에서 tmadmin 명령어를 실행하여 결과를 파일에 저장하면 이를 읽어서 처리하는 방식으로 큐 정보를 조회할 수 있다.

TUXEDO 어드민 모니터링 정보의 환경설정 형식은 다음과 같다.

```
##############################################################################
# TUXEDO Admin Monitoring information setting
##############################################################################
[tpadmin]

# Queue Data Query Mode (1=tpadmin -r, 2=tpadmin -r file name)
tpadmin.queue_inquary=

# Queue File Full Path
tpadmin.queue_file=
```

[그림] 3-8 pharostp.cfg – TUXEDO 어드민 모니터링 정보

선택항목

● tpadmin.queue_inquary = (1=tpadmin -r, 2=tpadmin -r file name)
- 범위 : 1 ~ 2
- 기본값 : 1
- 동적 변경 : 불가능
- Tuxedo도의 경우 큐 정보를 조회할 때 Tuxedo admin 명령어를 사용하여 큐 정보를 구하게 되는데, Tuxedo admin 명령어의 권한이 없어 실행하지 못하는 문제가 있다. 이를 위해 Tuxedo 계정에서 admin 명령어로 큐 정보를 파일로 만들어 주면 동 파일을 읽어서 처리할 수 있다. 동 항목은 Tuxedo admin명령어를 직접 사용할 것인지 아니면 파일에서 큐 정보를 조회할 것인지 여부를 등록한다
- 1 : tmadmin 명령어를 통하여 큐 정보 조회
- 2 : 파일을 통하여 큐 정보 조회

● tpadmin.queue_file = string
- 범위 : 126 자 이내
- 동적 변경 : 불가능
- queue_inquary 항목의 값이 "2"인 경우에 파일을 통해서 큐 정보를 조회하는데, 큐 정보가 저장된 파일명을 등록한다.

3.2.6. 모듈 모니터링 정보

대상 서비스에서 사용하는 모듈에 대한 성능정보 추출을 어떻게 할 것인지에 대한 정보를 등록한다. 등록하는 정보 중 일부 항목에 대해서 동적 변경이 가능하다.

● 모듈에 대한 임계치를 얼마로 할 것인가?

모듈 모니터링 정보의 환경설정 형식은 다음과 같다.

```
###############################################################################
# Module Monitoring information setting
###############################################################################
[module]
# 모듈 임계치(밀리초) : 임계치 이하 데이터는 Call-Tree에 나타나지 않음 [default=0]
# 단, 하위에 Entry가 존재시 임계치 이하 데이터도 포함
module.mod time limit=0
```

[그림] 3-9 pharostp.cfg – 모듈 모니터링 정보

선택항목

● module.mod_time_limit = numeric (Version 3.5.2 이상)

- 범위 : 0 ~ MAX_INT (밀리초)
- 기본값 : 0
- 동적 변경 : 가능
- 모듈 처리 시간에 대한 임계치를 등록하는 항목이다. 동 항목을 설정하면 임계치 이상의 모듈에 대해서만 Call-Tree에 표현한다. 단, 해당 모듈 하위에 Call-Tree가 존재하는 경우에는 임계치에 관계없이 Call-Tree에 표현한다.

3.2.7. SQL 모니터링 정보

서비스에서 호출하는 SQL 문장에 대해 성능정보 추출을 어떻게 할 것인지에 대한 정보를 등록한다. 등록하는 정보 중 일부 항목에 대해서 동적 변경이 가능하다.

● 바인드 데이터를 수집할 것인가?
● SQL 텍스트에 대해 해싱을 할 것인가?

SQL 모니터링 정보의 환경설정 형식은 다음과 같다.

```
#############################################################################
# SQL Monitoring information setting
#############################################################################
[sql_monitoring]

# Session 정보 수집 여부 [true|false]
sql.sid_coll_flag=true

# SQL bind data 수집 여부 [true|false]
sql.bind_coll_flag=true

# SQL plan 여부 [true|false]
sql.sql_plan_flag=true

# SQL TEXT hashing 여부 [true|false]
sql.sql_hash_flag=true

# SQL 임계치 예외 건수: 임계치와 관계없이 call tree에 포함할 개수 [default=100]
sql.sql_stack_cnt=100

# SQL 임계치(밀리초) : 임계치 이하 데이터는 call tree에 나타나지 않음 [default=0]
sql.sql_time_limit=0

# code list for exception sql error code : [max 128 number]
sql.skip_err_code=
```

[그림] 3-10 pharostp.cfg – SQL 모니터링 정보

선택항목

● sql.sid_coll_flag = true | false
- 기본값 : false
- 서버 프로세스에서 데이터베이스 연결 정보인 세션 정보를 조회할 것인지 여부를 등록한다. 세션 정보를 조회하는 이유는 서버 프로세스에서 수행한 SQL과 데이터베이스 관리 툴과 연계하기 위함이다. DB 모니터링하는 별도의 시스템과 연계시키기 위한 변수이다.
- true : 세션 정보를 구함
- false : 세션 정보를 구하지 않음

● sql.bind_coll_flag = true | false
- 기본값 : false

- SQL 텍스트 중 바인드 데이터를 구할 것인지 여부를 등록한다.
- true : 바인드 데이터를 구함
- false : 바인드 데이터를 구하지 않음

● sql.sql_plan_flag = true | false
- 기본값 : false
- SQL 텍스트에 대해서 Query Plan 실행할 것인지 여부를 등록한다.
- true : Plan 처리
- false : Plan 처리하지 않음

참고

SQL Plan를 실행하기 위해서는 반드시 "3.2.2. TP Monitor 정보" 항목 중 "db_coninfo" 항목을 등록해야 한다. 또한 Plan을 실행하는 모든 테이블에 대해서 권한을 부여해야 한다.

● sql.sql_hash_flag = true | false
- 기본값 : false
- SQL 문장에 대해서 에이전트 내부적으로 해싱하여 SQL 문장을 보관하고, 동일 SQL 문장에 대해서는 해쉬코드만 수집서버에 전달하는 기능을 사용할 것인지 여부를 등록한다. 에이전트에 해싱된 SQL문장의 해쉬코드는 PHAROS 메니져(pharosadm)를 통하여 삭제 하거나, 또는 수집서버와 TCP/IP 연결이 단절되면 자동적으로 해쉬코드는 삭제된다.
- true : SQL문 해싱
- false : SQL문 해싱하지 않음

● sql.sql_stack_cnt = numeric
- 범위 : 1 ~ 1020
- 기본값 : 100
- 동적 변경 : 가능
- SQL 처리 시간에 대한 임계치가 설정되어 있는 경우 SQL 임계치 이하의 데이터는 Call-Tree에 표현하지 않는다. 그러나 모든 SQL이 SQL 임계치 이하이면 Call-Tree에 아무것도 표현하지 않기 때문에, 이것을 방지하기 위하여 기본적으로 나타나야 할 SQL Call-Tree 건수를 등록하는 항목이다. 동 항목에 정의된 개수만큼은 SQL 임계치에 관계없이 Call-Tree에 표현한다.

● sql.sql_time_limit = numeric
- 범위 : 0 ~ MAX_INT(밀리초)
- 기본값 : 0
- 동적 변경 : 가능
- SQL 처리 시간에 대한 임계치를 등록하는 항목이다. 동 항목을 설정하면 위의「sql.sql_stack_cnt」항목값에 설정된 Tree 건수에 추가적으로 임계치 이상의 SQL에 대해서만 Call-Tree만 표현한다. 즉,「sql.sql_stack_cnt」항목을 초과한 경우에는 동 항목에 정의된 SQL 임계치 이상의 SQL에 대해서만 Call-Tree에 포함한다.

● sql.skip_err_code = numeric, numeric, ...

- 범위 : 최대 128개
- 동적 변경 : 불가능
- SQL 처리 시 에러가 발생하는 경우 해당 에러를 에러로 인식하지 않고 정상으로 처리하고자 하는 경우 예외 에러코드를 등록한다.

3.2.8. 포지션 정보

서비스에서 사용하는 버퍼 타입이 STRING, CARRAY인 경우에 전문 내에서 글로벌 ID나 거래코드를 추출하기 위한 위치 정보를 등록한다.

● 글로벌 ID의 위치는 어디인가?

포지션 정보의 환경설정 형식은 다음과 같다.

```
##############################################################################
# Monitor information setting
##############################################################################
[position]

# Buffer type이 CARRAY, STRING인 경우 업무코드 위치 및 길이 (위치는 1부터 시작)
#position.appl_code_pos=96
#position.appl_code_len=4

# Buffer type이 CARRAY, STRING인 경우 거래코드 위치 및 길이 (위치는 1부터 시작)
#position.svc_name_flag=true
#position.txcode_pos=100
#position.txcode_len=10

# Buffer type이 CARRAY, STRING인 경우 글로벌ID 위치 및 길이 (위치는 1부터 시작)
#position.gid_pos=10
#position.gid_len=30

# Buffer type이 CARRAY, STRING인 경우 글로벌ID 일련번호 위치 및 길이 (위치는 1부터 시작)
#position.gid_seq_pos=40
#position.gid_seq_len=2
```

```
# Buffer type이 CARRAY, STRING인 경우 사용자 메시지 위치 및 길이 (위치는 1부터 시작)
#position.user_msg_pos=10
#position.user_msg_len=30

# Buffer type이 CARRAY, STRING인 경우 채널코드 위치 및 길이 (위치는 1부터 시작)
#position.chnl_code_pos=112
#position.chnl_code_len=3
```

[그림] 3-11 pharostp.cfg – SQL 포지션 정보

선택항목

● position.appl_code_pos = numeric (Version 3.5.2 이상)
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 업무코드 시작 위치를 등록한다.
- 시작위치는 1부터 시작한다.

● position.appl_code_len = numeric (Version 3.5.2 이상)
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 업무코드 길이를 등록한다.
- 수집한 업무코드는 UI의 메타 정보에 표현된다.

● position.svc_name_flag = true|false
- 기본값 : false
- 거래코드가 서비스 명과 동일한지 여부를 등록한다.
- true : 서비스 명과 거래코드가 동일
- false : 서비스 명과 거래코드가 동일하지 않음

● position.txcode_pos = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 거래코드 시작 위치를 등록한다.
- 시작위치는 1부터 시작한다.

● position.txcode_len = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 거래코드 길이를 등록한다.

● position.gid_pos = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 글로벌 ID 시작 위치를 등록한다.

- 시작위치는 1부터 시작한다.

● position.gid_len = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 글로벌 ID 길이를 등록한다.

● position.gid_seq_pos = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 글로벌 ID 일련번호 시작 위치를 등록한다.
- 시작위치는 1부터 시작한다.
- 현재는 사용하지 않기 때문에 등록하지 않아도 된다.

● position.gid_seq_len = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 글로벌 ID 일련번호 길이를 등록한다.
- 현재는 사용하지 않기 때문에 등록하지 않아도 된다.

● position.user_msg_pos = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 데이터의 일부를 UI에 표현할 수 있다. UI에 표현할 전문데이터의 시작 위치를 등록한다.
- 시작위치는 1부터 시작한다.

● position.user_msg_len = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 UI에 표현할 전문 데이터의 길이를 등록한다.

● position.chnl_code_pos = numeric (Version 4.0.3 이상)
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 채널코드 시작 위치를 등록한다.
- 시작위치는 1부터 시작한다.

● position.chnl_code_len = numeric (Version 4.0.3 이상)
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 전문 내에서 채널코드 길이를 등록한다.
- 수집한 채널코드는 UI의 메타 정보에 표현되면 또한 채널코드별 통계용으로 활용할 수 있다.

3.2.9. 에러위치 정보

버퍼 타입이 STRING, CARRAY인 경우에 거래처리 결과가 정상인지 에러인지를 판단하기 위하여 전문

내에서 에러코드 및 메시지 위치 정보를 등록한다. 하지만 현재 사용하지 않는다.

● 에러 유무를 판단할 위치나 코드는 무엇인가?
● 에러코드 위치는 어디인가?

에러위치 정보의 환경설정 형식은 다음과 같다.

```
##############################################################################
# Application error information setting
##############################################################################
[app_errinfo]
# 에러전문 판단정보. 메시지판단 위치정보, 메시지판단 코드길이, 에러전문코드 값
# 위치정보는 1부터 시작
#errinfo.errmsg_pos=221
#errinfo.errmsg_len=1
#errinfo.errmsg_code=2

# 에러코드 위치 정보 (위치는 1부터 시작)
#errinfo.code_pos=771
#errinfo.code_len=10

# 에러메시지 위치 정보 (위치는 1부터 시작)
#errinfo.msg_pos=781
#errinfo.msg_len=40
```

[그림] 3-12 pharostp.cfg – 에러위치 정보

선택항목

● errinfo.errmsg_pos = numeric
  - 기본값 : 0
  - 버퍼 타입이 CARRAY나 STRING인 경우 응답 전문 내에서 에러 유무를 판단하기 위한 코드의 시작 위치를 등록한다.
  - 시작위치는 1부터 시작한다.

● errinfo.errmsg_len = numeric
  - 기본값 : 0
  - 버퍼 타입이 CARRAY나 STRING인 경우 응답 전문 내에서 에러 유무를 판단하기 위한 코드의 길이를 등록한다.

● errinfo.errmsg_code = string
  - 범위 : 60자 이내

- 버퍼 타입이 CARRAY나 STRING인 경우 응답 전문 내에서 에러 유무를 판단하기 위한 에러코드를 등록한다. 위의 항목에서 정의한 코드가 동 항목에 등록된 코드와 일치하면 에러로 판단한다.
- 등록된 값은 에러로 판단한다.

● errinfo.code_pos = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 응답 전문 내에서 에러코드 시작 위치를 등록한다.
- 시작위치는 1부터 시작한다.

● errinfo.code_len = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 응답 전문 내에서 에러코드 길이를 등록한다

● errinfo.msg_pos = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 응답 전문 내에서 에러메시지 시작 위치를 등록한다
- 시작위치는 1부터 시작한다.

● errinfo.msg_len = numeric
- 기본값 : 0
- 버퍼 타입이 CARRAY나 STRING인 경우 응답 전문 내에서 에러메시지 길이를 등록한다

3.2.10. 로그 정보

서비스 종료 로그, TP Monitor 시스템 로그, CORE 파일 위치와 같이 각종 로그에 대한 정보를 등록한다.

● 서비스 종료 로그를 남길 것인가?
● CORE 파일 위치를 변경할 것인가?

로그 정보의 환경설정 형식은 다음과 같다.

```
#############################################################################
# Trace Logging information setting
#############################################################################
[log_info]

# 서비스 종료 로그 출력 여부
log.log_flag=false

# Hooking 에러로그 로깅 path
#log.log_path=/home/PharosTP/apm/log

# TP Monitor sys log file name
#log.sys_log_path=/home/tmax/tmax5/log/slog/slog.$(MMDDYYYY)

# |=or, &=and, !=not ex) ERROR|WARN|!tms
log.sys_log_pattern=CLH

# interval for log file check (millisecond)
log.sys_log_interval=5000

# Core File info
log.core_file_flag=false
log.core_file_path=/home/PharosTP/apm/log/corelog
```

[그림] 3-13 pharostp.cfg – 로그 정보

선택항목

● log.log_flag = true | false
- 기본값 : false
- 서비스 처리가 종료된 경우 종료 정보를 에이전트 로그에 남길 것인지 여부를 등록한다.
- 단순한 디버그 용으로 사용한다.
- true : 로그 남김
- false : 로그 남기지 않음

● log.log_path = string
- 범위 : 126자 이내
- Hooking 모듈에서 디버그 로그를 저장하기 위한 path를 등록한다. 단순한 디버그 용으로 사용한다.

● log.sys_log_path = string
- 범위 : 126자 이내

- TP Monitor 시스템 로그 파일에 대한 Full Path를 등록한다. 동 항목을 등록하면 시스템 로그를 주기적으로 조사하여 로그 패턴 항목「log.sys_log_pattern」에 정의되어 있는 정보에 따라서 해당 로그 정보를 UI에 표현한다.
- Ex)/home/PharosTP/apm/log/slog/slog.$(MMDDYYYY)

● log.sys_log_pattern = string
- 범위 : 1020자 이내
- TP Monitor 시스템 로그 파일에 대한 Tail 기능 처리시 조건에 합당한 로그만 출력할 수 있도록 패턴 정보를 등록한다 패턴 정보는 AND(&)나 OR(|), NOT(!)을 사용하여 등록할 수 있다.
- 패턴을 등록하지 않으면 모든 로그를 수집서버에 전송한다.
- Ex) ERROR|WARN|!tms (ERROR나 WARNING 중에서 "tms" 스트링으로 시작하지 않은 로그 메시지)

● log.sys_log_interval = numeric
- 범위 : 0 ~ MAX_INT (밀리초)
- 기본값 : 없음
- TP Monitor 시스템 로그 파일 Tail 간격을 밀리초 단위로 등록한다.

● log.core_file_flag = true | false
- 기본값 : false
- 서버 프로세스가 비정상적으로 종료되는 경우에 CORE 파일을 지정된 디렉터리로 옮길 것인지 여부를 등록한다.「true」로 등록하면 반드시「log.core_file_path」항목을 등록해야 한다.
- true : CORE 파일 Path 지정
- false : CORE 파일 Path 지정하지 않음

● log.core_file_path = string
- 범위 : 126자 이내
- CORE 파일이 생성될 위치를 등록한다
- Ex)/home/PharosTP/apm/log/corelog

3.2.11. 공유메모리 정보

성능정보 추출을 위하여 환경 정보나 또는 거래추적 정보를 일시적으로 저장할 공유메모리 정보를 등록한다.

● 공유메모리 키를 무엇으로 할 것인가?
● 서버 프로세스 수는 얼마인가?

공유메모리 정보의 환경설정 형식은 다음과 같다.

```
###############################################################################
# Monitoring Global Area information setting
###############################################################################
[mga_info]

# APM 정보 저장을 위한 공유메모리 키 값 [default=78437]
mga.shmkey=numeric

# 거래추적 대상 서버프로세스 수  [default=512]
mga.svr_process=numeric

# 서비스 단위별 거래추적 총 호출 건수  [max=1024]
mga.call_seq=256

# SQL TEXT 저장 개수  [default=10000]
mga.sql_hash_num=100000
```

[그림] 3-14 pharostp.cfg – 공유메모리 정보

필수항목

● mga.shmkey = numeric
- 범위 : 1 ~ MAX_INT
- 기본값 : 78437
- 성능정보 추출을 위해서 필요한 정보 및 Active Call-Tree 정보를 저장할 공유메모리의 키를 등록한다. 동 키 값은 현재 시스템에서 사용하지 않는 키 값 이어야 한다.

참고

공유메모리 키는 동 항목에 정의한 key를 기준으로 2개가 생성된다. 그러므로 key 값을 정의할 때 연속해서 2개를 정의할 수 있는 시작 key를 정의해야 한다

● mga.svr_process = numeric
- 범위 : 512 ~ 10000
- 기본값 : 512
- TP Monitor 환경에서 실행되는 서버 프로세스 중 성능정보 추출 대상 서버 프로세스 최대 수를 정의한다.
- TP Monitor 환경 파일에 정의된 서버 프로세스의 MAX 항목 값의 합보다 커야 한다.

● mga.call_seq = numeric
- 범위 : 1 ~ 1024

- 기본값 : 1
- 하나의 서비스에서 내부적으로 호출할 모듈, ATMI call, SQL 문 등의 최대 Call-Tree 개수를 등록한다.
- 동 항목에 정의된 개수보다 많은 호출을 하는 정보는 자동적으로 삭제된다.

● mga.sql_hash_num = numeric
- 범위 : 10000 ~ 100000
- 기본값 : 10000
- SQL 문의 해쉬코드를 에이전트 내부적으로 몇 개까지 저장할 것인지 건수를 등록한다.
- 동 항목에 정의된 개수보다 SQL문이 많으면 LRU 알고리즘에 따라 가장 오랫동안 사용하지 않은 해쉬코드를 삭제하고 새로운 해쉬코드를 저장한다

3.2.12. 연계서비스 정보

거래추적에서 타 시스템과 연동하는 AnyLink을 하나의 트랜잭션으로 묶기 위해 AnyLink 서비스를 통하여 다른 시스템과 연계할 때 연계에 사용되는 AnyLink 서비스를 등록한다.

● 연계서비스는 있는가?

연계서비스 정보의 환경설정 형식은 다음과 같다.

```
##############################################################################
# Monitoring System Link Service information setting
##############################################################################
[link_service]

# AnyLink call service
linksvc.anylink_svc=
```

[그림] 3-15 pharostp.cfg – 연계서비스 정보

선택항목

● linksvc.anylink_svc = string
- 범위 : 100개 서비스 이내
- AnyLink 서비스 명을 등록한다. 서비스가 여러 개인 경우에는 콤마로 분리하여 등록한다.
- 거래추적을 위해서는 하나의 글로벌 ID로 서로간의 호출관계를 표현해야 한다. 그러나 AnyLink는 Hooking 방식을 사용할 수 없고 AnyLink에서 남기는 로그를 기준으로 처리하기 때문에 APM 헤더를 사용하여 서비스간 호출관계를 표현할 수 없다.
- 그래서 AnyLink 서비스인 경우에는 호출관계를 표현하는 방법은 "mon.hashcode_type" 에서 지정한 방식에 따라서 호출관계를 표현한다.
- AnyLink 시스템을 거래추적에 포함할 경우에는 동 서비스에 AnyLink 서비스명을 등록하고 또한 "mon.hashcode_type" 항목에 호출관계를 표현하기 위한 방법을 등록해야 한다.

3.2.13. Advice

Advice 카테고리는 사이트별로 사용자가 서비스 마다 커스터마이징 할 수 있도록 사용자 라이브러리 명을 등록하는 절이다. 거래추적에서 사이트마다 연계하는 방법이 다를 수도 있고, 또는 특정 서비스의 전문에서 특정 데이터를 추출하여 UI의 메타정보를 표현하거나 할 수도 있다. 이처럼 사이트마다 별도로 추가적인 처리를 위해서 사용자가 작성한 프로그램을 라이브러리화 하여 등록하는 카테고리이다. 사용자 라이브러리 생성 방법은 부록 "G. Advice 프로그램 작성 방법"을 참조한다.

Advice 정보의 환경설정 형식은 다음과 같다.

```
##########################################################################
# Site advice library information setting
##########################################################################
[advice]

# 사이트별 라이브러리 파일명 : 절대경로 방식으로 등록
advice.lib_name=
```

[그림] 3-16 pharostp.cfg – Advice 라이브러리 등록

선택항목

● advice.lib_nbame = string
- 범위 : 250자 이내
- 사용자 커스터마이징 라이브러리 명을 등록한다. 등록할 라이브러리 명은 절대경로 방식으로 등록한다.
- 추가적인 처리가 필요하지 않으면 등록하지 않는다. 동 항목에 등록된 경우에만 사용자 함수를 호출하기 때문에 등록하지 않으면 사용자 함수는 호출하지 않는다.

3.2.14. 거래코드 정보 등록

「mon.txcode_flag」 항목에 등록된 정보에 따라서 아래에 등록하는 거래코드가 성능정보 추출 대상에 포함될 것인지 아니면 제외될 것인지를 판단한다. 거래코드 등록 방법은 포함 여부에 관계없이 동일하다.

● 성능정보 추출에서 제외할 거래코드가 있는가?

거래코드 파일 정보의 환경설정 형식은 다음과 같다.

```
################################################################################
# Transaction code for skip list
################################################################################
[code_list]
txcode=ABDC1
txcode=ABDC2
txcode=ABDC3
txcode=ABDC4
txcode=ABDC5
```

[그림] 3-17 txcode.cfg – 거래코드 파일 정보

선택항목

● tx_code : string
- 범위 : 500개 거래코드 이내
- 전문 상에서 추출할 수 있는 거래코드를 등록한다.

참고

동 항목을 등록한 경우에는 반드시 전문 내에서 거래코드를 구할 수 있는 방법을 등록해야 한다. TP 버퍼 타입이 STRING이나 CARRAY인 경우 [position] 절의 txcode_pos, txcode_len 항목의 값을 반드시 등록해야 한다.

3.2.15. 데몬 프로세스 정보 등록

TP Monitor 환경에서 TP에 등록된 서버 프로세스 이 외에 데몬 형식으로 실행되는 프로세스에 대해 성능 정보를 수집하기 위해서 데몬 프로세스의 정보를 등록한다. 데몬 프로세스를 등록하는 경우에는 반드시 데몬 프로세스에서 호출하는 함수들에 대해서 함수의 형식을 등록하고 소스를 생성하여 공유 라이브러리를 만들어야 한다. 데몬 프로세스에서 호출하는 함수는 반드시 공유 라이브러리에 있는 함수들이어야 한다. 이에 대한 자세한 사항은 "5. 데몬 프로세스 성능정보 추출 방법"을 참조한다.

● 데몬 프로세스가 존재하는가?

데몬 프로세스 정보의 환경설정 형식은 다음과 같다.

```
################################################################################
# TP DAEMON SERVER process config information file
################################################################################
[Dpush]
alias=Dpush
txcode_pos=1287
txcode_len=8
tcp_recv_ip=175.118.115.24
tcp_recv_port=30004
tcp_send_ip=175.118.115.24
tcp_send_port=30003
recv_msgq_sekey=0x30000200-0x30000228
recv_msgq_key=0x3000005e
send_msgq_sekey=0x31000001-0x3100270f
send_msgq_key=0x3000006e
mq_send_file=sname
mq_recv_file=rname
follow_process=mciclient
command_arg=Dpush -i 175.118.115.24 -p 30003
```

[그림] 3-18 데몬 프로세스 파일 정보

위의 파일에서 모든 프로세스마다 모든 정보를 등록하는 것이 아니라 해당 프로세스에서 사용하는 항목만 등록한다. 예를 들어 메시지 큐를 읽어서 TCP로 전송하는 데몬 프로세스인 경우에는 아래와 같이 등록한다.

```
[Dpush]
alias=Dpush
txcode_pos=1287
txcode_len=8
tcp_send_ip=175.118.115.24
tcp_send_port=30003
recv_msgq_sekey=0x30000200-0x30000228
recv_msgq_key=0x3000005e
follow_process=mciclient
command_arg=-i 175.118.115.24 -p 30003
```

선택항목

- alias : string
  - 범위 : 46자 이내

- 데몬 프로세스 명 대신에 별명을 등록하고자 하는 경우에 별명을 등록한다. 동 항목을 등록하면 UI 상의 Call-Tree에 해당 명이 서버명과 서비스 명으로 표현된다. 동 항목은 동일한 데몬 프로세스가 여러 개인 경우에 이를 구분하기 위해서 사용한다.

● txcode_pos = numeric
- 기본값 : 0
- 전문 내에서 거래코드의 시작 위치를 등록한다. 추출할 거래코드가 없는 경우에는 등록하지 않는다.
- 시작위치는 1부터 시작한다.

● txcode_len = numeric
- 기본값 : 0
- 거래코드 길이를 등록한다. 추출할 거래코드가 없는 경우에는 등록하지 않는다.

● tcp_recv_ip : string
- 범위 : 30자 이내
- 데몬 프로세스가 TCP로 메시지를 수신 받는 경우에 상대방 프로세스가 실행되는 서버의 IP 주소를 등록한다. TCP로 수신 받는 프로세스가 아닌 경우에는 등록하지 않는다.

● tcp_recv_port : numeric
- 범위 : 1000 ~ MAX_INT
- 데몬 프로세스가 TCP로 메시지를 수신 받는 경우에 상대방 프로세스와 연결할 포트번호를 등록한다. TCP로 수신 받는 프로세스가 아닌 경우에는 등록하지 않는다.

● tcp_send_ip : string
- 범위 : 30자 이내
- 데몬 프로세스가 상대방 프로세스에게 TCP로 메시지를 송신하는 경우에 상대방 프로세스가 실행되는 서버의 IP 주소를 등록한다. TCP로 송신하지 않는 경우에는 등록하지 않는다.

● tcp_send_port : numeric
- 범위 : 1000 ~ MAX_INT
- 데몬 프로세스가 TCP로 메시지를 송신하는 경우에 상대방 프로세스와 연결할 포트번호를 등록한다. TCP로 송신하지 않는 경우에는 등록하지 않는다.

● recv_msgq_sekey : numeric
- 범위 : 1 ~ MAX_INT
- 데몬 프로세스가 메시지 큐로부터 메시지를 읽는 경우에 메시지 큐의 키를 등록한다. 동 항목은 메시지 큐가 연속으로 여러 개인 경우에 시작번화 끝번호를 등록한다. 등록하는 키 값을 16진수 형식으로 등록해야 한다.
  예) recv_msgq_sekey=0x30000200-0x30000228

● recv_msgq_key : numeric
- 범위 : 1 ~ MAX_INT
- 데몬 프로세스가 메시지 큐로부터 메시지를 읽는 경우에 메시지 큐의 키를 등록한다. 동 항목은 메시지 큐가 연속되지 않고 하나씩인 경우에 등록한다. 등록하는 키 값을 16진수 형식으로 등록해야 한다.
  예) recv_msgq_key=0x3000025e,0x30000345

● send_msgq_sekey : numeric
- 범위 : 1 ~ MAX_INT
- 데몬 프로세스가 메시지 큐에 메시지를 저장하는 경우에 메시지 큐의 키를 등록한다. 동 항목은 메시지 큐가 연속으로 여러 개인 경우에 시작번화 끝번호를 등록한다. 등록하는 키 값을 16진수 형식으로 등록해야 한다.
  예) send_msgq_sekey=0x30000200-0x30000228

● send_msgq_key : numeric
- 범위 : 1 ~ MAX_INT
- 데몬 프로세스가 메시지 큐에 메시지를 저장하는 경우에 메시지 큐의 키를 등록한다. 동 항목은 메시지 큐가 연속되지 않고 하나씩인 경우에 등록한다. 등록하는 키 값을 16진수 형식으로 등록해야 한다.
  예) send_msgq_key=0x3000025e,0x30000345

● mq_send_file : numeric
- 범위 : 46자 이내
- 데몬 프로세스가 mq_send 명령어를 이용하여 큐에 메시지를 저장하는 경우에 큐의 명을 등록한다. 큐명이 여러 개인 경우에 Comma로 구분한다.
  예) mq_send_file=sfile1,sfile2

● mq_recv_file : numeric
- 범위 : 46자 이내
- 데몬 프로세스가 mq_receive 명령어를 이용하여 큐로부터 메시지를 읽는 경우에 큐의 명을 등록한다. 큐명이 여러 개인 경우에 Comma로 구분한다.
  예) mq_recv_file=sfile1,sfile2

● follow_process : string
- 범위 : 15자 이내
- 데몬 프로세스가 메시지를 전달할 프로세스의 서비스 명을 등록한다. 상대방 프로세스가 데몬 프로세스인 경우에는「alias」항목에 등록된 명칭을 등록한다. 동 항목에 등록된 명칭은 Call-Tree 상의 호출 서비스 명으로 표현된다.

● command_arg : string

- 범위 : 126자 이내
- 동일한 이름의 데몬 프로세스가 여러 개인 경우에 이를 구분할 수 있는 Command 아규먼트를 등록한다.

04

사용자 함수 성능정보 추출 방법

4. 사용자 함수 성능정보 추출 방법

4.1. 사용자 함수 등록

사용자 함수 중 성능정보 추출 대상에 대해서 먼저 config 파일을 생성해야 한다. config 파일 등록 방법은 아래와 같다.

```
[function]

# 사용자 함수명
function.name=string

# 호출할 함수명
function.callname_arg=string

# 함수의 리턴 타입
function.returntype=string

# 함수 형식
function.prototype=string
```

[그림] 4-1 customgen.cfg – 사용자 함수 등록

필수항목

● function.name = string
- 범위 : 64자 이내
- 사용자가 공통으로 사용할 목적으로 작성한 함수명을 등록한다. 동 함수는 반드시 공유 라이브러리에서 제공하는 함수이어야 한다.
- 함수명은「function.prototype」항목에서 등록하는 함수명과 같아야 한다.

● function.returntype = string
- 범위 : 16자 이내
- 함수의 리턴 타입을 등록한다.
- 리턴 타입으로는 short, int, long, void 네 종류가 있다.

● function.prototype = string
- 범위 : 1024자 이내
- 함수의 형식을 등록한다.

선택항목

● function.callname_arg = string

- 범위 : 128자 이내
- 해당 함수가 다른 함수를 호출하는 공통 함수인 경우 호출할 함수명이 저장된 입력 아규먼트 명을 등록한다.

4.2. 사용자 모듈 생성

사용자 함수는 성능 정보를 추출하기 위하여 소스를 생성하는 방식으로 처리하기 때문에 위에서 설명한 사용자 함수들에 대해서 환경설정을 한 다음에 Pharos TP 시스템에서 제공하는 명령어를 통하여 Hooking 모듈을 생성해야 한다.

참고

사용자 함수 Config 파일은 반드시 $PHOMEDIR/config에 존재해야 한다.

4.2.1. 소스 생성 방법

Hooking 모듈은 명령어 프롬프트 상태에서 pcustomgen 명령어를 통하여 생성한다.

```
PharosTP@AIX6:/home/PharosTP/apm/bin> pcustomgen -i customgen.cfg
```

-i 옵션은 사용자 함수 Config 파일 명을 입력한다.

4.2.2. 생성된 소스

pcustomgen 명령어를 통하여 생성되는 소스는 $PHOMEDIR/work에 생성된다.

config 정보

```
# custommer function info
[function]
function.name=tdlcall
function.callname_arg=funcname
function.returntype=int
function.prototype=tdlcall(char *funcname, void *args, long *urcode, int flags)
```

```
 [function]
function.name=tdlcall2
function.callname_arg=funcname
function.returntype=int
function.prototype=tdlcall2(char *libname, char *funcname, void *args, long *urcode, int flags)

[function]
function.name=tdlcall2s
function.callname_arg=funcname
function.returntype=int
function.prototype=tdlcall2s(char *libname, char *funcname, void *input, void *output, long *urcode,
int flags)
```

[그림] 4-2 customgen.cfg – 사용자 함수 등록

참고

첫 번째 [function] 에서

function.name의 값은 function.prototype의 함수 이름과 같아야 하고,

Ex) function.name=**usr_module** ➔ function.prototype=**usr_module**(char *data, long len, long flag)=usr_module(char *data, long len, long flag)

function.callname_arg의 값은 function.prototype의 첫 번째 파라미터인 char* funcname과 같아야 한다.

Ex) function.callname_arg=**funcname** ➔ function.prototype=tdlcall(char ***funcname**, void *args, long *urcode, int flags)

Hooking 소스

```
/*
 * @file                customhook.c
 * @brief               Custommer hooking source
 * @author
 * @library
 *
 * @history
 *
 *  성     명  :  일     자  :              변        경        내        용
 *  --------      --------      ------------------------------------------------------------------
 *
 */

/*
 *
 * @pseudo code
 *
 */

/* Linux requires _GNU_SOURCE macro for RTLD_NEXT */
#if defined(__linux) || defined(linux) || defined(Linux) || defined(_LINUX) || defined(_LINUX64)
#    ifndef _GNU_SOURCE
#        define _GNU_SOURCE
#    endif
#endif

/* ------------------------------------- include files ------------------------------------- */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <signal.h>
#include <fcntl.h>
#include <unistd.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```
#include <sys/timeb.h>
#include <stdarg.h>
#include <math.h>
#include <strings.h>
#include <sys/param.h>
#include <sys/time.h>
#include <time.h>
#include <dlfcn.h>
#include <pthread.h>


/* ---------------------------------------- constant ---------------------------------------- */
/* ------------------------------------- macro definitions ---------------------------------- */

/* value return type function hook */
#define BEGIN_FUNCTION_HOOK_DECLARE(RETTYPE, NAME, ...)
typedef RETTYPE (*NAME##_hook_t)(__VA_ARGS__);
RETTYPE NAME(__VA_ARGS__)
{
    static NAME##_hook_t NAME##_hook = NULL;
    RETTYPE hook_rc;

    if (NAME##_hook == NULL ) {
        NAME##_hook = (NAME##_hook_t)dlsym(RTLD_NEXT, #NAME);
        if (NAME##_hook == NULL) {
            fprintf(stderr,"ERROR: unsatisfied symbol [%s]₩n", #NAME);
            exit(1);
        }
    }

#define END_FUNCTION_HOOK_DECLARE
    return hook_rc;
}
```

```
#define ORIGIN_FUNCTION_CALL_RETURN(NAME, ...)
	do {
		hook_rc = (*(NAME##_hook))(__VA_ARGS__);
	} while(0)

#define ORIGIN_FUNCTION_CALL_RETURN_NOARGS(NAME)
	do {
		hook_rc = (*(NAME##_hook))();
	} while(0)

/* void return type function hook */
#define BEGIN_FUNCTION_HOOK_DECLARE_VOID(NAME, ...)
typedef void (*NAME##_hook_t)(__VA_ARGS__);
void NAME(__VA_ARGS__)
{
	static NAME##_hook_t NAME##_hook = NULL;

	if (NAME##_hook == NULL ) {
		NAME##_hook = (NAME##_hook_t)dlsym(RTLD_NEXT, #NAME);
		if (NAME##_hook == NULL) {
			fprintf(stderr,"ERROR: unsatisfied symbol [%s]₩n", #NAME);
			exit(1);
		}
	}

#define END_FUNCTION_HOOK_DECLARE_VOID
	return;
}

#define ORIGIN_FUNCTION_CALL_VOID(NAME, ...)
	do {
		(*(NAME##_hook))(__VA_ARGS__);
	} while(0)

#define ORIGIN_FUNCTION_CALL_VOID_NOARGS(NAME)
	do {
		(*(NAME##_hook))();
	} while(0)
```

```
/* ------------------------------ structure definitions ---------------------------------------- */
/* ---------------------------- exported global variables declarations ------------------------- */
/* -------------------------------- exported function declarations ---------------------------- */
extern long custom_function_start(char *);
extern long custom_function_end(char *, int);


/* ------------------------------------------------------------------------------------------

 * @function            int test_biz_module1()
 * @brief               test_biz_module1 API
 * @parameter
 * @return              0    : 정상
 *                      -1   : 에러
 * ------------------------------------------------------------------------------------------
 */
BEGIN_FUNCTION_HOOK_DECLARE(int, test_biz_module1, char *funcname)
{
    int rc = 0;

    /* ------------------------------------------------------------------- */
    /* user function call                                                  */
    /* ------------------------------------------------------------------- */
    custom_function_start(funcname);

    /* call the original */
    ORIGIN_FUNCTION_CALL_RETURN(test_biz_module1, funcname);

    /* ------------------------------------------------------------------- */
    /* user end function call                                              */
    /* ------------------------------------------------------------------- */
    custom_function_end(funcname, hook_rc);
}
END_FUNCTION_HOOK_DECLARE
```

```
/* ------------------------------------------------------------------------------------

 * @function          int usr_module()
 * @brief             usr_module API
 * @parameter
 * @return            0    : 정상
 *                    -1   : 에러
 * ------------------------------------------------------------------------------------
 */
BEGIN_FUNCTION_HOOK_DECLARE(int, usr_module, char *data, long len, long flag)
{
    int rc = 0;

    /* ------------------------------------------------------------------ */
    /* user function call                                                 */
    /* ------------------------------------------------------------------ */
    custom_function_start("usr_module");

    /* call the original */
    ORIGIN_FUNCTION_CALL_RETURN(usr_module, data, len, flag);

    /* ------------------------------------------------------------------ */
    /* user end function call                                             */
    /* ------------------------------------------------------------------ */
    custom_function_end("usr_module", hook_rc);
}
END_FUNCTION_HOOK_DECLARE
```

[그림] 4-3 customhook.c

4.2.3. 생성된 makefile

생성된 소스를 컴파일 할 수 있도록 Makefile도 해당 시스템에 맞게 자동으로 생성된다. 예를 들어 AIX 시스템인 경우 아래와 같이 생성된다.

```
#=================================================================#
# customhook makefile
#=================================================================#
BASE       = customhook
TARGET    = lib$(BASE)$(OS_SO_SUFFIX)

# ---------------------------------------------------------------------------------------------- #
# Objects
# ---------------------------------------------------------------------------------------------- #
OBJS       = customhook.o

# ---------------------------------------------------------------------------------------------- #
# MY DEFINE
# ---------------------------------------------------------------------------------------------- #
MY_C_DEFINE = -DNO_DEBUG
MY_PROC_DEFINE    =
MY_IPATH       = -I. $(TP_IPATH)
MY_CFLAGS     =
MY_LDPATH     = -L. -L$(PHOMEDIR)/lib
MY_LDLIBS     = -lpharostphm -lpharostpcom -lpharos $(OS_THRLIBS)
# ---------------------------------------------------------------------------------------------- #
# OS Environment
# ---------------------------------------------------------------------------------------------- #
OS_CFLAGS           = -qlanglvl=extc99 -q64 -qinfo=pro -brtl -O2 -qcpluscmt
OS_CDEFINE          = -D_IBM64 -D_REENTRANT
OS_IPATH            =
OS_LDFLAGS          = -qlanglvl=extc99 -brtl
OS_LDPATH           =
OS_OSLIBS           = -lc -lm
OS_LDLIBS           =
OS_THRLIBS          = -lpthread
OS_MONLIBS          =
OS_PMONLIBS         = -lperfstat
OS_SO_FLAGS         = -G -bshared -q64
OS_SO_SUFFIX        = .so
OS_COMPILER         = cc
```

```
# ------------------------------------------------------------------------------------------- #
# APM Environment
#   ------------------------------------------------------------------------------------------- #
APM_CDEFINE        = -DNO_DEBUG
APM_CFLAGS         =
APM_IPATH          = -I$(PHOMEDIR)/inc -I$(PHAROS_TPHOME)/inc
APM_LDPATH         = -L$(PHOMEDIR)/lib
APM_LDLIBS         = -lPharos -lPharostpcom

###############################Compile Options################################
# ------------------------------------------------------------------------------------------- #
# C Compiler Flag
# ------------------------------------------------------------------------------------------- #
CC                 = $(OS_COMPILER)
C_FLAGS            = $(MY_CFLAGS)     $(APM_CFLAGS)    $(OS_CFLAGS)
C_DEFINE           = $(MY_C_DEFINE) $(APM_CDEFINE) $(OS_CDEFINE)
I_FLAGS            = $(MY_IPATH)      $(APM_IPATH)     $(OS_IPATH)
CFLAGS             = $(C_FLAGS)       $(C_DEFINE)      $(I_FLAGS)

# ------------------------------------------------------------------------------------------- #
# LD Compiler Flag
# ------------------------------------------------------------------------------------------- #
LD         = $(OS_COMPILER)
LD_PATH        = $(MY_LDPATH) $(APM_LDPATH) $(TP_LDPATH) $(DATABASE_LDPATH) $(OS_LDPATH)
LD_LIBS        = $(MY_LDLIBS) $(OS_LDLIBS)
LD_FLAGS       = $(OS_LDFLAGS) $(LD_PATH)    $(LD_LIBS)

# ------------------------------------------------------------------------------------------- #
# Compile suffix rule
# ------------------------------------------------------------------------------------------- #
.SUFFIXES : .c .o

.c.o:
      $(CC) $(CFLAGS) -c $<

# ------------------------------------------------------------------------------------------- #
# Dependency
# ------------------------------------------------------------------------------------------- #
```

```
all: $(TARGET)

$(TARGET): $(OBJS)
	$(LD) $(OS_SO_FLAGS) -o $@ $(OBJS) $(LD_FLAGS)
	mv $(TARGET) $(PHOMEDIR)/lib

mv:
	mv $(TARGET) $(PHOMEDIR)/lib

clean:
	-rm -f core $(OBJS) $(TARGET)

#=================================================================#
# Make Dependency File
#=================================================================#
```

[그림] 4-4 customhook.mk

4.2.4. 모듈 생성

생성된 소스를 컴파일하여 모듈(공유 라이브러리)를 생성해야 한다.

모듈을 생성하는 방법은 아래와 같다.

```
PharosTP@AIX6:/home/pharostp/apm/work> make -f customhook.mk
        cc -G -bshared -q64 -o libcustomhook.so customhook.o -qlanglvl=extc99 -brtl -L. -L/home/pharostp/apm/lib -lpharostphm -lpharostpcom -lpharos -lpthread
        mv libcustomhook.so /home/pharostp/apm/lib
Target "all" is up to date.
PharosTP@AIX6:/home/pharostp/apm/work> cd ../lib
PharosTP@AIX6:/home/pharostp/apm/lib> ls -al
total 1840
drwxr-x---    2 PharosTP dba        4096 Aug  2 18:05 .
drwxr-xr-x   10 PharosTP dba         256 Aug  2 16:39 ..
-rwxr-xr-x    1 PharosTP dba        5773 Aug  2 18:05 libcustomhook.so
-rwxr-xr-x    1 PharosTP dba      171197 Jun 22 16:00 libpharos.so
-rwxr-xr-x    1 PharosTP dba      180896 Jun 22 16:00 libpharosdb.so
-rwxr-xr-x    1 PharosTP dba      103951 Jun 22 16:00 libpharostpcom.so
-rwxr-xr-x    1 PharosTP dba      125506 Jun 22 16:00 libpharostphm.dbg.so
-rwxr-xr-x    1 PharosTP dba      113189 Jun 22 16:00 libpharostphm.so
-rwxr-xr-x    1 PharosTP dba       31790 Jun 22 16:00 libpharostpsh.dbg.so
```

```
-rwxr-xr-x    1 PharosTP dba        29800 Jun 22 16:00 libPharostpsh.so
-rwxr-xr-x    1 PharosTP dba       112525 Jun 22 16:00 libpsysrmon.dbg.so
-rwxr-xr-x    1 PharosTP dba        37025 Jun 22 16:00 libpsysrmon.so
-rwxr-xr-x    1 PharosTP dba         1575 Jun 22 16:00 libsigusrproc.so
```

4.3. 사용자 모듈 적용

생성된 Hooking모듈을 적용하고 TP Monitor 시스템을 재 시작해야 적용된다. 적용하는 방법은 TP Monitor에 따라 다르게 적용해야 한다.

4.3.1. Tuxedo 서버 적용 방법

아래의 환경변수 설정 방법은 Tuxedo 시스템인 경우에 사용하는 방법이다.

<.profile> / <.bash_profile>

```
# export preload
if [ $OS_VENDER = ibm64 ]; then
  export LDR_PRELOAD64=$PHOMEDIR/lib/libpharostpsh.so:$PHOMEDIR/lib/liblibcustomhook.so
elif [ $OS_VENDER = ibm32 ]; then
      export LDR_PRELOAD=$PHOMEDIR/lib/libpharostpsh.so:$PHOMEDIR/lib/liblibcustomhook.so
else
      export LD_PRELOAD=$PHOMEDIR/lib/libpharostpsh.so :$PHOMEDIR/lib/liblibcustomhook.so
fi
```

<.cshrc>

```
# export preload
if [ $OS_VENDER = ibm64 ]; then
      setenv LDR_PRELOAD64 $PHOMEDIR/lib/libpharostpsh.so:$PHOMEDIR/lib/liblibcustomhook.so
elif [ $OS_VENDER = ibm32 ]; then
      setenv LDR_PRELOAD $PHOMEDIR/lib/libpharostpsh.so:$PHOMEDIR/lib/liblibcustomhook.so
else
      setenv LD_PRELOAD $PHOMEDIR/lib/libpharostpsh.so:$PHOMEDIR/lib/liblibcustomhook.so
fi
```

[그림] 4-5 Tuxedo – 사용자 모듈 적용

4.3.2. Tmax 서버 적용 방법

Tmax 시스템인 경우에는 PRELOAD 환경변수는 등록하면 안 되고, 대신 Link 옵션에 Hooking 모듈 libpharostpsh.so 다음에 libcustomhook.so를 추가하면 된다.

05

데몬 프로세스 성능정보 추출 방법

5. 데몬 프로세스 성능정보 추출 방법

5.1. 데몬 프로세스 함수 등록

데몬 프로세스의 성능정보를 추출하기 위하여 데몬 프로세스에서 호출하는 함수들에 대해서 먼저 config 파일을 생성해야 한다. 데몬 프로세스에서 호출하는 함수들은 반드시 공유 라이브러리에서 제공하는 함수들 이어야 한다. config 파일 등록 방법은 아래와 같다.

```
# tp daemon custom function info
[function]
function.type=SVRSTART
function.name=init_apm
function.returntype=int
function.prototype=init_apm(void)
function.prefunc=server_start_prefix()
function.postfunc=server_start_postfix()

 [function]
function.type=CUSTOM
function.name=RecvFromBKFep
function.returntype=int
function.prototype=RecvFromBKFep(char *psfepid, char *psmsg, int *pimsglen, int *piseq)
function.prefunc=recvfrombkfep_prefix(char *funcname, char *psmsg, int *pimsglen)
function.postfunc=recvfrombkfep_postfix(char *funcname, char *psmsg, int *pimsglen, long hook_rc)

[function]
function.type=RECVMQ
function.name=mq_receive
function.returntype=ssize_t
function.prototype=mq_receive(mqd_t mqdes, char *msg_ptr, size_t msg_len, unsigned *msg_prio)
function.prefunc=mq_recv_prefix(char *funcname, mqd_t mqdes, char *msgp, long msg_len)
function.postfunc=mq_recv_postfix(char *funcname, mqd_t mqdes, char *msgp, long hook_rc, long &hlen)
```

[그림] 5-1 daecustomgen.cfg – 데몬 프로세스 함수 등록

필수항목

- function.type = string

- 범위 : 16자 이내
- 함수마다 처리하는 방식이 다르기 때문에 데몬 프로세스에서 호출하는 함수의 종류를 등록한다. 등록된 타입에 맞게 Hooking 모듈을 생성한다.

타입	의미
SVRSTART	데몬 프로세스 시작 함수를 의미함
CUSTOM	데몬 프로세스에서 호출하는 함수가 사용자가 작성한 공유 라이브러리에서 제공하는 함수를 의미함
RECVMSGQUE	데몬 프로세스에서 호출하는 함수가 msgrcv 함수를 의미함
SENDMSGQUE	데몬 프로세스에서 호출하는 함수가 msgsnd 함수를 의미함
RECVMQ	데몬 프로세스에서 호출하는 함수가 mq_receive 함수를 의미함
SENDMQ	데몬 프로세스에서 호출하는 함수가 mq_send 함수를 의미함
SENDTCP	데몬 프로세스에서 호출하는 함수가 TCP send 함수를 의미함
RECVTCP	데몬 프로세스에서 호출하는 함수가 TCP recv 함수를 의미함

[표] 5-1 데몬 프로세스 함수 종류

● function.name = string
- 범위 : 64자 이내
- 데몬 프로세스에서 호출하는 함수명을 등록한다. 동 함수는 반드시 공유 라이브러리에서 제공하는 함수이어야 한다.
- 함수명은「function.prototype」항목에서 등록하는 함수명과 같아야 한다.

● function.returntype = string
- 범위 : 16자 이내
- 함수의 리턴 타입을 등록한다.
- 리턴 타입으로는 short, int, long, void, ssize_t, unsigned 여섯 종류가 있다.

● function.prototype = string
- 범위 : 1024자 이내
- 함수의 형식을 등록한다.

선택항목

● function.prefunc = string
- 범위 : 1024자 이내
- 데몬 프로세스에서 호출하는 함수를 실행하기 전에 성능정보 추출을 위하여 호출할 함수의 형식을 등록한다. 동 함수는 함수 타입에 따라서 이미 정의되어 있기 때문에 사용자 임의대로 변경할 수 없다.
- 함수 종류별 선 호출 함수의 형식은 아래와 같다.

타입	함수의 형식
SVRSTART	server_start_prefix()
CUSTOM	사용자함수명_prefix(char *funcname, char *readbuf, int *buflen)
RECVMSGQUE	msgqueue_recv_prefix(char *funcname, int msqid, void *msgp, long msgsz)
SENDMSGQUE	msgqueue_send_prefix(char *funcname, int msqid, char *hbuf, long &hlen)
RECVMQ	mq_recv_prefix(char *funcname, mqd_t mqdes, char *msgp, long msg_len)
SENDMQ	mq_send_prefix(char *funcname, mqd_t mqdes, char *hbuf, long &hlen)
SENDTCP	tcp_send_prefix(char *funcname, int fd, const char *buf, long len, char *hbuf, long &hlen)
RECVTCP	tcp_recv_prefix(char *funcname, int fd, char *buf, long len)

[표] 5-2 함수 종류별 선 호출 함수의 형식

function.postunc = string

- 범위 : 1024자 이내
- 데몬 프로세스에서 호출하는 함수를 실행한 후에 성능정보 추출을 위하여 호출할 함수의 형식을 등록한다. 동 함수는 함수 타입에 따라서 이미 정의되어 있기 때문에 사용자 임의대로 변경할 수 없다.
- 함수 종류별 후 호출 함수의 형식은 아래와 같다.

타입	함수의 형식
SVRSTART	server_start_postfix()
CUSTOM	사용자함수명_postfix(char *funcname, char *readbuf, int *buflen, long hook_rc)
RECVMSGQUE	msgqueue_recv_postfix(char *funcname, int msqid, void *qmsg, long hook_rc, long &hlen)
SENDMSGQUE	msgqueue_send_postfix(char *funcname, int msqid, void *qmsg, long msgsz, long hook_rc)
RECVMQ	mq_recv_postfix(char *funcname, mqd_t mqdes, char *msgp, long hook_rc, long &hlen)
SENDMQ	mq_send_postfix(char *funcname, mqd_t mqdes, char *msgp, long msg_len, long hook_rc)
SENDTCP	tcp_send_postfix(char *funcname, int fd, const char *buf, long len, int flags, long hook_rc)
RECVTCP	tcp_recv_postfix(char *funcname, int fd, const char *buf, long len, int flags, long hook_rc)

[표] 5-3 함수 종류별 후 호출 함수의 형식

5.2. 데몬 프로세스 모듈 생성

데몬 프로세스의 성능 정보를 추출하기 위하여 데몬 프로세스에서 호출하는 함수들에 대해서 소스를 생성하는 방식으로 처리하기 때문에 위에서 설명한 데몬 프로세스 함수 환경설정을 한 다음에 Pharos TP 시스템에서 제공하는 명령어를 통하여 Hooking 모듈을 생성해야 한다.

참고

데몬 프로세스 함수 정보를 등록한 Config 파일은 반드시 $PHOMEDIR/config에 존재해야 한다.

5.2.1. 소스 생성 방법

Hooking 모듈은 명령어 프롬프트 상태에서 pdaecustomgen 명령어를 통하여 생성한다.

```
PharosTP@AIX6:/home/PharosTP/apm/bin> pdaecustomgen -i daecustomgen.cfg
```

-i 옵션은 데몬 프로세스 함수 정보를 등록한 Config 파일 명을 입력한다.

5.2.2. 생성된 소스

pdaecustomgen 명령어를 통하여 생성되는 소스는 $PHOMEDIR/work에 생성된다.

config 정보

```
# tp daemon custom function info
[function]
function.type=SVRSTART
function.name=init_apm
function.returntype=int
function.prototype=init_apm(void)
function.prefunc=server_start_prefix()
function.postfunc=server_start_postfix()

[function]
function.type=CUSTOM
function.name=RecvFromFep_
function.returntype=int
function.prototype=RecvFromFep_(char *arg1, char *arg2, char *readbuf, int *buflen)
function.prefunc=recvfromfep_prefix(char *funcname, char *readbuf, int *buflen)
function.postfunc=recvfromfep_postfix(char *funcname, char *readbuf, int *buflen, long hook_rc)
```

```
[function]
function.type=RECVMQ
function.name=mq_receive
function.returntype=ssize_t
function.prototype=mq_receive(mqd_t mqdes, char *msg_ptr, size_t msg_len, unsigned *msg_prio)
function.prefunc=mq_recv_prefix(char *funcname, mqd_t mqdes, char *msgp, long msg_len)
function.postfunc=mq_recv_postfix(char *funcname, mqd_t mqdes, char *msgp, long hook_rc, long &hlen)

[function]
function.type=CUSTOM
function.name=SendToBKFep
function.returntype=int
function.prototype=SendToBKFep(char *psfepid, char *psmsg, int imsglen, int *piseq)
function.prefunc=sendtobkfep_prefix(char *funcname, char *psmsg, int imsglen)
function.postfunc=sendtobkfep_postfix(char *funcname, char *psmsg, long imsglen, long hook_rc)

[function]
function.type=SENDMQ
function.name=mq_send
function.returntype=int
function.prototype=mq_send(mqd_t mqdes, const char *msg_ptr, size_t msg_len, unsigned msg_prio)
function.prefunc=mq_send_prefix(char *funcname, mqd_t mqdes, char *hbuf, long &hlen)
function.postfunc=mq_send_postfix(char *funcname, mqd_t mqdes, char *msgp, long msg_len, long hook_rc)

[function]
function.type=SENDTCP
function.name=send
function.returntype=ssize_t
function.prototype=send(int fd, const void *buf, size_t len, int flags)
function.prefunc=tcp_send_prefix(char *funcname, int fd, const char *buf, long len, char *hbuf, long &hlen)
function.postfunc=tcp_send_postfix(char *funcname, int fd, const char *buf, long len, int flags, long hook_rc)
```

[그림] 5-2 daecustomgen.cfg – 데몬 프로세스 함수 등록

Hooking 소스

```
/*
 * @file                daecustomhook.c
 * @brief               Daemon hooking source
 * @author
 * @library
 *
 * @history
 *
 *  성     명  :  일     자  :              변        경        내        용
 *  --------      --------      ------------------------------------------------------------------
 *
 */

/*
 *
 * @pseudo code
 *
 */

/* Linux requires _GNU_SOURCE macro for RTLD_NEXT */
#if defined(__linux) || defined(linux) || defined(Linux) || defined(_LINUX) || defined(_LINUX64)
#    ifndef _GNU_SOURCE
#        define _GNU_SOURCE
#    endif
#endif

/* ------------------------------------- include files ------------------------------------- */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <signal.h>
#include <fcntl.h>
#include <unistd.h>
#include <mqueue.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```
#include <sys/timeb.h>
#include <stdarg.h>
#include <math.h>
#include <strings.h>
#include <sys/param.h>
#include <sys/time.h>
#include <time.h>
#include <dlfcn.h>
#include <pthread.h>


/* ---------------------------------------- constant ---------------------------------------- */
#define SYS_FALSE                                   0
#define SYS_TRUE                                    1


/* ------------------------------------- macro definitions ---------------------------------- */
/* value return type function hook */
#define BEGIN_FUNCTION_HOOK_DECLARE(RETTYPE, NAME, ...)
typedef RETTYPE (*NAME##_hook_t)(__VA_ARGS__);
RETTYPE NAME(__VA_ARGS__)
{
    static NAME##_hook_t NAME##_hook = NULL;
    RETTYPE hook_rc;

    if (NAME##_hook == NULL ) {
        NAME##_hook = (NAME##_hook_t)dlsym(RTLD_NEXT, #NAME);
        if (NAME##_hook == NULL) {
            fprintf(stderr,"ERROR: unsatisfied symbol [%s]₩n", #NAME);
            exit(1);
        }
    }

#define END_FUNCTION_HOOK_DECLARE
    return hook_rc;
}
```

```
#define ORIGIN_FUNCTION_CALL_RETURN(NAME, ...)
	do {
		hook_rc = (*(NAME##_hook))(__VA_ARGS__);
	} while(0)

#define ORIGIN_FUNCTION_CALL_RETURN_NOARGS(NAME)
	do {
		hook_rc = (*(NAME##_hook))();
	} while(0)

/* void return type function hook */
#define BEGIN_FUNCTION_HOOK_DECLARE_VOID(NAME, ...)
typedef void (*NAME##_hook_t)(__VA_ARGS__);
void NAME(__VA_ARGS__)
{
	static NAME##_hook_t NAME##_hook = NULL;

	if (NAME##_hook == NULL ) {
		NAME##_hook = (NAME##_hook_t)dlsym(RTLD_NEXT, #NAME);
		if (NAME##_hook == NULL) {
			fprintf(stderr,"ERROR: unsatisfied symbol [%s]₩n", #NAME);
			exit(1);
		}
	}

#define END_FUNCTION_HOOK_DECLARE_VOID
	return;
}

#define ORIGIN_FUNCTION_CALL_VOID(NAME, ...)
	do {
		(*(NAME##_hook))(__VA_ARGS__);
	} while(0)

#define ORIGIN_FUNCTION_CALL_VOID_NOARGS(NAME)
	do {
		(*(NAME##_hook))();
	} while(0)

.....
```

```
/* ------------------------------ structure definitions ---------------------------------------- */
typedef struct {
#if defined(_IBM) || defined(_IBM64)
      mtyp_t msgtype;
#elif defined(_SUN) || defined(_SUN64)
      char     resvd[4];
      int      msgtype;
#else
      long     msgtype;
#endif
      char     msg[1];
} msgq_t;

/* ---------------------------- exported global variables declarations ------------------------- */
extern long     g_proc_unreg_flag;
extern long     g_queue_hook_flag;

msgrcv_hook_t msgrcv_hook = NULL;
msgsnd_hook_t msgsnd_hook = NULL;
send_hook_t     send_hook    = NULL;

/* --------------------------------- exported function declarations ---------------------------- */
extern long server_start_prefix(void);
extern long server_start_postfix(void);
extern long recvfromfep_prefix(char *funcname, char *readbuf, int *buflen);
extern long recvfromfep_postfix(char *funcname, char *readbuf, int *buflen, long hook_rc);
extern long recvfrombkfep_prefix(char *funcname, char *psmsg, int *pimsglen);
extern long recvfrombkfep_postfix(char *funcname, char *psmsg, int *pimsglen, long hook_rc);
extern long mq_recv_prefix(char *funcname, mqd_t mqdes, char *msgp, long msg_len);
extern long mq_recv_postfix(char *funcname, mqd_t mqdes, char *msgp, long hook_rc, long *hlen);
.....
```

```
/* ------------------------------------------------------------------------------------------
 * @function            int init_apm()
 * @brief               init_apm API
 * @parameter
 * @return              0    : 정상
 *                      -1   : 에러
 * ------------------------------------------------------------------------------------------
 */
BEGIN_FUNCTION_HOOK_DECLARE(int, init_apm, void)
{
    int   rc = 0;

    /* ------------------------------------------------------------------ */
    /* user prefix function call                                          */
    /* ------------------------------------------------------------------ */
    server_start_prefix();

    /* ------------------------------------------------------------------ */
    /* call the original                                                   */
    /* ------------------------------------------------------------------ */
    ORIGIN_FUNCTION_CALL_RETURN_NOARGS(init_apm);

    /* ------------------------------------------------------------------ */
    /* user postfix function call                                         */
    /* ------------------------------------------------------------------ */
    server_start_postfix();
}
END_FUNCTION_HOOK_DECLARE

.....................
```

[그림] 5-3 daecustomhook.c

5.2.3. 생성된 makefile

생성된 소스를 컴파일 할 수 있도록 Makefile도 해당 시스템에 맞게 자동으로 생성된다. 예를 들어 AIX 시스템인 경우 아래와 같이 생성된다.

```
#=================================================================#
# daecustomhook makefile
#=================================================================#
BASE       = daecustomhook
TARGET    = lib$(BASE)$(OS_SO_SUFFIX)

# ---------------------------------------------------------------------------------------------- #
# Objects
# ---------------------------------------------------------------------------------------------- #
OBJS       = daecustomhook.o

# ---------------------------------------------------------------------------------------------- #
# MY DEFINE
# ---------------------------------------------------------------------------------------------- #
MY_C_DEFINE = -DNO_DEBUG
MY_PROC_DEFINE    =
MY_IPATH       = -I. $(TP_IPATH)
MY_CFLAGS     =
MY_LDPATH     = -L. -L$(PHOMEDIR)/lib
MY_LDLIBS     = -lpharostpdaehm -lpharostpcom -lpharos $(OS_THRLIBS)
# ---------------------------------------------------------------------------------------------- #
# OS Environment
# ---------------------------------------------------------------------------------------------- #
OS_CFLAGS           = -qlanglvl=extc99 -q64 -qinfo=pro -brtl -O2 -qcpluscmt
OS_CDEFINE          = -D_IBM64 -D_REENTRANT
OS_IPATH            =
OS_LDFLAGS          = -qlanglvl=extc99 -brtl
OS_LDPATH           =
OS_OSLIBS           = -lc -lm
OS_LDLIBS           =
OS_THRLIBS          = -lpthread
OS_MONLIBS          =
OS_PMONLIBS         = -lperfstat
OS_SO_FLAGS         = -G -bshared -q64
OS_SO_SUFFIX        = .so
OS_COMPILER         = cc
```

```
# ------------------------------------------------------------------------------------------- #
# APM Environment
#   ------------------------------------------------------------------------------------------- #
APM_CDEFINE        = -DNO_DEBUG
APM_CFLAGS         =
APM_IPATH          = -I$(PHOMEDIR)/inc -I$(PHAROS_TPHOME)/inc
APM_LDPATH         = -L$(PHOMEDIR)/lib
APM_LDLIBS         = -lpharos -lpharostpcom

###############################Compile Options################################
# ------------------------------------------------------------------------------------------- #
# C Compiler Flag
# ------------------------------------------------------------------------------------------- #
CC                 = $(OS_COMPILER)
C_FLAGS            = $(MY_CFLAGS)     $(APM_CFLAGS)    $(OS_CFLAGS)
C_DEFINE           = $(MY_C_DEFINE) $(APM_CDEFINE) $(OS_CDEFINE)
I_FLAGS            = $(MY_IPATH)      $(APM_IPATH)     $(OS_IPATH)
CFLAGS             = $(C_FLAGS)       $(C_DEFINE)      $(I_FLAGS)

# ------------------------------------------------------------------------------------------- #
# LD Compiler Flag
# ------------------------------------------------------------------------------------------- #
LD         = $(OS_COMPILER)
LD_PATH        = $(MY_LDPATH) $(APM_LDPATH) $(TP_LDPATH) $(DATABASE_LDPATH) $(OS_LDPATH)
LD_LIBS        = $(MY_LDLIBS) $(OS_LDLIBS)
LD_FLAGS       = $(OS_LDFLAGS) $(LD_PATH)    $(LD_LIBS)

# ------------------------------------------------------------------------------------------- #
# Compile suffix rule
# ------------------------------------------------------------------------------------------- #
.SUFFIXES : .c .o

.c.o:
      $(CC) $(CFLAGS) -c $<

# ------------------------------------------------------------------------------------------- #
# Dependency
# ------------------------------------------------------------------------------------------- #
```

```
all: $(TARGET)

$(TARGET): $(OBJS)
	$(LD) $(OS_SO_FLAGS) -o $@ $(OBJS) $(LD_FLAGS)
	mv $(TARGET) $(PHOMEDIR)/lib

mv:
	mv $(TARGET) $(PHOMEDIR)/lib

clean:
	-rm -f core $(OBJS) $(TARGET)

#=================================================================#
# Make Dependency File
#=================================================================#
```

[그림] 5-4 daecustomhook.mk

5.2.4. 모듈 생성

생성된 소스를 컴파일하여 모듈(공유 라이브러리)를 생성해야 한다.

모듈을 생성하는 방법은 아래와 같다.

```
PharosTP@AIX6:/home/pharostp/apm/work> make -f daecustomhook.mk
        cc -G -bshared -q64 -o libdaecustomhook.so daecustomhook.o -qlanglvl=extc99 -brtl -L. -
L/home/pharostp/apm/lib      -lpharostpdaehm -lpharostpcom -lpharos -lpthread
        mv libdaecustomhook.so /home/pharostp/apm/lib
Target "all" is up to date.
PharosTP@AIX6:/home/pharostp/apm/work> cd ../lib
PharosTP@AIX6:/home/pharostp/apm/lib> ls -al
total 1840
drwxr-x---    2 PharosTP dba          4096 Aug  2 18:05 .
drwxr-xr-x   10 PharosTP dba           256 Aug  2 16:39 ..
-rwxr-xr-x    1 PharosTP dba          5773 Aug  2 18:05 libdaecustomhook.so
-rwxr-xr-x    1 PharosTP dba        171197 Jun 22 16:00 libpharos.so
-rwxr-xr-x    1 PharosTP dba        180896 Jun 22 16:00 libpharosdb.so
-rwxr-xr-x    1 PharosTP dba        103951 Jun 22 16:00 libpharostpcom.so
-rwxr-xr-x    1 PharosTP dba        125506 Jun 22 16:00 libpharostphm.dbg.so
-rwxr-xr-x    1 PharosTP dba        113189 Jun 22 16:00 libpharostphm.so
-rwxr-xr-x    1 PharosTP dba         31790 Jun 22 16:00 libpharostpsh.dbg.so
```

```
-rwxr-xr-x    1 PharosTP dba        29800 Jun 22 16:00 libpharostpsh.so
-rwxr-xr-x    1 PharosTP dba       112525 Jun 22 16:00 libpsysrmon.dbg.so
-rwxr-xr-x    1 PharosTP dba        37025 Jun 22 16:00 libpsysrmon.so
-rwxr-xr-x    1 PharosTP dba         1575 Jun 22 16:00 libsigusrproc.so
```

5.3. 데몬 프로세스 모듈 적용

생성된 Hooking모듈을 적용하고 데몬 프로세스를 재 시작해야 적용된다.

<.profile> / <.bash_profile>

```
# export preload
if [ $OS_VENDER = ibm64 ]; then
  export LDR_PRELOAD64=$PHOMEDIR/lib/libdaecustomhook.so
elif [ $OS_VENDER = ibm32 ]; then
      export LDR_PRELOAD=$PHOMEDIR/lib/libdaecustomhook.so
else
      export LD_PRELOAD=$PHOMEDIR/lib/libdaecustomhook.so
fi
```

<.cshrc>

```
# export preload
if [ $OS_VENDER = ibm64 ]; then
      setenv LDR_PRELOAD64 $PHOMEDIR/lib/libdaecustomhook.so
elif [ $OS_VENDER = ibm32 ]; then
      setenv LDR_PRELOAD $PHOMEDIR/lib/libdaecustomhook.so
else
      setenv LD_PRELOAD $PHOMEDIR/lib/libdaecustomhook.so
fi
```

[그림] 5-5 데몬 프로세스 모듈 적용

06

Pharos TP 실행 및 종료

6. Pharos TP 실행 및 종료

6.1. Pharos TP 실행

거래 추적을 위한 Pharos TP는 다음과 같은 순으로 실행한다.

6.1.1. 시스템 부팅 후 최초 실행

시스템을 부팅하고 최초로 Pharos TP 엔진을 실행하는 경우에는 반드시 TP Monitoring 보다 먼저 Pharos TP 엔진(pharostp)을 실행해야 한다. 왜냐하면 Pharos TP 엔진에서 TP Monitor 서버 프로세스의 거래추적 정보를 저장하기 위한 저장 공간을 확보하기 때문이다. 또한, 먼저 Pharos TP 엔진(pharostp)보다 Pharos 매니저(pmm)를 먼저 실행해야 한다. 그렇지 않으면 Pharos TP 엔진 프로세스가 실행되지 않는다.

1. Pharos 매니저(pmm) 실행
2. Pharos TP 엔진(pharostp) 실행
3. TP Monitoring 실행

참고

TP Monitor를 먼저 실행하면 일부 서버 프로세스는 거래추적을 하지 못한다.

6.1.1.1. 환경변수 확인

TP Monitor를 실행하는 사용자 계정의 환경변수와 Pharos TP 엔진을 실행하는 사용자 계정의 환경변수가 정확히 등록되어 있는지 확인한다.

6.1.1.2. Pharos TP 엔진 실행

Pharos TP 엔진 계정으로 로그인하여 Pharos TP 엔진을 실행한다. PHAROS TP엔진은 ptpboot 명령어로 실행한다.

```
PharosTP@AIX6:/home/pharostp> cd $PHOMEDIR/bin
PharosTP@AIX6:/home/pharostp/apm/bin> ptpboot
```

실행 프로세스 확인

```
PharosTP@AIX6:/home/pharostp/apm/bin> ps -ef|grep PharosTP
    PID      TTY      TIME  CMD
 8585390   pts/4   0:00  pmm –m etc
13959244   pts/4   0:00  -ksh
14811272   pts/4   0:00  psysrmon –r 150.100.202.51 –p 44001 –I 5 –n dmciaps01_I
18677894   pts/4   0:00  pharostp
19660814   pts/4   0:00  ps
20054206   pts/4   0:00  paladmin
```

공유 메모리 확인 ➔ ipcs –ma|grep PharosTP계정명

```
PharosTP@AIX6:/home/pharostp/apm/bin> ipcs -ma|grep PharosTP
m  38797336 0x000133ff --rw-rw-rw-  PharosTP     dba  pharostp     dba      2  20657784
9306166 21364938 21:16:57 21:19:40 21:16:51
```

참고

PHAROS 엔진에서 사용하는 공유메모리를 찾는 방법은 PHAROS 환경 파일의 **[mga_info]**절의 **mga.shmkey=78437**의 값을 hex로 변환하여 ipcs 결과와 확인한다.

메시지 큐 확인 ➔ ipcs –qa|grep PharosTP계정명

```
PharosTP@AIX6:/home/pharostp/apm/bin> ipcs -qa|grep PharosTP
q 540016701 0x00009c2c -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q  96469054 0x00009c2b -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q 344981568 0x00009c2a -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q   6291521 0x00009c29 -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q   1048642 0x00009c28 -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q   1048643 0x00009c27 -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q   1048644 0x00009c26 -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q   1048645 0x00009c25 -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q   1048646 0x00009c24 -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q   1048647 0x00009c23 -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q   1048648 0x00009c22 -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
q   1048649 0x00009c21 -Rrw-rw-rw-  PharosTP        dba PharosTP        dba            0            0
4194304        0        0 no-entry no-entry 21:34:51
```

참고

Pharos TP 엔진에서 사용하는 공유 메모리를 찾는 방법은 Pharos TP 환경 파일의 **[mga_info]**절의 **mga.shmkey=78437**의 값을 hex로 변환하여 ipcs 결과와 확인한다.

6.1.2. Pharos TP 엔진 재 실행

Pharos TP 엔진만을 종료하고 재실행하고자 하는 경우에는 간단하게 Pharos TP 엔진만 재실행하면 된다.

6.1.2.1. 환경변수 확인

Pharos TP 엔진을 실행하는 사용자 계정의 환경변수가 정확히 등록되어 있는지 확인한다.

6.1.2.2. Pharos TP 엔진 실행

Pharos TP 엔진 계정으로 로그인하여 Pharos TP 엔진을 실행한다. Pharos TP엔진은 ptpboot 명령어로 실행한다.

```
pharosTP@AIX6:/home/pharostp/apm> cd $PHOMEDIR/bin
pharosTP@AIX6:/home/pharostp/apm/bin> ptpdown
*WARNING* [pharostp] may be killed: pid=[25493572]
*WARNING* [ptpadmin] may be killed: pid=[22741124]
*WARNING* [psysrmon] may be killed: pid=[19071220]
*WARNING* [pmm] may be killed: pid=[3539158]
pharosTP@AIX6:/home/PharosTP/apm/bin> ptpboot
 Pharos Manager process starting ...
 Pharos Adapter process starting ...
 Pharos TP Admin process starting ...
 Pharos System Monitoring process starting ...
pharosTP@AIX6:/home/pharostp/apm/bin> ps -ef|grep PharosTP
pharosTP 19070984 24379462    2 21:42:15 pts/13   0:00 ps -ef
pharosTP 19333188 24379462    0 21:42:15 pts/13   0:00 grep pharosTP
pharosTP 21758120          1    0 21:41:52 pts/13   0:00 pmm
pharosTP 22347836          1    0 21:42:01 pts/13   0:00 psysrmon -r 175.118.115.24 -p 54002 -i 5 -n
aix6_TP
pharosTP 22741142          1    0 21:41:58 pts/13   0:00 ptpadmin
pharosTP 24379462 22872252    0 18:04:28 pts/13   0:00 -ksh
pharosTP 25493598          1    0 21:41:55 pts/13   0:00 pharosp
```

참고

psysrmon 프로세스 아규먼트

- -r : 수집서버 IP Address
- -p: 수집서버 연결 포트번호
- -i : 정보 전송 간격
- -n : 에이전트 명

6.2. Pharos TP 종료

Pharos TP엔진을 종료하고자 하는 경우에는 반드시 정상적으로 종료해야 한다. 특히 강제종료(kill -9)로 종료하지 말아야 한다.

```
PharosTP@AIX6:/home/pharostp/apm> cd $PHOMEDIR/bin
PharosTP@AIX6:/home/pharostp/apm/bin> ptpdown
*WARNING* [Pharostp] may be killed: pid=[25493572]
*WARNING* [ptpadmin] may be killed: pid=[22741124]
*WARNING* [psysrmon] may be killed: pid=[19071220]
*WARNING* [pmm] may be killed: pid=[3539158]
```

6.3. 장애조치 방법

6.3.1. 거래추적 공유 library 파일이 삭제된 경우

거래추적 공유 library가 삭제된 상태(lib 디렉터리 내용 중 일부 파일 삭제)에서 TP Monitor 계정으로 로그인하면 아래와 같은 오류가 발생하여 명령어를 입력할 수 없다. 이와 같은 오류가 발생하는 것은 TP Monitor 계정으로 로그인하면 자동으로 환경변수에 정의된 공유 library를 프리로딩 하는데 해당 파일이 없어서 로딩할 수 없기 때문에 발생하는 에러이다. 이런 경우에는 환경변수 중 "LDR_PRELOAD (기타: LD_PRELOAD)" 환경변수를 삭제하면 된다. 삭제 방법은 아래와 같다.

- 에러 내용

```
PharosTP@AIX6:/home/pharostp> ls
Exec() : 0509-036 다음 오류 때문에 ls 프로그램을 로드할 수 없습니다.
         0509-150 종속 모듈 /hom/pharos/lib/libpharostp2.so이(가) 로드되지 않았습니다.
        0509-150   /home/pharos/lib/libpharostp2.so 모듈을 로드할 수 없습니다..
        0509-150 시스템 오류 : 경로 이름에 있는 파일이나 디렉터리가 존재하지 않습니다.
PharosTP@AIX6:/home/pharostp>
```

- 조치 방법: export LDR_PRELOAD= [Enter] ← 환경변수 Clear

```
PharosTP@AIX6:/home/pharostp> export LDR_PRELOAD=
```

위와 같이 임시 조치 후에 해당 파일을 복원하면 된다.

6.3.2. 환경변수가 삭제된 경우

Pharos TP 환경변수는 삭제되지 않고 TP Monitor나 오라클 환경변수가 삭제된 경우에도 "6.3.1. 거래추적 공유 Library 파일이 삭제된 경우"와 동일한 에러가 발생한다. 이 경우에는 위의 조치 방법에 따라서 처리한다.

6.3.3. 거래추적을 종료하고자 하는 경우

TP Monitor 서버 프로세스에 대한 거래추적을 종료하고자 하는 경우에 TP Monitor 계정의 환경변수를 코

멘트 처리하고 TP Monitor를 재 기동하면 거래추적을 종료할 수 있다. 그러나 이 경우에 거래추적을 다시 하려면 환경변수를 재 설정하고 TP Monitor 재 기동해야 한다. 이와 같은 번거로움을 해소하기 위하여 거래 추적을 종료하는 방법은 아래의 방법 중 하나를 선택하여 처리하면 된다

6.3.3.1. Pharos TP admin이용

Pharos admin을 이용하여 거래추적을 종료할 수 있다. Pharos admin옵션 중 "**-t**"옵션을 이용하여 거래추적을 종료할 수 있다.

```
PharosTP@AIX6:/home/pharostp/apm/bin> pharosadm -t 0
PharosTP@AIX6:/home/pharostp/apm/bin> pharosadm -c

------------------------- version ------------------------------
PHAROS TP VERSION      = [3.5.6.0]
------------------------- instaqnce info -----------------------
resource_name          = [aix6_TP]
resource_id            = [100]
------------------------- tp monitor info ----------------------
tp_name                = [TMAX]
tp_version             = [5.0]
tp_username            = [Tmax]
db_name                = [oracle]
db_coninfo             = [Pharosdb/Pharosdb@FAUST11G]
engine_proc:num        = [4]
engine_proc:proc       = [tmm                ]
engine_proc:proc       = [clh                ]
engine_proc:proc       = [cll                ]
engine_proc:proc       = [tms_ora            ]
```

6.3.3.2. Pharos TP 엔진 종료

Pharos admin을 이용하는 방법 외에 Pharos TP 엔진을 종료하면 자동으로 거래추적을 종료한다.

```
PharosTP@AIX6:/home/pharostp/apm/bin> ptpdown
*WARNING* [Pharostp] may be killed: pid=[22347798]
*WARNING* [ptpadmin] may be killed: pid=[21037278]
*WARNING* [psysrmon] may be killed: pid=[19726502]
*WARNING* [pmm] may be killed: pid=[13369542]
PharosTP@AIX6:/home/PharosTP/apm/bin> pharosadm -c
------------------------- version ------------------------------
PHAROS TP VERSION      = [3.5.6.0]
------------------------- instaqnce info -----------------------
resource_name          = [tux01_Yoona]
resource_id            = [1]
------------------------- tp monitor info ----------------------
tp_name                = [TUXEDO]
tp_version             = [11.0]
tp_username            = [tuxapp]
db_name                = [oracle]
db_coninfo             = [pharosdb/pharosdb]
------------------------- server info --------------------------
address                = [1.235.117.149]
send_port              = [44001]
send_session_num       = [1]
recv_port              = [44001]
recv_session_num       = [1]
----------------------- monitoring info ------------------------
mon.type               = [tp]
mon.msgkey             = [39969]
mon.svc_cp             = [0]
mon.trace_level        = [0]
mon.act_svc_int        = [5]
mon.act_svc_cp         = [3]
mon.sys_info_int       = [5]
```

참고

Pharos 엔진을 종료할 때 강제종료(kill -9)하지 말고 정상적인 ptpdown 명령을 이용하여 종료해야 한다.

07

Pharos TP 관리도구

■ 7.1 pharosadm

7. Pharos TP 관리도구

PHAROS TP 시스템이 동작하고 있다면, 현재 환경설정에 대한 정보를 확인해 보고 이를 동적으로 변경한다든지 등의 시스템에 대한 관리가 필요하다. 이를 위해 PHAROS 시스템은 pharonadm을 제공하여 command interpreter 형태의 명령어를 이용하여 동적 시스템 관리를 할 수 있다.

7.1. pharosadm

pharosadm 프로그램은 UNIX 환경의 셸과 비슷한 command interpreter 형태로 명령어를 입력하면 이를 해석하여 실행한다.

```
usage: pharosadm [-h] [-c] [-d] [-L] [-V {서버명}] [-z {서비스명}] [-a] [-t {0|1|2|3|4|}] [-p seconds]
                 [-s {0|1}] [-q] [-b seconds] [-e interval] [-w count] [-y seconds] [-f interval]
                 [-i {0|1}] [-j seconds] [-k {0|1}] [-l {0|1}] [-m seconds] [-n {0|1}] [-o seconds]
                 [-O seconds] [-r count] [-u {0|1}] [-g {0|1}] [-v seconds] [-Q count]
                 [-B [0|1}] [-P {0|1}] [-A service name] [-D service name] [-E {0|1|2}]
                 [-F txcode reload file name] [-M seconds]
        -h: Help
        -c: Displays config information
        -d: Dump shared memory information
        -L: Lists all servers in the register
        -V: Displays information to the register server
        -z: Displays information to the register service
        -a: Flags to displays active service
        -t: Levels to transaction trace (mon.trace_level)
        -p: Critical point for transaction trace (mon.svc_cp)
        -s: Flags to saving sql text hashcode (sql.sql_hash_flag)
        -q: Clears saved sql text hashcode
        -b: Critical point for active transaction trace (mon.act_svc_cp)
        -e: Interval for send active transaction trace (mon.act_svc_int)
        -w: Sql default statement number for call tree (sql.sql_stack_cnt)
        -y: Sql critical point for transaction trace (sql.sql_time_limit)
        -f: Interval for send system cpu & memory stats information (mon.sys_info_int)
        -i: Sum flags for transaction trace (mon.sum_flag)
        -j: Interval for send sum information (mon.sum_send_int)
        -k: Flags for service system usage information (mon.svc_sys_usage)
        -l: Create flags to process's cpu usage and memory usage information (mon.proc_info_flag)
```

-m: Interval for send process cpu & memory information (mon.proc_info_int)
-n: Create flags to monitor administrator information (mon.adm_info_flag)
-o: Interval for send to monitor administrator config information (mon.adm_cfg_info_int)
-O: Interval for send to monitor administrator real time information (mon.adm_rt_info_int)
-r: Buffering count for merge (mon.merge_cnt)
-u: Flags for error data buffering (mon.err_buf_flag)
-g: Flags for service end logging (log.log_flag)
-v: Interval for tp monitor sys log tail (log.sys_log_interval)
-Q: Call tree number for service trace (mga.call_seq)
-B: Flags for bind data collection (sql.bind_coll_flag)
-P: Flags for sql text plan (sql.sql_plan_flag)
-A: Flags for add skip service (mon.skip_svc)
-D: Flags for delete skip service (mon.skip_svc)
-E: Txcode used flag (mon.txcode_flag)
-F: Txcode reload file name (mon.txcode_path)
-M: Critical point for module trace (mod.mod_time_limit)

다음은 pharosadm의 옵션에 대한 설명이다.

옵션	값	설명
-h		Help Pharosadm 프로그램에서 처리하는 옵션에 대한 상세 설명을 출력한다.
-c		Pharos TP 시스템에 현재 적용된 Config 정보를 출력한다.
-d		서버 등록정보가 저장된 공유메모리의 어드레스 정보를 출력한다. 동 옵션은 디버그 용으로 사용하기 때문에 일반 사용자에게는 의미가 없다.
-L		PHAROS TP 시스템에 등록된 서버 프로세스 List 를 출력한다.
-V		PHAROS TP 시스템에 등록된 특정 서버의 상세 정보를 출력한다.
-z		PHAROS TP 시스템에 등록된 특정 서비스의 상세 정보를 출력한다.
-a		현재 실행되고 있는 모든 서비스의 Active Call Tree 정보를 출력한다.
-t	0 : no trace 1 : service level 2 : module level 3 : sql level 4 : atmi level	거래추적 대상의 레벨을 동적으로 변경한다. 거래추적 대상이 서비스 단위인지 아니면 SQL 단위까지 거래추적을 할 것인지 등 거래추적 대상의 상세 레벨을 변경할 수 있다. 거래추적 레벨에 대한 설명은 "3. 환경파일 설정"을 참조한다.
-p	seconds	서비스의 임계치를 초단위로 변경한다. 지정된 임계치 이하의 데이터는 Call Tree에는 나타나지 않지만 통계에는 나타난다.
-s	0 : no saving 1 : saving	SQL 해쉬코드 저장할 것인지 여부를 변경한다.
-q		지금까지 저장되어있는 SQL 해쉬코드를 모두 clear하고 다시 SQL 해쉬코

		드를 저장할 수 있도록 변경한다.
-b	seconds	Active 서비스에 대한 임계치를 변경한다.
-e	seconds	Active Call Tree 전송 간격을 변경한다.
-w		SQL 임계치와 관계없이 기본적으로 Call Tree에 나타나야 할 SQL Call Tree 수를 변경한다.
-y	Milliseconds	SQL Call Tree가 너무 많은 경우 임계치 이상의 Call Tree만 나타내기 위하여 SQL에 대한 임계치를 변경한다.
-f	seconds	시스템의 CPU나 사용량이나 MEMORY 사용량 정보를 전송할 간격을 변경한다.
-i	0 : no sum 1 : sum	에이전트 집계처리 여부를 변경한다.
-j	seconds	집계 데이터 출력 간격을 변경한다.
-k	0 : no check 1 : check	서비스 수행 시 소요된 CPU 사용량이나 Memory Leek을 조사할 것인지 여부를 변경한다.
-l	0 : no create 1 : create	서버 프로세스별 CPU 사용량이나 Memory 사용량에 대한 정보를 추출할 것인지 여부를 변경한다.
-m	seconds	"-l" 옵션으로 추출한 정보를 수집서버로 전송할 간격을 변경한다.
-n	0 : no create 1 : create	TP Monitor Admin 정보를 추출할 것인지 여부를 변경한다.
-o	seconds	TP Monitor Admin 정보 중 Config성 정보를 추출할 간격을 변경한다.
-O	seconds	TP Monitor Admin 정보 중 Real Time성 정보를 추출할 간격을 변경한다.
-r	0 : no buffering 1 : buffering	임계치 이하의 데이터에 대한 통계 정보를 매번 수집서버에 전송하면 부하가 발생할 수 있으므로 이를 묶어서 전송 할 수 있다. 이때 몇 개씩 묶어서 전송 할 것인지 건수를 변경한다.
-u	0 : no buffering 1 : buffering	서비스 처리 임계치 이하의 데이터 중 에러가 발생한 거래에 대해서 Call Tree를 전송할 것인지 여부를 변경한다.
-g	0 : no logging 1 : logging	서비스 종료 시점에 서비스가 종료되었다는 정보를 로그로 출력할 것인지 여부를 변경한다.
-v	seconds	TP Monitor 시스템 로그에 대한 Tail 간격을 변경한다.
-Q	number	거래 추적에 대한 Call Tree 건수를 변경한다.
-B	0 : no collection 1 : collection	SQL 텍스트에서 사용한 바인드 데이터를 추출할 것인지 여부를 변경한다.
-P	0 : no plan 1 : plan	SQL 텍스트에 대한 Plan 정보를 볼 것인지 여부를 변경한다. Plan 정보를 보기 위해서는 데이터베이스 연결 정보와 권한을 부여해야 한다.
-A	Service name	거래추적에서 제외할 서비스를 추가하기 위한 서비스 명을 입력한다.
-D	Service name	거래추적에서 제외할 서비스를 삭제하기 위한 서비스 명을 입력한다.
-C	File name	거래추적에서 제외할 거래코드를 재 로딩하기 위한 파일명을 입력한다. Version 3.5.2 이상

-M	seconds	모듈에 대한 임계치를 초단위로 변경한다. 지정된 임계치 이하의 데이터는 Call Tree에는 나타나지 않는다. 단, 모듈 내에 하위 Call Tree가 있으면 임계치와 관계없이 나타난다 Version 3.5.2 이상

[표] 7-1 pharosadm 명령어

7.1.1. 환경 파일 정보 출력 (-c)

Pharos TP 시스템에 적용된 환경 파일의 정보를 출력한다.

- 실행방법 : pharosadm -c

```
------------------------- version ------------------------------
PHAROS TP VERSION      = [3.5.6.0]
------------------------- instaqnce info -----------------------
resource_name           = [AIX5_tmax5]
resource_id             = [100]
------------------------- tp monitor info ----------------------
tp_name                  = [TMAX]
tp_version              = [5.0]
tp_username              = [tmax]
db_name                  = [oracle]
db_coninfo              = [pharosdb/pharosdb@FAUST11G]
engine_proc:num         = [4]
engine_proc:proc        = [tmm                 ]
engine_proc:proc        = [clh                 ]
engine_proc:proc        = [cll                 ]
engine_proc:proc        = [tms_ora             ]
------------------------- server info --------------------------
address                 = [1.235.117.149]
send_port               = [44001]
send_session_num        = [1]
recv_port               = [44001]
recv_session_num        = [1]
----------------------- monitoring info ------------------------
mon.type                = [tp]
mon.msgkey               = [39969]
mon.svc_cp              = [0]
```

```
mon.trace_level          = [0]
mon.act_svc_int          = [5]
mon.act_svc_cp           = [3]
mon.sys_info_int         = [5]
mon.sum_flag             = [1]
mon.sum_send_int         = [60]
mon.svc_sys_usage        = [1]
mon.proc_info_flag       = [1]
mon.proc_info_int        = [5]
mon.adm_info_flag        = [1]
mon.adm_cfg_info_int = [600]
mon.adm_rt_info_int    = [5]
mon.merge_cnt            = [10]
mon.err_buf_flag         = [1]
mon.svr_skip_num         = [4]
mon.skip_svr             = [tmm,clh,cll,tms_ora,]
mon.svc_skip_num         = [0]
mon.txcode_flag          = [0]
mon.txcode_path          = []
mon.apm_header_flag    = [0]
mon.async_to_tx          = [0]
mon.apm_skip_svc_num = [0]
----------------------module monitoring info ----------------------
mod.mod_time_limit     = [0]
----------------------sql monitoring info ------------------------
sql.sid_coll_flag        = [1]
sql.bind_coll_flag       = [1]
sql.sql_plan_flag        = [1]
sql.sql_hash_flag        = [1]
sql.sql_stack_cnt        = [100]
sql.sql_time_limit       = [0]
----------------------- position info ---------------------------
position.appl_code_pos    = [0]
position.appl_code_len    = [0]
position.txcode_pos       = [0]
position.txcode_len       = [0]
position.gid_pos          = [0]
position.gid_len          = [0]
position.gid_seq_pos      = [0]
```

```
position.gid_seq_len       = [0]
position.apm_header_pos = [0]
position.user_msg_pos     = [0]
position.user_msg_len     = [0]
------------ application error info ----------------------------
errinfo.errmsg_pos     = [0]
errinfo.errmsg_len     = [0]
errinfo.errmsg_code    = []
errinfo.code_pos        = [0]
errinfo.code_len        = [0]
errinfo.msg_pos          = [0]
errinfo.msg_len          = [0]
---------------------- trace logging info ---------------------
log.log_flag              = [0]
log.log_path               = []
log.sys_log_path        = []
log.sys_log_pattern    = [CLH]
log.sys_log_interval = [5]
log.core_file_flag     = [0]
log.core_file_path     = [/home/pharostp/apm/log/corelog]
----------------------- mga info -------------------------------
mga.shmkey                 = [78847]
mga.svr_process          = [1100]
mga.call_seq               = [512]
mga.sql_hash_num        = [100000]
---------------------- link service info ----------------------
linksvc.svc_num          = [0]
----------------------- Adapter Active Time --------------------
Active Interval          = [3]
Active Time                = [16:00:27]
```

08

부록

8. 부록

A. Pharos TP 설치 CHECK LIST

A.1. Check List

번호	체크사항	비고
1	Pharos 관련 환경변수가 Middleware실행 계정 profile에 등록되었는가?	
2	Pharos 관련 환경변수가 Pharos 실행 계정 profile에 등록되었는가?	
3	Pharos TP 엔진 파일의 권한이 실행가능 권한으로 변경되었는가?	
4	환경파일이 시스템 환경에 맞게 변경되었는가?	
5	환경파일에 정의된 포트번호가 기 사용 중인가?	
6	Pharos TP 엔진 실행 후 공유메모리가 정상적으로 생성되었고, 권한이 0666으로 설정되었는가?	
7	Pharos TP 엔진 실행 후 메시지 큐가 정상적으로 생성되었고, 권한이 0666으로 설정되었는가?	
8	Pharos 매니저를 통하여 공유메모리 정보가 조회되는가?	
9	netstat를 통하여 수집서버와 연결이 되었는가?	
10	Pharos TP 엔진 로그 파일에 에러 내용 확인?	

[표] 8-1 Check List

1. Pharos 관련 환경변수가 Middleware 실행 계정 profile에 등록되었는가?

● TUXEDO 계정 profile

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
export EXINIT='set ts=4'
umask 022

. /home/pharostp/apm/bin/pharostp.env
~
~
~
```

● env command 실행 결과

```
[tuxedo@Yoona ~]$ env | grep pharos
PHOMEDIR=/home/pharostp/apm
LD_PRELOAD=/home/pharostp/apm/lib/libpharostpsh.so
OLDPWD=/home/pharostp/apm/bin
PHAROS_CONFIG=pharostp.cfg
PHAROS_LOGDIR=/home/pharostp/apm/log
[tuxedo@Yoona ~]$
```

2. Pharos 관련 함수가 Pharos 실행 계정 .profile에 등록되었는가?

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH

# ========================================== #
#  PharosTP Environment Set                  #
# ========================================== #
. /home/pharostp/apm/bin/pharosadt.env
```

3. Pharos TP 엔진 파일의 권한이 실행가능 권한으로 변경되었는가?

```
[pharostp@Yoona:/home/pharostp/apm/bin] ls -al
total 1128
drwxrwxr-x  2 pharostp dba    4096 Aug 25 16:33 ./
drwxrwxr-x 11 pharostp dba    4096 Jun 22 15:49 ../
-rwxrwxr-x  1 pharostp dba    3178 Jun  1 11:30 analysiscore.sh*
-rwxrwxr-x  1 pharostp dba    1969 Jun  1 11:30 datedif.sh*
-rwxrwxr-x  1 pharostp dba   15383 Jun 22 15:48 pcorelogsend*
-rwxrwxr-x  1 pharostp dba   42949 Jun 22 15:48 pcustomgen*
-rwxrwxr-x  1 pharostp dba   48427 Jun 22 15:48 pharosadm*
-rwxrwxr-x  1 pharostp dba     570 Jun  1 13:12 pharosadt.env*
-rwxrwxr-x  1 pharostp dba   48143 Jun 22 15:48 pharosmon*
-rwxrwxr-x  1 pharostp dba  325513 Jun 22 15:48 pharostp*
-rwxrwxr-x  1 pharostp dba  345611 Jun 22 15:48 pharostp.dbg*
-rwxrwxr-x  1 pharostp dba    1002 Jun  1 13:12 pharostp.env*
-rwxrwxr-x  1 pharostp dba    2766 Jun  1 11:30 plogrm.sh*
-rwxrwxr-x  1 pharostp dba   30508 Jun 22 15:48 pmm*
-rwxrwxr-x  1 pharostp dba    7254 Jun 22 15:48 psysrmon*
-rwxrwxr-x  1 pharostp dba    7254 Jun 22 15:48 psysrmon.dbg*
-rwxrwxr-x  1 pharostp dba   66327 Jun 22 15:48 ptpadmin*
-rwxrwxr-x  1 pharostp dba   72436 Jun 22 15:48 ptpadmin.dbg*
-rwxrwxr-x  1 pharostp dba   62439 Jun  8 18:13 ptpadmin.nodbg*
-rwxrwxr-x  1 pharostp dba    1035 Jun  1 11:30 ptpboot*
-rwxrwxr-x  1 pharostp dba    1472 Jun  1 11:30 ptpdown*
-rwxrwxr-x  1 pharostp dba     198 Jun  4 11:24 tmax.env*
[pharostp@Yoona:/home/pharostp/apm/bin]
```

4. 환경파일이 시스템 환경에 맞게 변경되었는가?

```
##################################################################################
# PHAROS TP config information file
##################################################################################

##################################################################################
# Instance information
##################################################################################
[instance_info]

# adapter name
instance.res_name=tux01_Yoona

# adapter unique number
instance.res_id=1


##################################################################################
# TP Monitor information
##################################################################################
[tpmon_info]

# 거래추적 대상 tp monitor name
tpmon.tp_name=TUXEDO

# 거래추적 대상 tp monitor version
tpmon.tp_ver=11.0

# 거래추적 대상 tp monitor 실행 계정 명
tpmon.tp_username=tuxapp

# 거래추적 대상 DB 종류
tpmon.db_name=oracle

# 거래추적 대상 DB 접속정보
tpmon.db_coninfo=pharosdb/pharosdb

# 거래추적 대상 DB 종류
tpmon.engine_proc=BBL,LMS,WSL,WSH,DMADM,GWADM,GWTDOMAIN
```

5. 환경파일에 정의된 포트번호가 사용 중인가?

- netstat –a|grep 44001
- 44001 포트번호로 ESTALISHED 된 상태가 있으면 해당 포트가 사용중임

6. Pharos TP 엔진 실행 후 공유메모리가 정상적으로 생성되었고, 권한이 0666으로 설정되었는가?

```
[pharostp@Yoona:/home/pharostp/apm/config] ipcs -m | grep pharos
0x000133ff 7405575    pharostp  666        20628720   3
0x00013400 7438344    pharostp  666        226023624  1
[pharostp@Yoona:/home/pharostp/apm/config]
[pharostp@Yoona:/home/pharostp/apm/config]
```

7. Pharos TP 엔진 실행 후 메시지 큐가 정상적으로 생성되었고, 권한이 0666으로 설정되었는가?

```
[pharostp@Yoona:/home/pharostp/apm/config] ipcs -q | grep pharos
0x00009c21 0          pharostp   666        0            0
0x00009c22 32769      pharostp   666        0            0
0x00009c23 65538      pharostp   666        0            0
0x00009c24 98307      pharostp   666        0            0
0x00009c25 131076     pharostp   666        0            0
0x00009c26 163845     pharostp   666        0            0
0x00009c27 196614     pharostp   666        0            0
0x00009c28 229383     pharostp   666        0            0
0x00009c29 262152     pharostp   666        0            0
0x00009c2a 294921     pharostp   666        0            0
0x00009c2b 327690     pharostp   666        0            0
0x00009c2c 360459     pharostp   666        0            0
[pharostp@Yoona:/home/pharostp/apm/config]
```

8. Pharos 매니저를 통하여 공유메모리 정보가 조회되는가?

● pharosadm –c

```
[pharostp@Yoona:/home/pharostp/apm/config] pharosadm -c

------------------------- version -------------------------------
PHAROS TP VERSION    = [3.5.6.0]
------------------------- instaqnce info ------------------------
resource_name        = [tux01_Yoona]
resource_id          = [1]
------------------------- tp monitor info -----------------------
tp_name              = [TUXEDO]
tp_version           = [11.0]
tp_username          = [tuxapp]
db_name              = [oracle]
db_coninfo           = [pharosdb/pharosdb]
engine_proc:num      = [7]
engine_proc:proc     = [BBL             ]
engine_proc:proc     = [LMS             ]
engine_proc:proc     = [WSL             ]
engine_proc:proc     = [WSH             ]
engine_proc:proc     = [DMADM           ]
engine_proc:proc     = [GWADM           ]
engine_proc:proc     = [GWTDOMAIN       ]
------------------------- server info ---------------------------
address              = [1.235.117.149]
send_port            = [44001]
send_session_num     = [1]
recv_port            = [44001]
recv_session_num     = [1]
------------------------ monitoring info ------------------------
mon.type             = [tp]
mon.msgkey           = [39969]
mon.svc_cp           = [0]
mon.trace_level      = [5]
mon.act_svc_int      = [5]
mon.act_svc_cp       = [3]
mon.sys_info_int     = [5]
mon.sum_flag         = [1]
mon.sum_send_int     = [60]
mon.svc_sys_usage    = [1]
mon.proc_info_flag   = [1]
mon.proc_info_int    = [5]
mon.adm_info_flag    = [1]
mon.adm_cfg_info_int = [600]
mon.adm_rt_info_int  = [2]
mon.merge_cnt        = [10]
mon.err_buf_flag     = [1]
mon.svr_skip_num     = [20]
```

● pharosadm –l

```
ptp5.0@faust:/data/ptp5.0> pharosadm -L
-------------------------------------------------------------------------
INDEX     서버명              그룹ID       서버ID          PID
-------------------------------------------------------------------------
 0540    svr2                    1       827474       827474
-------------------------------------------------------------------------
 서버 개수 = [1]
-------------------------------------------------------------------------
ptp5.0@faust:/data/ptp5.0>
```

9. netstat를 통하여 수집서버와 연결이 되었는가 확인?

```
[pharostp@Yoona:/home/pharostp/apm/config] netstat -a | grep 44001
tcp        0      0 Yoona:51844                 1.235.117.149:44001           ESTABLISHED
tcp        0      0 Yoona:51845                 1.235.117.149:44001           ESTABLISHED
tcp        0      0 Yoona:51841                 1.235.117.149:44001           ESTABLISHED
tcp        0      0 Yoona:54216                 1.235.117.149:44001           ESTABLISHED
tcp        0      0 Yoona:30643                 1.235.117.149:44001           ESTABLISHED
tcp        0      0 Yoona:54215                 1.235.117.149:44001           ESTABLISHED
tcp        0      0 Yoona:54217                 1.235.117.149:44001           ESTABLISHED
tcp        0      0 Yoona:54218                 1.235.117.149:44001           ESTABLISHED
tcp        0      0 Yoona:51842                 1.235.117.149:44001           ESTABLISHED
tcp        0      0 Yoona:30642                 1.235.117.149:44001           ESTABLISHED
[pharostp@Yoona:/home/pharostp/apm/config]
```

10. Pharos TP 엔진 로그 파일에 에러 내용 확인?

```
(I) 173327.866130:573662:[main.c:collect_svr_connect:377] :수집서버 연결 완료: fd=[4]
(W) 173349.697123:573662:[signal.c:sig_ignore:282] :SIGNAL 발생: signal [19] catched: ignored
(E) 173349.697220:573662:[sigusrproc.c:sigusrfunc:59] :signal [15] catched: 프로그램 비정상 종료
(I) 173408.364195:504044:[main.c:collect_svr_connect:377] :수집서버 연결 완료: fd=[4]
```

- Pharos TP 엔진을 실행하면 메시지 큐를 삭제하므로 한번은 위와 같은 에러 메시지가 나옴

B. 사용자함수 Hooking방법

Pharos TP 시스템은 공유 라이브러리 형식으로 개발된 사용자 함수도 Hooking하여 Call-Tree에 추가 시킬 수 있다. 단, Hooking 하고자 하는 사용자 함수는 "static"으로 선언되어 있지 않아야 한다. 사용자 함수를 Hooking 하기 위한 절차는 다음과 같다

B.1. 사용자 함수 등록

Hooking 하고자 하는 함수는 먼저 사용자 함수 환경 파일에 등록해야 한다. 환경 파일에 등록하는 방법은 다음과 같다.

```
# custommer function info
[function]
function.name=tdlcall
function.callname_arg=funcname
function.returntype=int
function.prototype=tdlcall(char *funcname, void *args, long *urcode, int flags)

[function]
function.name=tdlcall2
function.callname_arg=funcname
function.returntype=int
function.prototype=tdlcall2(char *libname, char *funcname, void *args, long *urcode, int flags)

....

[function]
function.name=usr_module
function.callname_arg=
function.returntype=long
function.prototype=usr_module(char *data, long len, long flag)
```

사용자 함수 등록은 항상 "[function]" 레이블로 시작한다. "[function]" 레이블이 없으면 해당 함수는 무시된다.

- function.name : 사용자 함수 명을 등록한다.
- function.callname_arg : 사용자 함수가 다른 함수를 호출하는 공통 함수인 경우 실제적으로 호출하는 함수명이 있는 argument를 등록한다. 다른 함수를 호출하지 않는 일반 사용자 함수인 경우에 동 항목은 등록하지 않는다.

[다른 사용자 함수를 호출하는 사용자 함수인 경우]

```
[function]
function.name=tdlcall
function.callname_arg=funcname
function.returntype=int
function.prototype=tdlcall(char *funcname, void *args, long *urcode, int flags)
```

[일반 사용자 함수인 경우]

```
[function]
function.name=usr_module
function.callname_arg=
function.returntype=long
function.prototype=tdlcall(char *funcname, void *args, long *urcode, int flags)
```

- function.returntype : 사용자 함수의 리턴 타입을 등록한다.
- function.prototype : 사용자 함수의 Prototype을 등록한다.

B.2. 사용자 함수 Hooking 모듈 생성

환경 파일을 등록한 후에 등록된 사용자 함수 환경 파일을 기초로 하여 Hooking 할 수 있는 Hooking 모듈을 생성해야 한다. Hooking 모듈을 생성하는 방법은 아래의 명령어를 실행하면 된다.

```
PharosTP@AIX6:/home/PharosTP/apm/bin> pcustomgen -i customgen.cfg
```

위와 같이 pcustomgen 명령어를 실행하면 환경 파일을 기초로 하여 "$PHOMEDIR/work" 디렉터리에 Hooking 모듈 소스와 Makefile이 생성된다. 단, "-c" 옵션이 없는 경우에는 디폴트로 "$PHOMEDIR/config/customgen.cfg" 파일을 사용한다.

[Hooking 모듈 소스]

```
/*
 * @file                customhook.c
 * @brief               Custommer hooking source
 * @author
 * @library
 *
 * @history
*  성     명  :  일     자  :            변         경         내         용
 *  --------     --------     ---------------------------------------------------------------------
/*
* @pseudo code
*/
/* Linux requires _GNU_SOURCE macro for RTLD_NEXT */
#if defined(__linux) || defined(linux) || defined(Linux) || defined(_LINUX) || defined(_LINUX64)
#    ifndef _GNU_SOURCE
#          define _GNU_SOURCE
#    endif
#endif
/* ---------------------------------------- include files ---------------------------------------- */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <signal.h>
#include <fcntl.h>
#include <unistd.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/timeb.h>
#include <stdarg.h>
#include <math.h>
#include <strings.h>
#include <sys/param.h>
#include <sys/time.h>
#include <time.h>
#include <dlfcn.h>
#include <pthread.h>
```

```
/* ---------------------------------------- constant ---------------------------------------- */
/* ------------------------------------ macro definitions ------------------------------------ */
/* value return type function hook */
#define BEGIN_FUNCTION_HOOK_DECLARE(RETTYPE, NAME, ...)
typedef RETTYPE (*NAME##_hook_t)(__VA_ARGS__);
RETTYPE NAME(__VA_ARGS__)
{
    static NAME##_hook_t NAME##_hook = NULL;
    RETTYPE hook_rc;

    if (NAME##_hook == NULL ) {
        NAME##_hook = (NAME##_hook_t)dlsym(RTLD_NEXT, #NAME);
        if (NAME##_hook == NULL) {
            fprintf(stderr,"ERROR: unsatisfied symbol [%s]₩n", #NAME);
            exit(1);
        }
    }

#define END_FUNCTION_HOOK_DECLARE
    return hook_rc;
}

#define ORIGIN_FUNCTION_CALL_RETURN(NAME, ...)
    do {
        hook_rc = (*(NAME##_hook))(__VA_ARGS__);
    } while(0)

#define ORIGIN_FUNCTION_CALL_RETURN_NOARGS(NAME)
    do {
        hook_rc = (*(NAME##_hook))();
    } while(0)

/* void return type function hook */
#define BEGIN_FUNCTION_HOOK_DECLARE_VOID(NAME, ...)
typedef void (*NAME##_hook_t)(__VA_ARGS__);
```

```
void NAME(__VA_ARGS__)
{
	static NAME##_hook_t NAME##_hook = NULL;
	if (NAME##_hook == NULL ) {
		NAME##_hook = (NAME##_hook_t)dlsym(RTLD_NEXT, #NAME);
		if (NAME##_hook == NULL) {
			fprintf(stderr,"ERROR: unsatisfied symbol [%s]₩n", #NAME);
			exit(1);
		}
	}
#define END_FUNCTION_HOOK_DECLARE_VOID
	return;
}

#define ORIGIN_FUNCTION_CALL_VOID(NAME, ...)
	do {
		(*(NAME##_hook))(__VA_ARGS__);
	} while(0)

#define ORIGIN_FUNCTION_CALL_VOID_NOARGS(NAME)
	do {
		(*(NAME##_hook))();
	} while(0)
/* ------------------------------ structure definitions ------------------------------------- */
/* ---------------------------- exported global variables declarations ------------------------ */
/* --------------------------------- exported function declarations ---------------------------- */
extern long custom_function_start(char *);
extern long custom_function_end(char *, int);

/* ------------------------------------------------------------------------------------------

 * @function		int tdlcall()
 * @brief		tdlcall API
 * @parameter
 * @return		0	: 정상
 *			-1	: 에러
 * ------------------------------------------------------------------------------------------
 */
```

```
BEGIN_FUNCTION_HOOK_DECLARE(int, tdlcall, char *funcname, void *args, long *urcode, int flags)
{
	int rc = 0;

	/* ------------------------------------------------------------------ */
	/* user function call                                                 */
	/* ------------------------------------------------------------------ */
	custom_function_start(funcname);

	/* call the original */
	ORIGIN_FUNCTION_CALL_RETURN(tdlcall, funcname, args, urcode, flags);

	/* ------------------------------------------------------------------ */
	/* user end function call                                             */
	/* ------------------------------------------------------------------ */
	custom_function_end(funcname, hook_rc);
}
END_FUNCTION_HOOK_DECLARE


/* ---------------------------------------------------------------------------------------

 * @function		int tdlcall2()
 * @brief		tdlcall2 API
 * @parameter
 * @return		0	: 정상
 *			-1	: 에러
 * ---------------------------------------------------------------------------------------
 */
BEGIN_FUNCTION_HOOK_DECLARE(int, tdlcall2, char *libname, char *funcname, void *args, long *urcode, int flags)
{
	int rc = 0;

	/* ------------------------------------------------------------------ */
	/* user function call                                                 */
	/* ------------------------------------------------------------------ */
	custom_function_start(funcname);
```

```
/* call the original */
    ORIGIN_FUNCTION_CALL_RETURN(tdlcall2, libname, funcname, args, urcode, flags);

    /* ------------------------------------------------------------------ */
    /* user end function call                                             */
    /* ------------------------------------------------------------------ */
    custom_function_end(funcname, hook_rc);
}
END_FUNCTION_HOOK_DECLARE
/* ---------------------------------------------------------------------------------------------

 * @function          int tdlcall2s()
 * @brief             tdlcall2s API
 * @parameter
 * @return            0    : 정상
 *                    -1   : 에러
 * ---------------------------------------------------------------------------------------------
 */

BEGIN_FUNCTION_HOOK_DECLARE(int, tdlcall2s, char *libname, char *funcname, void *input, void
*output, long *urcode, int flags)
{
    int rc = 0;

    /* ------------------------------------------------------------------ */
    /* user function call                                                 */
    /* ------------------------------------------------------------------ */
    custom_function_start(funcname);

    /* call the original */
    ORIGIN_FUNCTION_CALL_RETURN(tdlcall2s, libname, funcname, input, output, urcode, flags);

    /* ------------------------------------------------------------------ */
    /* user end function call                                             */
    /* ------------------------------------------------------------------ */
    custom_function_end(funcname, hook_rc);
}
END_FUNCTION_HOOK_DECLARE
```

```
/* ------------------------------------------------------------------------------------------

 * @function          long usr_module()
 * @brief             usr_module API
 * @parameter
 * @return            0    : 정상
 *                    -1   : 에러
 * ------------------------------------------------------------------------------------------
 */
BEGIN_FUNCTION_HOOK_DECLARE(long, usr_module, char *data, long len, long flag)
{
    int rc = 0;

    /* ---------------------------------------------------------------- */
    /* user function call                                               */
    /* ---------------------------------------------------------------- */
    custom_function_start("usr_module");

    /* call the original */
    ORIGIN_FUNCTION_CALL_RETURN(usr_module, data, len, flag);

    /* ---------------------------------------------------------------- */
    /* user end function call                                           */
    /* ---------------------------------------------------------------- */
    custom_function_end("usr_module", hook_rc);
}
END_FUNCTION_HOOK_DECLARE
```

B.3. Hooking 모듈 생성 컴파일

사용자 함수 Hooking 모듈을 생성한 다음에 이를 컴파일 하여 공유 라이브러리로 만들어야 한다. 컴파일은 간단하게 함께 생성된 Makefile을 실행하면 된다.

```
ptp5.0@faust:/data/ptp5.0/pharostp/work> make -f customhook.mk
        cc -qlanglvl=extc99 -q64 -qinfo=pro -brtl -O2 -qcpluscmt -DNO_DEBUG -DNO_DEBUG -D_IBM64 -D_REENTRANT -I. -I/data/ptp5.0/pharostp/inc -I/inc -c customhook.c
        cc -G -bshared -q64 -o libcustomhook.so customhook.o -qlanglvl=extc99 -brtl -L. -L/data/ptp5.0/pharostp/lib -L/data/ptp5.0/pharostp/lib -lpharostphm -lpharostpcom -lpharos -lpthread
        mv libcustomhook.so /data/ptp5.0/pharostp/lib
Target "all" is up to date.
pharostp@Yoona:/home/pharostp/apm/work>
```

B.4. Hooking 모듈 적용

컴파일 하여 Hooking 모듈을 생성한 다음에 이를 Hooking 가능하도록 등록해야 한다. TP 모니터가 Tuxedo인 경우에는 "2.2.2 Tuxedo 서버 프로세스 환경변수 설정" 에서 설명한 방법대로 등록한다. Hooking 모듈을 두 개 이상 등록하는 경우에는 콜론(:)을 이용하여 분리한다. Tmax인 경우에는 Tmax 서버 프로세스를 다시 한번 Link 해 주어야 한다.

<.profile> / <.bash_profile>

```
# export preload
if [ $OS_VENDER = ibm64 ]; then
    export LDR_PRELOAD64=$PHOMEDIR/lib/libpharostpsh.so:$PHOMEDIR/lib/libcustomhook.so
elif [ $OS_VENDER = ibm32 ]; then
    export LDR_PRELOAD=$PHOMEDIR/lib/libpharostpsh.so::$PHOMEDIR/lib/libcustomhook.so
else
    export LD_PRELOAD=$PHOMEDIR/lib/libpharostpsh.so:$PHOMEDIR/lib/libcustomhook.so
fi
```

<.cshrc>

```
if [ $OS_VENDER = ibm64 ]; then
    setenv LDR_PRELOAD64=$PHOMEDIR/lib/libpharostpsh.so:$PHOMEDIR/lib/libcustomhook.so
elif [ $OS_VENDER = ibm32 ]; then
    setenv LDR_PRELOAD=$PHOMEDIR/lib/libpharostpsh.so::$PHOMEDIR/lib/libcustomhook.so
else
    setenv LD_PRELOAD=$PHOMEDIR/lib/libpharostpsh.so:$PHOMEDIR/lib/libcustomhook.so
fi
```

B.5. TP 시스템 재기동

마지막으로 Tuxedo 시스템인 경우에는 환경변수를 적용한 후에 재 기동하면 되고, Tmax 시스템인 경우에는 사용자 Hooking 모듈을 Link 한 이후에 재 기동하면 된다.

C. DBMS 분리방법

DBMS와 분리하여 순수하게 TP 모니터링만 원하는 경우 사용한다. 절차는 다음과 같다.

C.1. pharostpnodb 파일 변경

```
PharosTP@AIX6:/home/pharostp/apm/bin>mv pharostpnodb pharostp
```

C.2. libpharostp 파일 변경

```
PharosTP@AIX6:/home/pharostp/apm/lib>mv libpharosdb libpharos
```

※ 단 주의할 점은 기존의 파일을 백업해 두도록 권장한다.

C.3. TP 시스템 재기동

바이너리 파일을 변경하고 PharosTP 시스템을 재기동 하면, DBMS에 상관없이 순수한 TP 모니터링만 할 수 있다.

D. pharostp.cfg config 예제

D.1. Tuxedo config 예제

아래 예제는 Tuxedo 시스템에서 사용하는 Pharos TP의 config 파일 예제이다.

```
##############################################################################
# PHAROS TP config information file
##############################################################################

##############################################################################
# Instance information
##############################################################################
[instance_info]

# adapter name
instance.res_name=tmax_faust

# adapter name make flag
# true : agent name = res_name + tp_username
instance.res_name_join=false

# adapter unique number
instance.res_id=100

##############################################################################
# TP Monitor information
##############################################################################
[tpmon_info]

# 거래추적 대상 tp monitor name
tpmon.tp_name=TUXEDO

# 거래추적 대상 tp monitor version
tpmon.tp_ver=12gr1

# 거래추적 대상 tp monitor 실행 계정 명
tpmon.tp_username=tmax

# 거래추적 대상 DB 종류
tpmon.db_name=oracle
```

```
# 거래추적 대상 DB 접속정보
tpmon.db_coninfo=pharosdb/pharosdb@FAUST11G

# 거래추적 대상 DB 종류
tpmon.engine_proc=BBL, WSL, WSH, DMADM, GWADM, GWTDOMAIN

##############################################################################
# Server connection information setting
##############################################################################
[server_info]

# 수집서버 IP-ADDR
server.address=175.118.115.33

# 수집서버에 추적 정보를 전송할 포트번호 [default=8877]
server.send_port=33012

# 수집서버에 추적 정보를 전송할 세션 수 [min=1, max=10]
server.send_session_num=1

# 수집서버로부터 요청받을 포트번호 [default=8887]
server.recv_port=33012

# 수집서버로부터 요청받을 세션 수  [min=1, max=10]
server.recv_session_num=1

##############################################################################
# Monitoring information setting
##############################################################################
[monitoring]

# APM type [tp]
mon.type=tp

# APM내부 프로세스간 데이터 송수신 Message queue 키 값 [default=39829]
mon.msgkey=39969

# 거래추적 LEVEL [0=미추적 | 1=서비스 | 2=모듈 | 3=SQL | 4=ATMI]
# 거래추적 LEVEL 값은 추적하고자 하는 LEVEL보다 1큰 값을 등록해야 함
mon.trace_level=5

# 서비스 임계치(밀리초) : 임계치 이하의 데이터는 수집서버에 전송하지 않음 [default=0]
mon.svc_cp=0
```

```
# Active 서비스 리스트에 대한 임계치(밀리초) : 임계치 이상의 데이터만 전송  [default=10000]
mon.act_svc_cp=3000

# Active 서비스 리스트 전송 interval(밀리초) [default=5000]
mon.act_svc_int=5000

# 시스템의 CPU & MEMORY 정보 전송 interval(밀리초) [default=5000]
mon.sys_info_int=5000

# 거래추적 데이터에 대한 집계여부 [true|false]
mon.sum_flag=true

# 집계 데이터 전송 interval(밀리초) [default=60000]
mon.sum_send_int=60000

# 서비스 usage(CPU & Memory Leek) 생성 여부 [true|false]
mon.svc_sys_usage=true

# 프로세스별 CPU & Memory 정보 생성 여부 [true|false]
# "tpmon.tp_username" 항목에 등록된 계정의 프로세스에 한함
mon.proc_info_flag=true

# 프로세스별 CPU & Memory 정보 전송 interval(밀리초) [default=60000]
mon.proc_info_int=10000

# Request 정보 전송 interval(밀리초) [default=1000]
mon.req_send_int=1000

# TP Monitor Admin 정보 추출 여부 [true|false]
mon.adm_info_flag=true

# TP Monitor Config성 Admin 정보 전송 interval(밀리초) [default=600000]
mon.adm_cfg_info_int=600000

# TP Monitor Real Time Admin 정보 전송 interval(밀리초) [default=5000]
mon.adm_rt_info_int=5000

# 임계치 이하 데이터 버퍼링 건수 지정 [min=1, max=100]
mon.merge_cnt=10

# 임계치 이하 데이터 중 에러 데이터 버퍼링 여부 [true|false]
mon.err_buf_flag=true
```

```
# 거래추적에서 제외할 서버 프로세스 명 : comma로 구분
mon.skip_svr=BBL, DBBL, LMS, WSL, WSH, DMADM, GWADM, GWTDOMAIN, tmboot, tmshutdown, tmadmin, JSL,
JREPSVR, tmunloadcf, tlisten, tuxadm, apaddusr, tmipcrm, tpkill, tms_ora

# 거래추적에서 제외할 서비스 명 : comma로 구분
mon.skip_svc=

# 아래 거래코드만 거래추적할 것인지 아니면 제외할 것인지 여부 [positive|negative|false]
mon.txcode_flag=false
# 거래추적관련 거래코드 파일명 (default: config 파일 directory path)
mon.txcode_path=

# APM 헤더 처리 방법 [0=미사용|1=APM 헤더]
mon.apm_header_flag=0
# APM 헤더 flag가 '1'인 경우 skip 서비스 목록 : comma로 구분
mon.apm_skip_svc=

# flag for transaction tracking by hashcode : [0=Not used | 1=Message | 2=GUID]
mon.hashcode_type=0

# 비동기 호출을 하나의 call tree로 처리할 것인지 여부 [true|false]
mon.async_to_tx=false

# TP DAEMON process config 파일명
mon.proc_config=

# skip count for memory leak check : [default=10]
mon.leak_skip_cnt=10

# flag for polling data send : [true|false]
mon.polling_flag=true
# interval for polling data send(millisecond) : [default=30000]
mon.polling_int=30000
```

```
################################################################################
# TUXEDO Admin Monitoring information setting
################################################################################
[tpadmin]

# Queue Data Query Mode (1=tpadmin -r, 2=tpadmin -r file name)
tpadmin.queue_inquary=1

# Queue File Full Path
tpadmin.queue_file=

################################################################################
# Module Monitoring information setting
################################################################################
[module]

# 모듈 임계치(밀리초) : 임계치 이하 데이터는 call tree에 나타나지 않음 [default=0]
# 단, 하위에 Entry가 존재시 임계치 이하 데이터도 포함
module.mod_time_limit=0

################################################################################
# SQL Monitoring information setting
################################################################################
[sql_monitoring]

# Session 정보 수집 여부 [true|false]
sql.sid_coll_flag=true

# SQL bind data 수집 여부 [true|false]
sql.bind_coll_flag=true

# SQL plan 여부 [true|false]
sql.sql_plan_flag=true

# SQL TEXT hashing 여부 [true|false]
sql.sql_hash_flag=true

# SQL 임계치 예외 건수: 임계치와 관계없이 call tree에 포함할 개수 [default=100]
sql.sql_stack_cnt=100

# SQL 임계치(밀리초) : 임계치 이하 데이터는 call tree에 나타나지 않음 [default=0]
sql.sql_time_limit=0
```

```
# 예외처리 SQL 에러코드 [최대 128개 까지 입력 가능]
sql.skip_err_code=

################################################################################
# Monitor information setting
################################################################################
[position]

# Buffer type이 CARRAY, STRING인 경우 업무코드 위치 및 길이 (위치는 1부터 시작)
#position.appl_code_pos=96
#position.appl_code_len=4

# Buffer type이 CARRAY, STRING인 경우 거래코드 위치 및 길이 (위치는 1부터 시작)
#position.svc_name_flag=true
#position.txcode_pos=100
#position.txcode_len=10

# Buffer type이 CARRAY, STRING인 경우 글로벌ID 위치 및 길이 (위치는 1부터 시작)
#position.gid_pos=10
#position.gid_len=30

# Buffer type이 CARRAY, STRING인 경우 글로벌ID 일련번호 위치 및 길이 (위치는 1부터 시작)
#position.gid_seq_pos=40
#position.gid_seq_len=2

# Buffer type이 CARRAY, STRING인 경우 사용자 메시지 위치 및 길이 (위치는 1부터 시작)
#position.user_msg_pos=10
#position.user_msg_len=30
```

```
#############################################################################
# Application error information setting
#############################################################################
[app_errinfo]

# 에러전문 판단정보. 메시지판단 위치정보, 메시지판단 코드길이, 에러전문코드 값
# 위치정보는 1부터 시작
#errinfo.errmsg_pos=221
#errinfo.errmsg_len=1
#errinfo.errmsg_code=2

# 에러코드 위치 정보 (위치는 1부터 시작)
#errinfo.code_pos=771
#errinfo.code_len=10

# 에러메시지 위치 정보 (위치는 1부터 시작)
#errinfo.msg_pos=781
#errinfo.msg_len=40

#############################################################################
# Trace Logging information setting
#############################################################################
[log_info]

# 서비스 종료 로그 출력 여부
log.log_flag=true

# Hooking 에러로그 로깅 path
#log.log_path=/home/pharostp/apm/log

# TP Monitor sys log file name
log.sys_log_path=/home/tmax/tmax5/log/slog/slog.$(MMDDYYYY)

# |=or, &=and, !=not ex) ERROR|WARN|!tms
log.sys_log_pattern=(E)|(W)|(I)

# interval for log file check (밀리초)
log.sys_log_interval=5000

# Core File info
log.core_file_flag=false
log.core_file_path=/home/pharostp/apm/log/corelog
```

```
##############################################################################
# Monitoring Global Area information setting
##############################################################################
[mga_info]

# APM 정보 저장을 위한 공유메모리 키 값 [default=78437]
mga.shmkey=78847

# 거래추적 대상 서버프로세스 수  [default=512]
mga.svr_process=1000

# 서비스 단위별 거래추적 총 호출 건수  [max=1024]
mga.call_seq=512

# SQL TEXT 저장 개수  [default=10000]
mga.sql_hash_num=100000
```

D.2. Tmax config 예제

아래 예제는 Tmax 시스템에서 사용하는 Pharos TP의 config 파일 예제이다.

```
##############################################################################
# PHAROS TP config information file
##############################################################################

##############################################################################
# Instance information
##############################################################################
[instance_info]

# adapter name
instance.res_name=tmax_faust

# adapter name make flag
# true : agent name = res_name + tp_username
instance.res_name_join=false

# adapter unique number
instance.res_id=100

##############################################################################
# TP Monitor information
##############################################################################
[tpmon_info]

# 거래추적 대상 tp monitor name
tpmon.tp_name=TUXEDO

# 거래추적 대상 tp monitor version
tpmon.tp_ver=12gr1

# 거래추적 대상 tp monitor 실행 계정 명
tpmon.tp_username=tmax

# 거래추적 대상 DB 종류
tpmon.db_name=oracle

# 거래추적 대상 DB 접속정보
tpmon.db_coninfo=pharosdb/pharosdb@FAUST11G
```

```
# 거래추적 대상 DB 종류
tpmon.engine_proc=tmm,clh,cll,tms_ora

##############################################################################
# Server connection information setting
##############################################################################
[server_info]

# 수집서버 IP-ADDR
server.address=175.118.115.33

# 수집서버에 추적 정보를 전송할 포트번호 [default=8877]
server.send_port=33012

# 수집서버에 추적 정보를 전송할 세션 수 [min=1, max=10]
server.send_session_num=1

# 수집서버로부터 요청받을 포트번호 [default=8887]
server.recv_port=33012

# 수집서버로부터 요청받을 세션 수  [min=1, max=10]
server.recv_session_num=1

##############################################################################
# Monitoring information setting
##############################################################################
[monitoring]

# APM type [tp]
mon.type=tp

# APM내부 프로세스간 데이터 송수신 Message queue 키 값 [default=39829]
mon.msgkey=39969

# 거래추적 LEVEL [0=미추적 | 1=서비스 | 2=모듈 | 3=SQL | 4=ATMI]
# 거래추적 LEVEL 값은 추적하고자 하는 LEVEL보다 1큰 값을 등록해야 함
mon.trace_level=5

# 서비스 임계치(밀리초) : 임계치 이하의 데이터는 수집서버에 전송하지 않음 [default=0]
mon.svc_cp=0
```

```
# Active 서비스 리스트에 대한 임계치(밀리초) : 임계치 이상의 데이터만 전송  [default=10000]
mon.act_svc_cp=3000

# Active 서비스 리스트 전송 interval(밀리초) [default=5000]
mon.act_svc_int=5000

# 시스템의 CPU & MEMORY 정보 전송 interval(밀리초) [default=5000]
mon.sys_info_int=5000

# 거래추적 데이터에 대한 집계여부 [true|false]
mon.sum_flag=true

# 집계 데이터 전송 interval(밀리초) [default=60000]
mon.sum_send_int=60000

# 서비스 usage(CPU & Memory Leek) 생성 여부 [true|false]
mon.svc_sys_usage=true

# 프로세스별 CPU & Memory 정보 생성 여부 [true|false]
# "tpmon.tp_username" 항목에 등록된 계정의 프로세스에 한함
mon.proc_info_flag=true

# 프로세스별 CPU & Memory 정보 전송 interval(밀리초) [default=60000]
mon.proc_info_int=10000

# Request 정보 전송 interval(밀리초) [default=1000]
mon.req_send_int=1000

# TP Monitor Admin 정보 추출 여부 [true|false]
mon.adm_info_flag=true

# TP Monitor Config성 Admin 정보 전송 interval(밀리초) [default=600000]
mon.adm_cfg_info_int=600000

# TP Monitor Real Time Admin 정보 전송 interval(밀리초) [default=5000]
mon.adm_rt_info_int=5000

# 임계치 이하 데이터 버퍼링 건수 지정 [min=1, max=100]
mon.merge_cnt=10

# 임계치 이하 데이터 중 에러 데이터 버퍼링 여부 [true|false]
mon.err_buf_flag=true
```

```
# 거래추적에서 제외할 서버 프로세스 명 : comma로 구분
mon.skip_svr=tmm, clh, cll, tms_ora

# 거래추적에서 제외할 서비스 명 : comma로 구분
mon.skip_svc=

# 아래 거래코드만 거래추적할 것인지 아니면 제외할 것인지 여부 [positive|negative|false]
mon.txcode_flag=false
# 거래추적관련 거래코드 파일명 (default: config 파일 directory path)
mon.txcode_path=

# APM 헤더 처리 방법 [0=미사용|1=APM 헤더]
mon.apm_header_flag=0
# APM 헤더 flag가 '1'인 경우 skip 서비스 목록 : comma로 구분
mon.apm_skip_svc=

# flag for transaction tracking by hashcode : [0=Not used | 1=Message | 2=GUID]
mon.hashcode_type=0

# 비동기 호출을 하나의 call tree로 처리할 것인지 여부 [true|false]
mon.async_to_tx=false

# TP DAEMON process config 파일명
mon.proc_config=

# skip count for memory leak check : [default=10]
mon.leak_skip_cnt=10

# flag for polling data send : [true|false]
mon.polling_flag=true
# interval for polling data send(millisecond) : [default=30000]
mon.polling_int=30000
```

```
################################################################################
# Module Monitoring information setting
################################################################################
[module]

# 모듈 임계치(밀리초) : 임계치 이하 데이터는 call tree에 나타나지 않음 [default=0]
# 단, 하위에 Entry가 존재시 임계치 이하 데이터도 포함
module.mod_time_limit=0

################################################################################
# SQL Monitoring information setting
################################################################################
[sql_monitoring]

# Session 정보 수집 여부 [true|false]
sql.sid_coll_flag=true

# SQL bind data 수집 여부 [true|false]
sql.bind_coll_flag=true

# SQL plan 여부 [true|false]
sql.sql_plan_flag=true

# SQL TEXT hashing 여부 [true|false]
sql.sql_hash_flag=true

# SQL 임계치 예외 건수: 임계치와 관계없이 call tree에 포함할 개수 [default=100]
sql.sql_stack_cnt=100

# SQL 임계치(밀리초) : 임계치 이하 데이터는 call tree에 나타나지 않음 [default=0]
sql.sql_time_limit=0

# 예외처리 SQL 에러코드 [최대 128개 까지 입력 가능]
sql.skip_err_code=
```

```
################################################################################
# Monitor information setting
################################################################################
[position]

# Buffer type이 CARRAY, STRING인 경우 업무코드 위치 및 길이 (위치는 1부터 시작)
#position.appl_code_pos=96
#position.appl_code_len=4

# Buffer type이 CARRAY, STRING인 경우 거래코드 위치 및 길이 (위치는 1부터 시작)
#position.svc_name_flag=true
#position.txcode_pos=100
#position.txcode_len=10

# Buffer type이 CARRAY, STRING인 경우 글로벌ID 위치 및 길이 (위치는 1부터 시작)
#position.gid_pos=10
#position.gid_len=30

# Buffer type이 CARRAY, STRING인 경우 글로벌ID 일련번호 위치 및 길이 (위치는 1부터 시작)
#position.gid_seq_pos=40
#position.gid_seq_len=2

# Buffer type이 CARRAY, STRING인 경우 사용자 메시지 위치 및 길이 (위치는 1부터 시작)
#position.user_msg_pos=10
#position.user_msg_len=30
```

```
#############################################################################
# Application error information setting
#############################################################################
[app_errinfo]

# 에러전문 판단정보. 메시지판단 위치정보, 메시지판단 코드길이, 에러전문코드 값
# 위치정보는 1부터 시작
#errinfo.errmsg_pos=221
#errinfo.errmsg_len=1
#errinfo.errmsg_code=2

# 에러코드 위치 정보 (위치는 1부터 시작)
#errinfo.code_pos=771
#errinfo.code_len=10

# 에러메시지 위치 정보 (위치는 1부터 시작)
#errinfo.msg_pos=781
#errinfo.msg_len=40

#############################################################################
# Trace Logging information setting
#############################################################################
[log_info]

# 서비스 종료 로그 출력 여부
log.log_flag=true

# Hooking 에러로그 로깅 path
#log.log_path=/home/pharostp/apm/log

# TP Monitor sys log file name
log.sys_log_path=/home/tmax/tmax5/log/slog/slog.$(MMDDYYYY)

# |=or, &=and, !=not ex) ERROR|WARN|!tms
log.sys_log_pattern=(E)|(W)|(I)

# interval for log file check (밀리초)
log.sys_log_interval=5000

# Core File info
log.core_file_flag=false
log.core_file_path=/home/pharostp/apm/log/corelog
```

```
#############################################################################
# Monitoring Global Area information setting
#############################################################################
[mga_info]

# APM 정보 저장을 위한 공유메모리 키 값 [default=78437]
mga.shmkey=78847

# 거래추적 대상 서버프로세스 수  [default=512]
mga.svr_process=1000

# 서비스 단위별 거래추적 총 호출 건수  [max=1024]
mga.call_seq=512

# SQL TEXT 저장 개수  [default=10000]
mga.sql_hash_num=100000
```

E. 성능 최적화 방법

E.1. 임계치 조정

거래량의 폭주로 인하여 수집서버에 부하가 심하다면 Pharos TP 에이전트에서 거래량을 조절하여 수집서버에 전송되는 성능정보 데이터량을 줄일 수 있다. 수집서버로 전송되는 데이터의 대부분은 Request 단위로 전송되는 Call-Tree 정보이다. 이를 줄임으로써 수집서버의 부하를 줄일 수 있다.

1. pharostp.cfg 파일에서「monitoring」절의「mon.svc_cp」항목에 밀리초 단위로 Request에 대한 임계값을 설정한다. 동 항목은 Request에 대해서 지정된 시간 이하에 대해서는 해당 Request에 대한 통계정보만 전송하고 Call-Tree 정보를 전송하지 않도록 함으로써 수집서버로 전송되는 데이터 량을 획기적으로 줄일 수 있다. 예를 들어 동 항목에 1000을 설정하면 1초 미만의 Request에 대해서는 통계정보만 전송한다.

```
[monitoring]

# 2초 미만의 데이터에 대해서 통계정보만 전송하고자 하는 경우
mon.svc_cp=2000
```

2. 다음으로「monitoring」절의「mon.merge_cnt」항목에 한번에 수집서버로 전송할 건수를 등록한다. 동 항목은 임계값 이하의 통계정보를 매번 수집서버로 전송하면 부하가 발생하기 때문에 일정 개수로 모아서 한번에 전송함으로써 부하를 줄이기 위함이다. 동 항목에 등록된 건수만큼 모았다가 한번에 전송한다.

```
[monitoring]

# 50건씩 임계값 이하의 통계 데이터를 모았다가 한번에 전송한다.
mon.merge_cnt=50
```

3. 다음으로「monitoring」절의「mon.err_buf_flag」항목을 설정한다. Request 중 에러가 발생한 Request는 처리시간이 임계값 이하더라도 전체 Call-Tree 정보를 전송할 것인지 여부를 등록한다.「false」로 등록하면 처리시간이 임계값 이하일 지라도 Call-Tree 정보를 전송하고,「true」로 등록하면 정상 Request와 동일하게 임계값 이하의 Request는 통계 데이터만 전송한다.

```
[monitoring]

# 에러가 방생한 Request는 Call-Tree 정보를 전송하지 안고 통계 데이터만 전송
mon.err_buf_flag =true
```

위와 같이 Config 정보를 수정한 후에 Pharos TP 에이전트를 Restart한다.

E.2. SQL Call-Tree 조정

업무 AP 프로그램에서 SQL 질의가 많은 경우 Request에 대해서 Call-Tree 건수가 많아져 수집서버에 부하를 줄 수 있다. 모든 SQL 문장에 대해서 조사하면 이를 분석하는데 더 많은 시간이 소요되기 때문에 실질적으로 분석이 필요한 SQL에 대해서는 조사하는게 유리하다. 이렇게 하기 위해서 간단하게 Config의 몇 가지 항목만 변경하면 된다.

1. pharostp.cfg 파일에서「sql_monitoring」절의「sql.sql_stack_cnt」항목에 SQL 임계값에 관계없이 항상 조사해야 할 SQL 건수를 등록한다. 즉, 동 항목에 등록된 건수까지는 다음 항목에서 등록하는 SQL 임계값에 관계없이 항상 Call-Tree에 포함된다.

```
[sql_monitoring]

# 최소 100건까지 SQL 문장을 SQL 임계값에 관계없이 Call-Tree에 포함됨
sql.sql_stack_cnt=100
```

2. 다음으로 「sql_monitoring」 절의 「mon.sql_time_limit」 항목에 SQL 임계값을 등록한다. 「sql.sql_stack_cnt」항목에 등록된 건수를 초과하는 SQL에 대해서는 동 항목에 등록된 임계값 이하의 SQL은 Call-Tree에 포함하지 않고 통계정보만 생성한다. 이렇게 함으로써 Call-Tree 개수를 줄임으로써 수집서버의 부하를 줄일 수 있다.

```
[monitoring]

# SQL 처리시간이 0.1초 미만의 SQL은 Call-Tree에 포함하지 않는다.
mon.sql_time_limit=100
```

위와 같이 Config 정보를 수정한 후에 Pharos TP 에이전트를 Restart한다.

E.3. Function Call-Tree 조정

SQL 뿐만 아니라 업무 AP 프로그램에서 사용자 함수를 많이 호출하는 경우에도 동일하게 해당 함수에 대한 임계값을 설정하여 Call-Tree 개수를 줄일 수 있다. 함수에 대한 임계값을 조절하는 방법은 아래와 같다. 단, 해당 함수 내에서 다른 함수를 호출하거나 또는 SQL 질의가 있는 경우, 즉, 함수 내부에 하위 Call-Tree가 있는 경우에는 임계값에 적용되지 않고 항상 Call-Tree에 포함된다.

1. pharostp.cfg 파일에서「module」절의「module.mod_time_limit」항목에 함수의 임계값을 설정한다. 동 항목에 설정된 임계값 이하의 함수에 대해서는 Call-Tree에 포함하지 않는다. 단, 하위 함수 내에 하위 Call-Tree가 있으면 임계값에 관계없이 포함한다.

```
[module]

# 함수의 처리시간이 1초 미만인 경우 Call-Tree에 포함시키지 않는다.
module.mod_time_limit=1000
```

위와 같이 Config 정보를 수정한 후에 Pharos TP 에이전트를 Restart한다.

F. SQL PLAN 기능 설정 및 확인

F.1. SQL PLAN 기능 설정

SQL PLAN 기능은 agent 에서 DB 서버로 '실행계획' 요청을 보내고, 그 결과를 받아 처리 하는 방식이기 때문에 agent(pharostp 프로세스)를 위한 DB 계정과 그에 따르는 여러가지 권한이 필히 있어야 한다. 따라서 이 기능을 설정 하기 전에 DBA 와 협의를 하여 적절한 계정과 권한을 할당 받아야 한다.

< Agent 에서 EXPLAIN PLAN 을 수행 시 Oracle 에서 제공하는 'PLAN_TABLE' 을 사용하지 않는다. 내부적으로 'PHAROS_PLAN_TABLE' 을 자동 생성하여 사용한다. 따라서 DB 계정의 권한 중 테이블에 대한 CRUD 및 생성 권한이 필히 있어야 한다. >

SQL PLAN 을 기능을 정상적으로 활성화 하기 위한 DB 계정의 권한은 다음과 같다.

① 테이블 생성 권한 및 CRUD 권한
② Oracle Dictionary 의 ALL_OBJECTS select 권한
③ Oracle Dictionary 의 ALL_INDEXES select 권한
④ Oracle Dictionary 의 ALL_TABLES select 권한
⑤ Oracle Dictionary 의 ALL_TAB_COLUMNS select 권한
⑥ Oracle Dictionary 의 TAB select 권한
⑦ 스키마가 다른 대상 테이블에 대한 select 권한 (또는 타 스키마 테이블에 대한 public synonym 사용으로 가능)

1. pharostp.cfg 파일에서「tpmon_info」절의「tpmon.db_name」과「tpmon.db_coninfo」항목을 환경에 맞게 설정한다.「tpmon.db_coninfo」항목의 값은 agent 가 DB 접속 시 및 실행계획 요청 시에 사용되는 계정이다.

```
##############################################################################
# TP Monitor information
##############################################################################
[tpmon_info]

# database kind
tpmon.db_name=oracle

# database connection info
tpmon.db_coninfo=userid/passwd@sid
```

2. pharostp.cfg 파일에서「sql_monitoring」절의 항목들 중 아래의 3 항목을 모두 'true' 로 설정을 한 후 agent 를 재기동 한다.

```
#############################################################################
# SQL Monitoring information setting
#############################################################################
[sql_monitoring]

# flag for collection database session id : [true|false]
sql.sid_coll_flag=true

# flag for collection bind data : [true|false]
sql.bind_coll_flag=true

# flag for sql plan : [true|false]
sql.sql_plan_flag=true
```

F.2. DB 계정 권한 확인

DBA 로부터 부여 받아 pharostp.cfg 파일에 설정한 DB 계정의 각 권한을 확인한다. 이때 oracle 에서 제공하는 sqlplus 유틸리티를 이용하여 확인한다.

다음은 DB 계정 권한에 대한 체크 리스트이다.

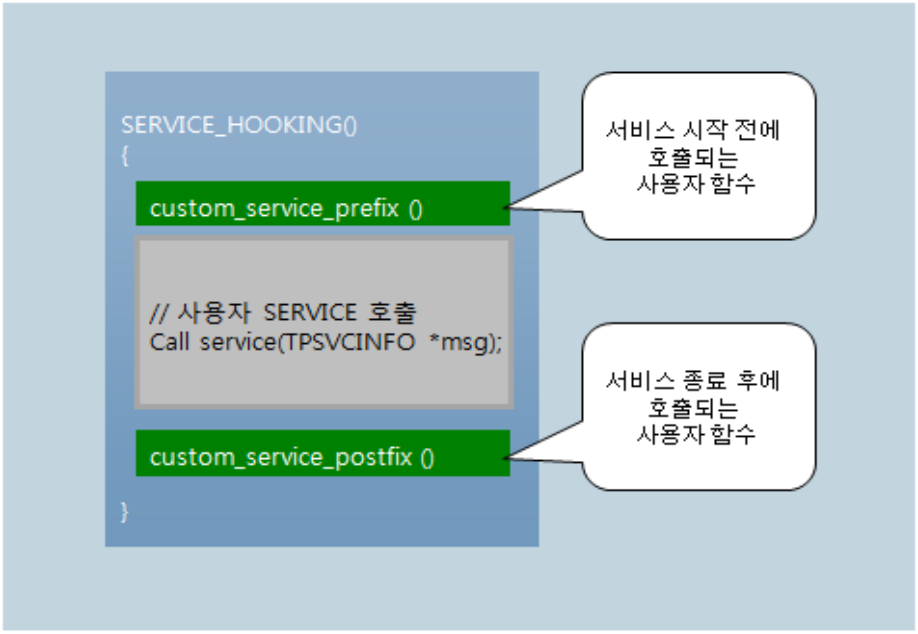
1. DB 계정으로 접속 확인
   SQL> sqlplus userid/passwd@sid
2. DB 계정으로 접속하여 다음의 쿼리들이 정상 동작 하는지 확인
   ① SQL> select count(*) form ALL_TABLES;
   ② SQL> select count(*) form ALL_OBJECTS;
   ③ SQL> select count(*) form ALL_INDEXES;
   ④ SQL> select count(*) form ALL_TAB_COLUMNS;
   ⑤ SQL> select count(*) form TAB;

3. 타 스키마 테이블에 대한 select 가능 여부 확인
4. 실행 계획 쿼리 수행 가능 여부 확인
   SQL> EXPLAIN PLAN SET STATEMENT_ID='abcd1234' FOR select count(*) form TAB;

G. Advice 프로그램 작성 방법

G.1. 호출 위치

Advice 프로그램은 크게 두 곳에서 호출한다. 하나는 서비스가 시작되는 시점에서 호출되고 다른 하나는 서비스가 종료된 이후에 호출된다. 정확한 호출 위치는 아래 그램과 같다.



G.2. 호출 함수

Advice 프로그램은 트게 두 개의 함수로 구성된다. 하나는 서비스가 시작되는 시점에서 호출되는 함수이고 다른 하나는 서비스가 종료된 이후에 호출되는 함수이다. 함수명은 고정되어 있으며 사용자마다 다르게 작명할 수는 없다.

[시작함수]

```
/* ------------------------------------------------------------------------------------- */
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;

    printf("=========================================================₩n");
    printf("custom_service_prefix start₩n");
    printf("gid=[%s] seqno=[%s]₩n", gid, seqno);
    printf("=========================================================₩n");

    ....

    return 0;
}
```

[시작함수 prototype]

int custom_service_prefix(const char *svcname, const char *data, int len)

설　　명: 동 함수는 사용자 서비스가 시작되기 전에 필요한 처리를 위해서 호출하는 함수이다.
아규먼트: const char *svcname : 호출된 TP 서비스 명
const char *data : 서비스를 호출할 때 전달한 전문 데이터
int len : 전문 데이터 길이
리턴 값 : 시작 함수는 리턴 값이 중요 함
0 : 정상적으로 성능정보 추출을 함. 즉, UI에 Call-Tree 정보가 표현됨
-1: 성능정보를 추출하지 않음. 즉, UI에 Call-Tree 정보가 표현되지 않고 또한 통계정보도 생성하지 않음
기타: 0과 동일하게 처리

[종료함수]

```
/* ------------------------------------------------------------------------------------------ */
int
custom_service_postfix(const char *svcname, const char *data, int len)
{
    long rc = 0;

    printf("=================================================₩n");
    printf("custom_service_postfix start₩n");
    printf("gid=[%s] seqno=[%s]₩n", gid, seqno);
    printf("=================================================₩n");

return 0;
}
```

[종료함수 prototype]

int custom_service_postfix(const char *svcname, const char *data, int len)

설　　명: 동 함수는 사용자 서비스가 종료된 후에 필요한 처리를 위해서 호출하는 함수이다.
아규먼트: const char *svcname : 호출된 TP 서비스 명
const char *data : 서비스를 호출할 때 전달한 전문 데이터
int len : 전문 데이터 길이
리턴 값 : 종료 함수의 리턴 값은 의미 없음

G.3. 헤더 파일

Advice 프로그램에서 사용할 수 있는 함수가 정의된 파일로 아래와 같다.

```
/*
 * prefix advice function
 * 서비스를 시작할 때 호출되는 사용자 함수
 * return : 0 <  : 해당 서비스에 대해서 성능정보를 추출하지 않음
 *          0 >= : 해당 서비스에 대해서 성능정보 추출
 */
int  custom_service_prefix(const char *svcname, const char *data, int len);

/*
 * postfix advice function
 * 서비스를 종료할 때 호출되는 사용자 함수
 * return : 무시
 */
int  custom_service_postfix(const char *svcname, const char *data, int len);

/*
 * getter function
 */
void getguid(char *guid);
void getsvrname(char *svrname);
void getguidseq(char *seqno);
void getselfseqno(char *selfseq);
void getcallseqno(char *svcname, char *callseq);

/*
 * setter function
 */
int  setguid(char *guid);
int  setguidseq(char *seqno);
int  setservicename(char *svcname);
int  settxcode(char *txcode);
int  setchnlcode(char *chnlcode);
int  setselfseqno(char *selfseq);
int  setcallseqno(char *svcname, char *callseq);
int  setparmdata(char *key, char *data);
int  setapperror(char *code, char *msg);
```

G.4. 호출 가능한 함수 설명

Advice 프로그램에서 호출 가능한 함수별 설명은 아래와 같다.

void getguid(char *guid)

설명: 성능정보 추출을 위한 유일한 키인 글로벌 ID를 조회하는 함수이다.

아규먼트: char *guid : 글로벌 ID를 저장할 버퍼의 포인터이다.

리턴값: 없음

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    char gid[64];

    memset(gid,   0x00, sizeof(gid));
    getguid(gid);

    printf("=====================================================₩n");
    printf("gid=[%s]₩n", gid,);
    printf("=====================================================₩n");

    ....
}
```

void getsvrname(char *svrname)

설명: 해당 서비스를 포함하고 있는 TP 서버 프로세스 명을 조회하는 함수이다.

아규먼트: char *svrname : 서버 프로세스 명을 저장할 버퍼의 포인터이다.

리턴값: 없음

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    char svrname[32];

    memset(svrname,  0x00, sizeof(svrname));
    getsvrname(svrname);

    printf("=======================================================₩n");
    printf("svrname=[%s]₩n", svrname,);
    printf("=======================================================₩n");

    ....
}
```

void getguidseq(char *seqno)

설명: 표준전문에 시스템간 호출시 호출번호로 사용하는 호출일련번호 항목이 있는 경우에 해당 항목의 값을 조회하는 함수이다.

아규먼트: char *seqno : 호출일련번호 값을 저장할 버퍼의 포인터이다.

리턴값: 없음

주의: 동 항목의 값을 조회하기 위해서는 환경파일(pharostp.cfg)에 아래의 항목을 설정해야 한다.

position.gid_seq_pos=40

position.gid_seq_len=2

자세한 설명은 POSITION 카테고리에서 설명한 부분을 참조하기 바랍니다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    char seqno[10];

    memset(seqno,  0x00, sizeof(seqno));
    getguidseq(seqno);

    printf("=======================================================₩n");
    printf("seqno=[%s]₩n", seqno,);
    printf("=======================================================₩n");

    ....
}
```

void getselfseqno(char *selfseq)

설명: 해당 서비스가 거래추적에 포함되는 경우에 서비스간 호출시 호출관계를 위한 번호를 할당합니다. 호출번호는 서비스 자신의 자체번호와 호출할 때 할당하는 호출번호가 있다. 두 개의 번호는 쌍으로 동일해야 한다. 동 함수는 자신의 자체번호를 조회하는 함수이다.

아규먼트: char *selfseq : 자신의 자체번호를 저장할 버퍼의 포인터이다.
리턴값: 없음

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    char selfseq[20];

    memset(selfseq,  0x00, sizeof(selfseq));
    getselfseqno(selfseq);

    printf("=========================================================₩n");
    printf("self seqno=[%s]₩n", selfseq);
    printf("=========================================================₩n");

    ....
}
```

void getcallseqno(char *svcname, char *callseq)

설명: 해당 서비스가 거래추적에 포함되는 경우에 서비스간 호출시 호출관계를 위한 번호를 할당합니다. 호출번호는 서비스 자신의 자체번호와 호출할 때 할당하는 호출번호가 있다. 두 개의 번호는 쌍으로 동일해야 한다. 동 함수는 호출번호를 조회하는 함수이다.

아규먼트: char *svcname : 서비스마다 호출번호가 다르기 때문에 조회하고자 하는 서비스 명을 지정한다.
char *callfseq : 호출번호를 저장할 버퍼의 포인터이다.

리턴값: 없음

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    char callseq[20];

    memset(callseq,  0x00, sizeof(callseq));
    getcallseqno("A0101_A", callseq);

    printf("=========================================================₩n");
    printf("A0101_A call seqno=[%s]₩n", callseq);
    printf("=========================================================₩n");

    ....
}
```

int setguid(char *guid)

설명: 표준전문이나 Hooking 모듈에서 자체적으로 생성한 글로벌 ID가 아니라 새로운 글로벌 ID로 해당 서비스의 성능정보를 추출하고자 하는 경우에 글로벌 ID를 새롭게 설정하는 함수이다. 동 함수를 호출하면 새롭게 지정된 글로벌 ID로 설정하기 때문에 동 서비스가 거래추적에 포함되는 경우에 글로벌 ID가 변경되면 거래추적에서 제외될 수도 있다.
표준전문에서 글로벌 ID를 가져오는 경우에 표준전문 데이터까지 변경되지 않는다.

아규먼트: char *guid : 새로운 글로벌 ID를 지정한다.

리턴값: 0 : 정상 처리
-1 : 처리 오류

주의: 글로벌 ID의 길이는 48자리를 넘을 수 없다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    char gid[64];

    memset(gid,  0x00, sizeof(gid));
    sttrcpy(gid, "123456789012345678901234567890123456789012");
    setguid(gid);
    getguid(gid);

    printf("=============================================================₩n");
    printf("gid=[%s]₩n", gid);
    printf("=============================================================₩n");

    ....
}
```

int setguidseq(char *seqno)

설명: 표준전문에 있는 호출일련번호 항목은 시스템 내의 최초 업무서비스 내에서 증가한다. 그러나 Hooking 모듈은 업무 서비스가 실행되기 전에 이미 표준전문에서 호출일련번호를 가져다가 사용하기 때문에 업무서비스의 호출일련번호와 맞지 않게 된다. 이러한 경우에 호출일련번호를 재 설정할 때 사용할 수 있는 함수이다.

표준전문의 호출일련번호 항목은 변경되지 않는다.

아규먼트: char *seqno : 새로운 저장할 호출일련번호를 지정한다.

리턴값: 0 : 정상 처리

-1 : 처리 오류

주의: 호출일련번호 길이는 8자리를 넘을 수 없다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    long val;
    char seqno[10];

    if (strcmp(svcname, "A0101_1") == 0) {
        memset(seqno, 0x00, sizeof(seqno));
        getguidseq(seqno);
        val = atol(seqno) + 1;
        sprintf(seqno, "%ld", val);
        setguidseq(seqno);

        printf("=============================================================₩n");
        printf("seqno=[%s]₩n", seqno);
        printf("=============================================================₩n");
    }
    ....
}
```

int setservicename(char *svcname)

설명: 서비스 명을 변경하고자 하는 경우에 사용할 수 있는 함수이다. 모든 성능정보에 대한 통계나 Call-Tree는 변경된 서비스 명으로 출력된다.

아규먼트: char *svcname : 새롭게 변경할 서비스 명을 지정한다.

리턴값: 0 : 정상 처리
-1 : 처리 오류
주의: 서비스명의 길이는 16자리를 넘을 수 없다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;

    if (strcmp(svcname, "B0101_1") == 0)
        setservicename("B0101_2");

    ....
}
```

int settxcode(char *txcode)

설명: 해당 서비스가 처리하고 있는 거래의 거래코드를 설정할 수 있는 함수이다. 환경파일(pharostp.cfg)의 설정을 통하여 거래코드를 표준전문에서 구할 수도 있고, 또는 사용자가 전문에서 거래코드를 구하여 동 함수를 이용하여 설정할 수도 있다.

아규먼트: char *txcode : 새롭게 변경할 거래코드를 지정한다.

리턴값: 0 : 정상 처리
-1 : 처리 오류

주의: 거래코드 길이는 24자리를 넘을 수 없다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;

    if (strcmp(svcname, "B0101_1") == 0)
        settxcode("TX00001");

    ....
}
```

int setchnlcode(char *chnlcode)

설명: 해당 서비스가 처리하고 있는 거래의 채널코드를 설정할 수 있는 함수이다. 환경파일(pharostp.cfg)의 설정을 통하여 채널코드를 표준전문에서 구할 수도 있고, 또는 사용자가 전문에서 채널코드를 구하여 동 함수를 이용하여 설정할 수도 있다.

아규먼트: char *chnlcode : 새롭게 변경할 채널코드를 지정한다.

리턴값: 0 : 정상 처리
-1 : 처리 오류

주의: 채널코드 길이는 8자리를 넘을 수 없다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;

    if (strcmp(svcname, "B0101_1") == 0)
        setchnlcode("C01");

    ....
}
```

int setselfseqno(char *selfseq)

설명: 해당 서비스가 거래추적에 포함되는 경우에 서비스간 호출시 호출관계를 위한 번호를 할당합니다. Hooking 모듈에서 생성한 자신의 일련번호를 사용하는데 때에 따라서 이렇게 생성한 일련번호가 거래추적에 맞지 않을 수도 있습니다. 이러한 경우에 사용자가 일련번호를 만들어서 설정할 수 있는 함수이다.

예를 들어 일련번호를 전문데이터를 이용하여 해쉬코드로 사용하는 경우에 거래추적에서 최초 서비스는 "0"번이어야 하는데 해쉬코드를 사용하기 때문에 "0"번이 되지 않는 문제가 발생합니다. 이러한 경우에 동 함수를 사용하여 자신의 일련번호를 "0"번으로 설정할 수 있습니다.

아규먼트: char *selfseq : 새롭게 변경할 자신의 일련번호를 지정한다.

리턴값: 0 : 정상 처리

-1 : 처리 오류

주의: 자신의 일련번호 길이는 32자리를 넘을 수 없다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;

    if (strcmp(svcname, "A0101_1") == 0)
        setselfseqno("0");

    ....
}
```

int setcallseqno(char *svcname, char *callseq)

설명: 해당 서비스가 거래추적에 포함되는 경우에 서비스간 호출시 호출관계를 위한 번호를 할당합니다. 호출번호는 서비스 자신의 자체번호와 호출할 때 할당하는 호출번호가 있다. 두 개의 번호는 쌍으로 동일해야 한다. 동 함수는 새로운 호출번호를 설정할 수 있는 함수이다.

아규먼트: char *svcname : 서비스마다 호출번호가 다르기 때문에 설정하고자 하는 서비스 명을 지정한다.

char *callfseq : 새롭게 변경할 호출번호를 지정한다.

리턴값: 0 : 정상 처리

-1 : 처리 오류

주의: 호출번호 길이는 32자리를 넘을 수 없다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    unsigned int hashcode;
    char buf[64];

    if (strcmp(svcname, "C0101_1") == 0) {
        hashcode = makehashcode(data, 100);
        sprintf(buf, "%d", hashcode);
        setcallseqno("D0101_1", buf);
    }
    ....
}
```

int setparmdata(char *key, char *value)

설명: UI의 메타정보에 표현할 데이터를 저장하는 함수이다. 동 함수는 key=value 형식으로 저장하며 UI의 메타정보에도 동일하게 표현된다.

아규먼트: char *key : 저장할 값에 대한 key 값을 지정한다.

char *value : UI에 표현할 값을 지정한다.

리턴값: 0 : 정상 처리

-1 : 처리 오류

주의: key=value의 길이가 300자리를 넘을 수 없다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    char value[24];

    if (strcmp(svcname, "A0101_1") == 0) {
        memset(value, 0x00, sizeof(value));
        strncpy(value, &data[100], 20);
        setparmdata("USRMSG", value);
    }
    ....
}
```

int setapperror(char *code, char *msg)

설명: 업무 에러코드와 메시지를 등록하는 함수이다. 동 함수를 통하여 등록한 거래는 업무 서비스에서 에러 발생한 것을 의미한다. 그렇기 때문에 UI의 성능분포도에도 업무에러로 표현되며 업무에로 통계로 잡히게 된다. 또한 메타정보에도 업무에러 코드가 표현된다.

아규먼트: char *code : 업무에러 코드를 의미한다.

char *msg : 업무 에러코드에 해당하는 메시지를 의미한다.

리턴값: 0 : 정상 처리

-1 : 처리 오류

주의: key=value의 길이가 300자리를 넘을 수 없다.

```
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
    long rc = 0;
    char code[10];
    char value[100];

    if (strcmp(svcname, "A0101_1") == 0) {
        memset(code, 0x00, sizeof(code));
        memset(msg,  0x00, sizeof(msg));
        strncpy(code, &data[64], 8);
        strncpy(msg, &data[245], 80);
        setapperror(code,  msg);
    }
    ....
}
```

G.5. Makefile

Makefile 파일은 Linux 기준으로 설명한다.

```
#=================================================================#
# makefile
#=================================================================#
BASE            = knb
TARGET          = lib$(BASE).so

# ------------------------------------------------------------------------------------------ #
# Objects
# ------------------------------------------------------------------------------------------ #
OBJS            = knb.o

# ------------------------------------------------------------------------------------------ #
# MY DEFINE
# ------------------------------------------------------------------------------------------ #
MY_C_DEFINE         = -DTRACE
MY_PROC_DEFINE    =
MY_CFLAGS           = -I/Pharos/tmax52/pharostp/apm/usrinc
MY_LDPATH           = -L/Pharos/tmax52/pharostp/apm/lib
MY_LDLIBS           = -lpharostpadvice

# ------------------------------------------------------------------------------------------ #
# include common env file
# ------------------------------------------------------------------------------------------ #
OS_CFLAGS           = -std=c99 -m64 -Wparentheses -fPIC
OS_CDEFINE          = -D_LINUX64 -D_REENTRANT
```

```
OS_LDFLAGS          =
OS_OSLIBS           = -L/usr/lib64 -lcrypt -lpthread
OS_COMPILER         = gcc
OS_SO_FLAGS         = -shared -m64 -Wl,-soname,$(TARGET)

# ------------------------------------------------------------------------------------------ #
# LD Compiler Flag
# ------------------------------------------------------------------------------------------ #
CC                = $(OS_COMPILER)
CFLAGS            = $(MY_CFLAGS)    $(OS_CFLAGS) $(MY_C_DEFINE) $(OS_CDEFINE)
LD_PATH           = $(MY_LDPATH)    $(OS_LDPATH)
LD_LIBS           = $(MY_LDLIBS)    $(OS_LDLIBS)
LD_FLAGS          = $(MY_LDFLAGS) $(OS_CFLAGS)    $(LD_PATH)     $(LD_LIBS)

# ------------------------------------------------------------------------- #
# ORACLE suffix rule
# ------------------------------------------------------------------------- #
.SUFFIXES : .pc .c .o

.c.o:
        $(CC) $(CFLAGS) -c $<

# ------------------------------------------------------------------------------------------ #
# Dependency
# ------------------------------------------------------------------------------------------ #
all: $(TARGET)

$(TARGET): $(OBJS)
        $(CC) $(OS_SO_FLAGS) -o $@ $(OBJS) $(LD_FLAGS)

mv:
        mv $(TARGET) $(PHOMEDIR)/lib

clean:
        -rm -f core $(OBJS) $(TARGET)
```

G.6. Sample source

Advice에 대한 sample 소스이다.

```
/* ------------------------------------ include files ------------------------------------ */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <tpadvice.h>

/* ------------------------------------------------------------------------------------------ */
int
custom_service_prefix(const char *svcname, const char *data, int len)
{
	long rc = 0;
	char gid[64];
	char seqno[64];
	static long count = 0;

	memset(gid,	0x00, sizeof(gid));
	memset(seqno, 0x00, sizeof(seqno));
	getguid(gid);
	getselfseqno(seqno);

	printf("=========================================================₩n");
	printf("custom_service_prefix start₩n");
	printf("gid=[%s] seqno=[%s]₩n", gid, seqno);
	printf("=========================================================₩n");

	if ((strcmp(svcname, "CALLTREE4S") == 0) ||
		(strcmp(svcname, "CALLTREE2")  == 0)) {
		strcpy(seqno, "0");
		setselfseqno(seqno);
		setparmdata("KNB", "AAAAAAAAAAAAA");
	}
```

```
count++;
	if (strcmp(svcname, "CALLTREE4S") == 0) {
		if (count > 10) {
			printf("SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS₩n");
			count = 0;
			return -1;
		}
	}

	return 0;
}



/* ---------------------------------------------------------------------------------------- */
int
custom_service_postfix(const char *svcname, const char *data, int len)
{
	long rc = 0;
	char gid[64];
	char seqno[64];

	memset(gid,	0x00, sizeof(gid));
	memset(seqno, 0x00, sizeof(seqno));
	getguid(gid);
	getselfseqno(seqno);

	printf("=========================================================₩n");
	printf("custom_service_postfix start₩n");
	printf("gid=[%s] seqno=[%s]₩n", gid, seqno);
	printf("=========================================================₩n");

	if (strcmp(svcname, "CALLTREE4R") == 0)
		setapperror("AAA1234567", "");

	return 0;
}
```