Project	PMS
Team	시스템팀
Document Title	GS – HA 프로토콜
Document Class	

Revision	v 1.1	
Author	계 동원(truewise@neowiz.com)	
Date	2009년 1월 19일	
Revision History	v0.1	First Release
	v0.2	HealthyCheck,\ r\ n\ o 항목 추가
	v0.25	응답에러 코드 항목추가
	v0.26	통계정보 수정
	v1.0	**Structure base protocol 추가 (ADL)**
	v1.1	PC 방 통계 추가
	v1.2	PC 방 통계(Text Protocol) 추가 (3.6)

Notes :

Windows 용 HA 는 ADL 에서 PC 방 통계를 지원하고

Linux용 HA 는 Text 에서 PC 방 통계를 지원

궁금한 부분이 있으면 truewise@neowiz.com 으로 문의주세요

1 Introdution

1.1 Overview

제휴 업체 게임 서버의 효율적인 관리 및 모니터링을 위해서 네오위즈 운영 관리 시스템이 있다. 제휴 업체의 게임 서버는 네오위즈 운영 관리 시스템의 한 구성 요소인 Host Agent와의 연결을 통해 게임 서버 관리 및 모니터링에 필요한 정보를 전달한다. Host Agent는 게임 서버가 동작하는 컴퓨터에 항상 상주하는 서비스 프로그램이다.

이 문서는 Host Agent와 게임 서버간의 통신을 위한 텍스트 기반 프로토콜을 정리한다.

1.2 용어 정의

1.2.1 Host

프로그램이 정상 동작할 수 있는 환경이 구축된 컴퓨터

1.2.2 Game Server (이하 GS)

게임 서비스를 구축하는데 필요한 기능을 구현하고 있는 서버

1.2.3 Game Server Instance (이하 GSI)

Host에서 Game Server가 실행되었을 때, 생성되는 하나의 Process

Host에는 여러 개의 GSI의 생성이 가능하다.

1.2.4 Host Agent(이하 HA)

한 Host 내의 여러 GSI들을 관리/모니터링 할 수 있는 기능을 구현하고 있는 서버

1.2.5 Host Agent Instance (이하 HAI)

Host에서 실행 중인 상태의 HA. 하나의 Host에는 하나의 HAI만 존재한다.

1.3 기본 가정 및 기능

1.3.1 GS는 HA를 통해서만 실행될 수 있다.

1.3.2 HA는 GS의 실행에 필요한 정보를 Argument로 전달한다.

1.3.3 GS와 HA는 소켓으로 연결을 유지하고, 프로세스가 종료할 때까지 세션을 유지한다

1.3.4 GS와 HA간의 연결이 끊긴 경우, GS는 HA와의 연결이 복원될 때까지, 일정 시간마다 재접속을 시도한다.

1.3.5 HA가 GS에 대한 정보를 요청하면, GS는 내부 자료 구조에 저장한 정보를 HA에게 응답으로 전송한다.

2 Sequence

하나의 HAI가 GS를 실행하여, 새로운 GSI가 생성된 후, GSI와 HAI간에 일어나는 Interaction을 기술한다.

2.1 GS 실행

2.1.1 GS는 HA에 의해서 실행 된다.

```
// 서비스일 경우도 동일한 argument 를 갖는다.
C:>gameserver.exe
/GSID:GSID
/HAIP:HA IP
/HAPORT:HA PORT
/CNFGFILE:Game Server ConfigFile 이름
```

GSID : 프로세스 별로 만들어지는 고유의 ID, 하나의 호스트에 유일한 값이다.

HAIP : 동일 호스트의 HA가 Listen하는 IP (두개 이상의 IP가 존재할 경우를 위해 지정)

HAPORT : 동일 호스트의 HA가 Listen하는 Port

CNFGFILE : 게임서버가 갖는 게임정보 configuration file , full path를 지정한다. (ex: c:\ GameXX\ xxx.ini). 네트웍 드라이브의 경우나 원격 머신의 파일도 가능하다. 게임서버가 쓰는 configuration file이 없다면 넣지 않는다.

Configuration file 의 위치가 \ \ ConfSvr에 존재하는 경우 /CNFGFILE:\ \ ConfSvr\ dist\ heat\ Configuration.ini

ex :

gameserver.exe /GSID:16843008 /HAIP:211.169.214.82 /HAPORT:9856 /c:\ GameXX\ xxx.ini

Configuration File은 GS가 성능 모니터링에 필요한 정보, 즉 GS가 어떤 종류의 성능 데이터를 구해야 하는지에 대한 정보가 있다. 현재 이 내용이 정의 되지 않았으나 차후 제휴 사와 네오위즈가 협의하여 Define한다.

2.1.2 GSI는 자기 진단을 한 후 정상적으로 실행될 수 있는지 판단한다.

1) 메모리가 충분한지 설정 파일이 필요하다면 그 파일이 유효한지 진단하여 GS가 정상적으로 실행 가능한지 검사한다.

2.1.3 HAI로의 접속 요청

1) GS는 정상 실행 여부 판단 후, HAI로 접속을 시도한다.
2) 접속이 성공적으로 이루어지면, 정상 실행 여부를 HAI로 전송한다.
   가) 정상 실행이 불가능한 경우, 사유를 HAI로 전송한다.
   나) 정상 실행이 가능한 경우, GSI의 프로세스 기본 정보를 HAI로 전송한다.

2.1.4 성능 모니터링 관련 정보

1) HAI가 주기적으로 GSI로 Performance Request 메시지를 전송한다.
2) GSI는 Performance Request에 대한 응답으로 Performance Answer 메시지를 전송한다.

2.1.5 정상 동작 여부 확인

1) HAI는 주기적으로 HeartBeat Request 메시지를 전송한다.
2) GSI는 응답으로 HeartBeat Answer 메시지를 전송한다.

2.1.6 사용자 접속 통계 정보

1) HAI가 주기적으로 Statistic Request 메시지를 전송한다.
2) GSI는 응답으로 Statistic Answer 메시지를 전송한다.

3 Protocol

GS – HA 간의 데이터 부분의 프로토콜을 정의한다.

GS와 HA간 메시지는 TextBase이며 delimiter로는 '%'를 사용하고, 메시지 종료를 나타내는 \ r\ n\ 0을 사용한다.

3.1 Initialize

GSI스스로 정상적으로 실행될 수 있는 상태인지를 점검한다.

후에 HA에 접속하고 메시지를 전달한다.

이 메시지는 GSI가 HA에 접속 할 때마다 보내는 메시지이다.

3.1.1 Notify

1) 자기진단 성공

Prefix : INIT

Format : INIT+ delimiter + ProcGSID + delimiter + dwProcessId + delimiter + strSvrIP + delimiter + dwPort

ex : INIT%16843008%1234%211.169.224.93%8956\ r\ n\ 0

IP가 두개 이상일 때

ex : INIT2%16843008%1234%2%211.169.224.93/211.166.223.92%8956/9000\ r\ n\ 0

내용

```
GSID    ProcGSID;         // 게임 서버 프로세스마다 할당되는 GSI 고유의ID값. GS실행시 인자로전달
DWORD dwProcessId;// GetCurrentProcessId()로 알아내는 게임서버 프로세스 Id
string      strSvrIP ;  // 게임 클라이언트가 접속하는 IP
DWORD    dwPort;         // 게임 클라이언트가 접속하기를 기다리는 Port
```

2) 자기진단 실패

Prefix : INFA

Format : INFA+ delimiter + ProcGSID + delimiter + dwErrCode

ex : INFA%16843008%1234\ r\ n\ 0

3.2 HeartBeat

일정 시간 간격으로 GSI의 동작 여부 점검한다

HA에서 GSI로 Request 패킷을 보낸다.

응답으로 GS에서 HA로 Answer 패킷을 보낸다.

1부터 1씩 증가하는 10진수의 Sequence No를 Prefix 다음에 붙여서 Sequence를 표시한다.

Sequence No는 65535를 MAX로 하여 MAX에 도달하면 0으로 초기화한다.

3.2.1 Request

Prefix : HBREQ

Format : HBREQ + delimiter + HeartBeat No.

ex : HBREQ%1\ r\ n\ 0

3.2.2 Answer

Request 시에 받은 Sequence No를 그대로 넣는다.

Prefix: HBANS

Format:HBANS+ delimiter + ProcGSID + delimiter + HeartBeatNo

ex : HBANS%1\ r\ n\ 0

3.2.3 Err Answer

적용되는 서버가 아닐경우에 이 메시지를 보낸다.

ERR_DONT_HB%ProcGSID

ex : ERR_DONT_HB%1234\ r\ n\ 0

3.3 Performance

GSI의 상태 모니터링 정보를 나타내는 값을 전달한다.

GS의 특성에 따라 측정치가 틀려 질 수 있다.

HA가 GSI로 Request를 하면 GSI는 HA로 Answer를 전달한다.

필요이상의 데이터는 0으로 설정한다.

이 내용은 GS마다 다른 내용이므로 제휴사는 GS성능 측정이 어떤 내용이며 측정치 값의 throughput에 대한 MAX 를 정해 네오위즈와 협의한다.

3.3.1 Request

Prefix:PERE

Format:PERE

ex : PERE\ r\ n\ 0

3.3.2 Answer

현재 GSI의 성능 정보를 순서대로 delimiter로 구분한다.

```
DWORD dwCount1;        // GS 성능모니터링 값
DWORD dwCount2;        // GS 성능모니터링 값
DWORD dwCount3;        // GS 성능모니터링 값
DWORD dwCount4;        // GS 성능모니터링 값
DWORD dwCount5;        // GS 성능모니터링 값
DWORD dwCount6;        // GS 성능모니터링 값
DWORD dwCount7;        // GS 성능모니터링 값
DWORD dwCount8;        // GS 성능모니터링 값
DWORD dwCount9;        // GS 성능모니터링 값
```

```
    DWORD dwCount10;     // GS 성능모니터링 값
```

Prefix : PEAN

Format:PEAN+ delimiter + ProcGSID + delimiter + count1 + delimiter + count2 + delimiter + count3… + delimiter + count10

ex : XX게임 제휴사의 GS는 성능 모니터링 리스트로 2가지가 있다

NetQueue : 네트웍 이벤트의 처리 Delay를 알기 위한값

DBQueue : DBGW를 사용하는 GS라면 DB 호출 명령의 Pending 현상을 측정하기 위한 값

이 내용은 네오위즈와 협의하여 결정된 값이다.

PEAN%1%10%0%0%0%0%0%0%0%0\ r\ n\ 0

3.3.3 Err Answer

적용되는 서버가 아닐경우에 이 메시지를 보낸다.

ERR_DONT_PF%ProcGSID

ex : ERR_DONT_PF%1234\ r\ n\ 0

3.4 Statistic

서버에 접속한 사용자의 통계 정보를 알아온다

HA에서 GSI로 Request 패킷을 보낸다.

응답으로 GSI에서 HA로 Answer 패킷을 보낸다.

GSI와 HA 가 같은 GSSTATISTICINFO 구조체를 가지고 nUserArray 배열의 각각의 멤버를 서로 동기화한다.

```
struct GSSTATISTICINFO
{
    int nCurrentUser;                    // 현재 사용자 수
    int nArrUserInfo[nMaxRegion]        // 지역코드
                    [nMaxAge]           // 연령코드
                    [nMaxGender   ];  // 성별코드
};
```

nCurrentUser : 현재 게임 서버에 접속한 사용자 수

nArrUserInfo : 게임 사용자에 대한 통계 nMaxRegion, nMaxAge, nGender 크기의 3차원배열을 생성한다.

게임서버와 배열 구조가 같아야 하며, 코드가 미리 약속되어 있어야 한다.

GSSTATISTICINFO 내의 모든 숫자는 10진수로 표시한다.

지역코드 : 서울, 경기 등의 지역을 나타낸다. nMaxRegion = 17

범위 : nArrUserInfo[0][][] – nArrUserInfo[16][][]

nArrUserInfo[0][][] : 경기

nArrUserInfo[1][][] : 서울

nArrUserInfo[2][][] : 인천

nArrUserInfo[3][][] : 강원

nArrUserInfo[4][][] : 충남

nArrUserInfo[5][][] : 충북

nArrUserInfo[6][][] : 대전

nArrUserInfo[7][][] : 경북

nArrUserInfo[8][][] : 경남

nArrUserInfo[9][][] : 대구

nArrUserInfo[10][][] : 부산

nArrUserInfo[11][][] : 울산

nArrUserInfo[12][][] : 전북

nArrUserInfo[13][][] : 전남

nArrUserInfo[14][][] : 광주

nArrUserInfo[15][][] : 제주

nArrUserInfo[16][][] : 해외

연령코드 : 사용자 나이대를 구분한다. nMaxAge = 7

nArrUserInfo[][0][] : 0세 ~ 13세

nArrUserInfo[][1][] : 14세 ~ 16세

nArrUserInfo[][2][] : 17세 ~ 19세

nArrUserInfo[][3][] : 20세 ~ 24세

nArrUserInfo[][4][] : 25세 ~ 29세

nArrUserInfo[][5][] : 30세 ~ 39세

nArrUserInfo[][6][] : 40세 ~ 99세

성별코드 : 사용자의 성별을 구분한다. nMaxGender = 2

nArrUserInfo[][][0] : 남자

nArrUserInfo[][][1] : 여자

3.4.1 Request

Prefix : STRE

현재 사용자 수 요청시 Format : STRE

ex : STRE\ r\ n\ 0

3.4.2 Answer

Prefix : STAN

사용자 통계정보 응답시 Format : STAN + delimiter + ProcGSID + delimiter + TotUser + delimiter + nArrUserInfo[0][0][0] + delimiter + ...

+ delimiter + nArrUserInfo[nMaxRegion-1][nMaxAge-1][nMaxGender-1]

총 nMaxRegion * nMaxAge * nMaxGender 개 필드의 값을 다음 순서로 보낸다.

ex : STAN%GSID%TOTUSER%20%30%10%20%30%.....\ r\ n\ 0

3.4.3 Err Answer

적용되는 서버가 아닐경우에 이 메시지를 보낸다.

ERR_DONT_ST%ProcGSID

ex : ERR_DONT_ST%1234\ r\ n\ 0

3.5 Announce

HA->GS로 공지를 전달한다.

Notification 패킷으로 응답을 받지 않는다.

3.5.1 Request

Prefix : ANNO

Format : ANNO + delimiter + 공지할 내용%YYYY-MM-DD-HH-MM-SS

ex : ANNO%정기 점검 시간입니다.%2003-07-24-09-04-12\ r\ n\ 0

3.5.2 Err Answer

적용되는 서버가 아닐경우에 이 메시지를 보낸다.

ERR_DONT_PF%ProcGSID

ex : ERR_DONT_AN%1234\ r\ n\ 0

3.6 Statistic for PC방

3.4 의 Statistic 항목과 동일하게 사용. 단, Prefix 만 다르다.

PC방 통계용 Request 메시지가 별도로 존재하는 것이 아니고,

통계 요청 메시지(STRE)를 받았을 때, PC방 통계가 존재할 경우 STPCAN 으로 응답하면 된다.

3.6.1 Request

Prefix : STRE

현재 사용자 수 요청시 Format : STRE

ex : STRE\ r\ n\ 0

3.6.2 Answer

Prefix : STPCAN

사용자 통계정보 응답시 Format : STPCAN + delimiter + ProcGSID + delimiter + TotUser + delimiter + nArrUserInfo[0][0][0] + delimiter + ...

+ delimiter + nArrUserInfo[nMaxRegion-1][nMaxAge-1][nMaxGender-1]

총 nMaxRegion * nMaxAge * nMaxGender 개 필드의 값을 다음 순서로 보낸다.

ex : STPCAN%GSID%TOTUSER%20%30%10%20%30%.....\ r\ n\ 0

4 Structured Protocol (ADL 사용) (v 1.0)

4.1 ADL Manager

4.1.1 PMS Function/Coding spec. 참고

4.1.2 PMSGS Common structure

- ✓ PMS(특히 HA)에서 사용하는 모든 메시지 및 data structures는 구분을 위해 PMS-로 표기한다.
- ✓ 추가되는 GS와의 메시지 부분은 기존의 PMS- 메시지와 충돌을 피하기 위해 PMSA-로 표기한다.

```
Class PMSUnitID
{
    DWORD dwGSID;
    DWORD dwSSN;
    DWORD dwCategory
}
```

- ✓ PMS에서 unique함을 보장하는 instance 단위.
- ✓ dwGSID : 서버(기계)당 하나씩 부여되는 구분자, 지정된 ID를 할당 받음.
- ✓ dwSSN, dwCategory : 서비스 구분(예:고스톱, 포커)과 하위 상세 구분(초보, 중수, ..)

```
class PMSNSAP
{
    DWORD dwIP;
    LONG lPort;
};
#define NSAPVecT ArcVectorT<PMSNSAP>
```

- ✓ Network address
- ✓ GS의 ADL과 연관성을 갖지 않기 위해 재 정의.

```
class PMSFaultInfo
{
    DWORD dwGSID;
    LONG lGSState;
    LONG lFaultID;
    LONG lLevel;
    LONG lReason;
    string sHAID;
    string sReason;
    string sDesc;
```

```
    string sProc;
    string sTime;
    string sSSN;
};
```

✓ GS의 Fault 정보를 HA에서 처리하기 위해 필요한 모든 정보

✓ Text base와 ADL base protocol의 정보를 모두 포함.

✓ lFaultID : GS의 fault notify를 DB에 등록하고, 결과로 return 되는 fault id.

```
#define vecPerformT ArcVectorT<LONG>
class PMSPerformInfo
{
    LONG lTimeOutCount;
    string sPerfName;
    DWORD dwGSID;

    DWORD dwCPU;
    DWORD dwMEM;
    vecPerformT vecPerform;
};
```

✓ GS의 performance 정보를 HA에서 처리하기 위해 필요한 모든 정보

```
class PMSStatBase
{
    PMSUnitID unitID;
    DWORD dwCU;
    DWORD dwSession;
    DWORD dwChannelCnt;
    DWORD dwRoomCnt;
    string sOptionInfo;
};
#define VecStatBaseT ArcVectorT<PMSStatBase>
```

✓ GS와 HA간의 statistics 정보를 전달하는데 필요한 정보

```
class PMSStatInfo
{
    LONG lTimeOutCount;
    PMSUnitID unitID;
    DWORD dwCU;
    DWORD dwSession;
    string sUpTime;
```

```
    DWORD dwChannelCnt;
    DWORD dwRoomCnt;
    string sOptionInfo;
};
```

✓ HA 내부에서 statistics 정보를 처리하는데 필요한 정보

```
#define vecRegionInfoT ArcVectorT<LONG>
class PMSRegionInfo
{
    PMSUnitID unitID;
    vecRegionInfoT vecRegion;
};
#define VecRegionInfo ArcVectorT<PMSRegionInfo>
```

✓ 지역별 분포 정보를 전달.

✓ 기존의 AMS에서 사용하던 형식을 그대로 사용

```
class PMSStartupInfo
{
    DWORD dwGSID;
    DWORD dwProcID;
    NSAPVecT vecNSAP;
    DWORD dwErrCode;
    string sStartTime;
    string sOptAddr;
};
```

4.1.3 HA -> GS message

```
#define ADL_VERSION        0x00010001
```

✓ GS와의 protocol version을 확인하기 위한 정보

✓ Header file에 pre defined 값을 사용하여, HA가 check.

```
class PMSAPerformReq
{
};
```

✓ Performance 정보 요청

✓ 자세한 내용은 아래의 answer message를 참고

```
class PMSAHeartBeatReq
{
```

```
    LONG lIndexNo;
};
```

✓ 아래의 answer message 참고

```
class PMSAStatInfoReq
{
};
```

✓ 아래의 answer message 참고

```
class PMSAStatInfoPCReq
{
};
```

✓ 아래의 answer message 참고

```
class PMSARegionInfoReq
{
};
```

✓ 아래의 answer message 참고

```
class PMSARegionInfoPCReq
{
};
```

✓ 아래의 answer message 참고

```
class PMSAAnnounceReq
{
    PMSUnitID pmsUnitID;
    string sMessage;
};
```

✓ 공지 메시지 전달

✓ pmsUnitID : 필요한 범위(target range)까지 값을 셋팅하여 사용.

```
class PMSAOrderReq
{
    DWORD dwMCID;
    string sOrderName;
    PMSUnitID pmsUnitID;
    LONG lTargetRange;
    string sOrderVal;
    string sReqTime;
};
```

✓ PMS를 통해 임의의 명령을 수행하도록 GS에 메시지 전달.

✓ 가능한 모든 종류의 명령을 전달가능하며, GS와 MT간의 약속을 통해 의미를 가진다.

- ✓ dwMCID : 명령 주체, 히스토리 관리, 응답 등을 위해 전달
- ✓ sOrderName : 명령 구분, 확장성을 위해 string으로 정의.
- ✓ pmsUnitID : 명령을 수행할 target 구분
- ✓ lTargetRange : 명령을 전달할 범위 code, 구현의 편의를 위해 사용
- ✓ sOrderVal : 명령의 수행에 필요한 값이 있을 경우 전달.
- ✓ sReqTime : 명령을 요청한 시간.

4.1.4 GS -> HA message

```
class PMSAGSInitNtf
{
    LONG lVersion;
    DWORD dwGSID;
    DWORD dwProcID;
    NSAPVecT vecNSAP;
    DWORD dwErrCode;
    string sOptAddr;
};
```

- ✓ HA로부터 기동된 GS가 정상적으로 시작 되었음을 알림.
- ✓ Version을 확인, DB에 미리 설정된 GS인지 판단하여 이상이 없을 경우 등록.

```
class PMSAInitFaultNtf
{
    PMSUnitID unitID;
    LONG lErrCode;
};
```

- ✓ HA로부터 기동된 GS가 정상적으로 기동되지 못할 경우를 알림

```
class PMSAPerformAns
{
    vector<LONG> vecPerform;
};
```

- ✓ GS가 스스로 파악할 수 있는 성능 관련 정보를 전달할 경우 사용.
- ✓ 기본적인 성능정보(CPU, MEM..)는 HA가 직접 얻으며, 위의 정보는 GS내부에서 생성하는 정보임.
- ✓ 어떤 성능정보를 담을지는 미리 정의되지 않았음.
- ✓ 사용하지 않을 경우 빈 컨테이너를 전송.

```
class PMSAHeartBeatAns
{
    LONG lIndexNo;
```

```
    DWORD dwGSID;
};
```

✓ GS-HA간의 heartbeat

✓ Index No : 필요할 경우 사용.

```
class PMSAStatInfoAns
{
    VecStatBaseT vecStatBase;
};

class PMSAStatInfoPCAns
{
    VecStatBaseT vecStatBase;
};
```

✓ GS의 서비스적인 통계 정보, 즉 게임별 cu, 사용자 분포 등을 전달.

✓ PC가 붙은 것은 PC방 통계용. (구조는 같음)

✓ 위의 PMSStatBase 참조

```
class PMSARegionInfoAns
{
    VecRegionInfo vecRegionInfo;
};

class PMSARegionInfoPCAns
{
    VecRegionInfo vecRegionInfo;
};
```

✓ GS에 접속한 사용자의 지역 분포를 전달.

✓ 정의된 순수에 따라 [SEX][SSN][REGION]의 형태.

```
class PMSAAnnounceAns
{
    DWORD dwGSID;
    LONG lErrCode;
};
```

✓ 공지 메시지 전달 요청에 대한 응답.

```
class PMSAOrderAns
{
    string sOrderName;
    PMSUnitID pmsUnitID;
```

```
    DWORD dwMCID;
    LONG lTargetRange;
    string sErrorMsg;
    string sReqTime;
    string sOrderVal;
};
```

✓ 명령 전달 결과를 알림.

✓ Async로 처리되는 메시지의 특성상 때문에 요청 시 전달 받은 메시지를 그대로 되돌려 줌.

✓ 위의 request message 참조.

```
class PMSAWarningNtf
{
    PMSUnitID unitID;
    LONG lErrLevel;
    string sWarnMsg;
    string sTreatMsg;
};
```

✓ GS가 내부적으로 판단하여 MC에 알리고자 하는 사항이 있을 경우 사용.

✓ 장애 여부는 HA에서 판단 가능하지만, 특화된 이상 증상의 판단은 GS만이 가능.

✓ 이를 처리하기 위한 메시지.

✓ lErrLevel : critical or alert 등 구분.

✓ sWarnMsg : 경고의 내용.

✓ sTreatMsg : 경고에 대한 대처 방법.

✓ GS에 특화된 정보이기 때문에, 따로 코드화하지 않고 GS에서 원하는 형태의 메시지를 그대로 전달 하기 위해 string 인자 사용.