System Service

API specification

Note/
CWORDXX is mask word for confidential information.

2019.10.17
TOYOTA MOTOR CORPRATION

Module Documentation

BaseSystem

System_service

Detailed Description

System_service

Interface_unified
Task_manager
Rom_access_library
Resource_manager
Config
System_manager
Logger_service
Power_service
Version_library

Detailed Description

Interface_unified

struct ARTIFACT_RESPONSE
struct THeartBeatSession
class CHeartBeatServiceIf
struct _TVINnumber
VIN numbers. struct _T_CWORD33_CANMileageInfo
CAN Diagnostic status data. More...
struct __CWORD62_DateAndTime
CAN current date and time. More...
struct _T_CWORD33_ScreenCaptureEvt
Screen Capture Event data. struct _SStatisticalCounter
Statistical counter. More...
struct _SEventLoggerCommonInfo
Event logger common information. More...

struct _SEventLoggerResetInfo

Event logger resets counter information. *More...*

struct _SEventLoggerEventInfo

Event logger CWORD56 events information. *More...*

struct _SEventCANLoggerEventInfo

Event logger CAN events information. *More...*

struct _PwCompleteAck
struct _SS_PSCurrentState
class Process
class TimerCtrl

***OSprocess*. Macros**

#define **SERVICE__CWORD69_** "_CWORD69_"
#define **SERVICE_AS_AUDIO** "AS_AudioService"
#define **SERVICE_AS_MODE** "AS_ModeService"
#define **SERVICE_BR_BROWSER** "BR_BrowserService"
#define **SERVICE_CAN_SHADOW** "VS_CANShadow"
#define **SERVICE__CWORD51__BT** "_CWORD51__BT"
#define **SERVICE__CWORD51__DR** "_CWORD51__DR"
#define **SERVICE__CWORD51___CWORD105_** "_CWORD51___CWORD105_"
#define **SERVICE_DAB** "RS_Master_Tuner"
#define **SERVICE_HMI** "HMI"
#define **SERVICE_HMI__CWORD46_** "HMI__CWORD46_Service"
#define **SERVICE_HMI__CWORD77_** "HMI__CWORD77_Service"
#define **SERVICE_IPC_MP_SHADOW** "PS_IPC_MP_Shadow"
#define **SERVICE_KEY_HANDLER** "PS_KeyHandler"
#define **SERVICE_MM_ICD** "MM_ICDService"
#define **SERVICE_MM_MEDIA** "MM_MediaService"
#define **SERVICE_MM__CWORD21_** "MM__CWORD21_Service"
#define **SERVICE_NAV** "NAV_NavigationService"
#define **SERVICE_NAV_LOC** "NAV_LocationService"
#define **SERVICE_NS_NPP** "NS_NPPService"
#define **SERVICE_NS_SHARED_MEM** "NS_SharedMem"
#define **SERVICE_NW_BT** "NW_BluetoothService"
#define **SERVICE_NW_CONNECTION** "NW_ConnectionService"
#define **SERVICE_NW_MESSAGING** "NW_MessagingService"
#define **SERVICE_NW_PHONE** "NW_PhoneService"
#define **SERVICE_NW_PHONE_BOOK** "NW_PhoneBookService"
#define **SERVICE_PSM_SHADOW** "PS_PSMShadow"
#define **SERVICE_PS_IPC** "PS_IPC"
#define **SERVICE_PS_SWDL_SHADOW** "PS_SoftwareUpdateShadow"
#define **SERVICE_SDARS** "RS_XSDARSService"
#define **SERVICE_TUNER** "RS_Radio"
#define **SERVICE_VS__CWORD4_** "VS__CWORD4_Service"
#define **SERVICE_VS_DIAG** "VS_DiagService"
#define **SERVICE_VS_DIAG_DISP** "DiagDispatcherService"
#define **SERVICE_VS__CWORD105_** "VS__CWORD105_Service"

#define **SERVICE_VS_RVC** "VS_RVCService"
#define **SERVICE_VS_VEHICLE** "VS_VehicleService"
#define **SS_ERROR_EVENT_START_REQ_TO_SEC** (5)
#define **SS_ERROR_EVENT_DEBUG_DUMP_RSPN_TO_SEC** (1)
#define **SS_ERROR_EVENT_BOOT_MICRO_LOG_RESPONSE_TO_SEC** (2)
#define **SS_ERROR_EVENT_CORE_FILE_POLL_TO_MS** (100)
#define **SS_ERROR_MSG_USB_DISCON** "UsbDisconnected"
#define **SS_ERROR_MSG__CWORD84__ABNORMAL_SHUTDOWN** "_CWORD84_AbnormalShutdown"
#define **MAX_STATISTICAL_BUFFER** 240
#define **SS_LOGGER_SAVE__CWORD33_LOG_FLAG** "/ramd/log/_CWORD33_log/SS_LOGGER_SAVE__CWORD33_LOG_FLAG"
#define **SS_DEV_DETECT_THD** "SS_Dev_Detect_Thd"
#define **SS_DEV_DETECT_SRV** MN_SS_DEVDETECTSRV
#define **SS_DEV_DETECT_BSP_THD** "SS_DeviceDetectErr"
#define **SS_GROUP_LAUNCH_TRIGGER** "SS_GroupLaunchTrigger"
#define **SERVICE_HEARTBEAT** "SS_HeartBeatService"
#define **SERVICE_LOGGER** MN_SS_LOGGERSRV
#define **SS_UDEV_DEV_DETECT_DLL** "SS_UDEV_DeviceDetect"
#define **SS_PLM_SERVICE** "SS_PLMService"
#define **SERVICE_POWER** MN_SS_POWERSERVICE
#define **SS_RESOURCE_MONITOR_SERVICE** "SS_ResourceMonitorService"
#define **SERVICE_SOFTWAREUPDATE** "SS_SofUpdateSrv"
#define **SERVICE_SYSMANAGER** MN_SS_SYSMANAGER
#define **SS_WINSYS** MN_SS_WINSYS
#define **SS_RESOURCE_MANAGER** MN_SS_RESOURCEMGR
#define **SS_TASK_MANAGER** MN_SS_TASKMANAGER
#define **SERVICE_VUP** MN_VUPSERVICE
#define **SS_UPDATESERVICE** MN_SS_UPDATESERVICE
#define SSDEBUGDUMP(args...) (SendDebugDumpResponseToSystemManager(TRUE, ## args))

Redirect either formatted, or unformatted response output to System Manager.

#define **SS_SMHeartbeat** "SM.Heartbeat"
#define **SS_SMLauncher** "SM.ProcLaunch"
#define **SS_SMLowMemMonitor** "SM.LowMemMon"
#define **_CWORD33__MAKE_DEFAULT_CALLBACK**(x)
#define NTFY_SSSystemMgrUserMode SERVICE_SYSMANAGER"/UserMode"
#define NTFY_SSSystemMgrDataReset SERVICE_SYSMANAGER"/DataReset"

Notification for data reset.

#define NTFY_SSLastSourceType SERVICE_SYSMANAGER"/LastSourceType"

Notification used by Audio and System Services for Dynamic Launch.

Typedefs

typedef enum _SS_Sources **SS_SOURCE**
typedef enum _SS_Sources * **PSS_SOURCE**
typedef enum _SS_Zones **SS_ZONE**

typedef enum _SS_Zones * **PSS_ZONE**
typedef enum E_AUDIO_PORT **SS_AUDIO_INPUT_PORT**
typedef enum E_AUDIO_PORT * **PSS_AUDIO_INPUT_PORT**
typedef enum _SS_Device_Context **SS_DEVICE_CONTEXT**
typedef enum _SS_Device_Context * **PSS_DEVICE_CONTEXT**
typedef std::string **SS_CONTENT_NAME_t**
typedef std::map< SS_CONT_CATEGORY_t, SS_CONTENT_NAME_t > **SS_LAST_INFO_t**
typedef struct __CWORD62_DateAndTime **STCanCurrentDateTime**
typedef enum _ELoggerErrorCodes **ELOGGERERR_CODES**
typedef enum _EEventLoggerSuccessCode **EEvtLoggerSuccessCode**
typedef enum _PwLVIStatus **PowerSrvLVIStatus**
typedef enum _PwLVIStatus **EPWER_LVI_STATUS**
typedef struct _PwCompleteAck **StartCompleteAck**
typedef struct _SS_PSCurrentState **SS_PSCurrentState**
typedef enum _SS_PowerServiceProtocol SS_PowerServiceProtocol
typedef std::map< SS_String, Process * > **ProcessMap**

Enumerations

enum _SS_Sources { **SS_SOURCE_NA** = ***, **SS_SOURCE_FM** = ***, **SS_SOURCE_AM** = ***,
SS_SOURCE_CD = ***, **SS_SOURCE_AUX** = ***, **SS_SOURCE_PHONE** = ***, **SS_SOURCE_TA** = ***, **SS_SOURCE_CHIMES** = ***, **SS_SOURCE_NAVI** = ***, SS_SOURCE_MSC = ***,
SS_SOURCE_MTP = ***, **SS_SOURCE__CWORD57_** = ***, SS_SOURCE__CWORD73__APP = ***,
SS_SOURCE__CWORD73__VIDEO_MODE = ***, SS_SOURCE_STREAMING = ***,
SS_SOURCE_BT_AUDIO = ***, SS_SOURCE_BT_SPP = ***, SS_SOURCE_SPEECH = ***,
SS_SOURCE_USB = SS_SOURCE_MSC, **SS_SOURCE__CWORD59_** = ***, **SS_SOURCE_ADRM** = ***, **SS_SOURCE__CWORD60_** = ***, **SS_SOURCE_ADRA** = ***,
SS_SOURCE_INCOMING_CALL = SS_SOURCE_PHONE, **SS_SOURCE_PRIVATE_MODE_CALL** = ***, **SS_SOURCE_HANDSFREE_SPEAKING** = ***, **SS_SOURCE_FM_DAB** = ***,
SS_SOURCE_SD_CARD = ***, **SS_SOURCE_SDARS** = ***, **SS_SOURCE_BROWSER_ENT** = ***,
SS_SOURCE_ML_ENT = ***, **SS_SOURCE_ML_INFO** = ***, **SS_SOURCE_DATA_CD** = ***,
SS_SOURCE_TAS = ***, **SS_SOURCE_DTV** = ***, **SS_SOURCE_DVD** = ***, **SS_SOURCE_BD** = ***, **SS_SOURCE_DELIVERY_APP** = ***, **SS_SOURCE_ENTUNE_ENT** = ***,
SS_SOURCE_HDD_AUDIO = ***, **SS_SOURCE_JOYFUL_TALK** = ***,
SS_SOURCE_CLEARANCE_SONAR = ***, **SS_SOURCE_DSRC** = ***,
SS_SOURCE_ENTUNE_INFO = ***, **SS_SOURCE_HELPNET** = ***, **SS_SOURCE_OTV** = ***,
SS_SOURCE_MIRACAST = ***, **SS_SOURCE_RSE** = ***, **SS_SOURCE_INSIDER** = ***,
SS_SOURCE_MAIL = ***, **SS_SOURCE_MAIL_RINGTONE** = ***, **SS_SOURCE_BD_REAR** = ***,
SS_SOURCE_HDMI_REAR = ***, **SS_SOURCE_SD_CARD_REAR** = ***,
SS_SOURCE_MIRACAST_REAR = ***, **SS_SOURCE_USB_REAR** = ***,
SS_SOURCE_DLNA_REAR = ***, **SS_SOURCE_BROWSER_INFO** = ***,
SS_SOURCE_DSRC_ENT = ***, **SS_SOURCE_RECDATA_PLAY** = ***, **SS_SOURCE_AMFM** = SS_SOURCE_FM, **SS_SOURCE_DLNA** = ***, **SS_SOURCE_USB2** = ***, **SS_SOURCE_VDSP** = ***,
SS_SOURCE_NAVI_VR = ***, **SS_SOURCE_STARTUPBGM** = ***,
SS_SOURCE_STARTUPBGM_INIT = ***, **SS_SOURCE_MAIL_INFO** = ***, **SS_SOURCE_DIAG** = ***, **SS_SOURCE_MAYDAY** = ***, **SS_SOURCE_DCM** = ***, **SS_SOURCE_SD_VIDEO** = ***,
SS_SOURCE_DELIVERY_APP_INFO = ***, **SS_SOURCE__CWORD27__INFO** = ***,
SS_SOURCE_POWER_OFF = *** }

enum _SS_Zones { **SS_ZONE_NA** = ***, SS_ZONE_DRIVER = ***, **SS_ZONE_REAR1**, **SS_ZONE_REAR2**, **SS_ZONE_REAR3**, **SS_ZONE_ALL** }

enum E_AUDIO_PORT { SS_AUDIO_INPUT_PORT_1 = ***, SS_AUDIO_INPUT_PORT_2 = ***, SS_AUDIO_INPUT_PORT_3 = ***, SS_AUDIO_INPUT_PORT_4 = ***, SS_AUDIO_INPUT_PORT_5 = *** }

enum **_SS_Device_Context** { **SINGLE_CONTEXT** = ***, **MULTI_CONTEXT** = *** }

enum **EErrorEventType** { **eErrorEventTypeProcessCrash** = ***, **eErrorEventTypeHeartBeatFailure**, **eErrorEventTypeSystemLowMemory**, **eErrorEventTypeReserved1**, **eErrorEventTypeUserInvokedUserForceReset**, **eErrorEventTypeReserved2**, **eErrorEventTypeReserved3**, **eErrorEventTypeUserInvokedCollectAllLogs**, **eErrorEventTypeBootMicroReset**, **eErrorEventTypeProcessExit**, **eErrorEventTypeUserInvokedCollectScreenShot**, **eErrorEventTypeUserInvokedCollect_CWORD33_Logs**, **eErrorEventTypeEelExport**, **eErrorEventType_CWORD33_EmmcLogs**, **eErrorEventTypeDiagEvent**, **eErrorEventTypeCanEvent**, **eErrorEventTypeDtcEvent**, **eErrorEventTypeModConnFailed**, **eErrorEventTypeStartRespFailed**, **eErrorEventTypeUserInvokedCollectDevLogs**, **eErrorEventTypeModuleInvokedResetRequest**, **eErrorEventTypeModuleInvokedCollectDebugLogs**, **eErrorEventTypeUserInvokedClearLogs**, **eErrorEventTypeUserInvokedCollectNaviLog**, **eErrorEventTypeGroupRelaunch**, **eErrorEventTypeMaxValue** }

enum **EArtifactId** { **eArtifactId_CWORD33_DebugLog** = ***, **eArtifactIdTransmitLog**, **eArtifactIdPerformanceLog**, **eArtifactIdBootMicroLog**, **eArtifactIdSystemDataCsv**, **eArtifactIdShowMemTxt**, **eArtifactIdProcessCore**, **eArtifactIdDebugDumpLog**, **eArtifactIdKernelLog**, **eArtifactIdDRInitialLog**, **eArtifactIdDRLocationLog**, **eArtifactIdCpuHighLoadMonteCarloLogs**, **eArtifactIdMetaCoreLogs**, **eArtifactIdCmsLogs**, **eArtifactIDInternalDTC**, **eArtifactIdComDebugLog**, **eArtifactIdRadioDebugLog**, **eArtifactIdConnectivityDebugLog**, **eArtifactIdKernelBootLog**, **eArtifactIdGraphicsDebugLog**, **eArtifactIdSysLog**, **eArtifactIdPstoreLog**, **eArtifactIdClearAllLog**, **eArtifactIdNaviLog**, **eArtifactIdWinSysLog**, **eArtifactIdAppFwLog**, **eArtifactIdTomoyoLog**, **eArtifactIdScreenShot**, **eArtifactIdDebugFolder2Content**, **eArtifactIdDebugFolderContent**, **eArtifactIdMaxValue** }

enum **EErrorEventPrio** { **eErrorEventPrioDiagLogRequest** = ***, **eErrorEventPrioUserInvokedCollectAllLogs** = ***, **eErrorEventPrioUserInvokedCollectScreenShot** = ***, **eErrorEventPrioUserInvokedCollect_CWORD33_Logs** = ***, **eErrorEventPrioModuleInvokedCollectDebugLogs** = ***, **eErrorEventPrioUserInvokedUserForceReset** = ***, **eErrorEventPrioEelStorageFilePresent** = ***, **eErrorEventPrioEmmcLogsFilePresent** = ***, **eErrorEventPrioDefault** = *** }

enum **SS_CONT_CATEGORY_t** { **SS_CCAT_F_VIDEO**, **SS_CCAT_F_SUB_VIDEO**, **SS_CCAT_F_AUDIO**, **SS_CCAT_R_VIDEO**, **SS_CCAT_R_AUDIO**, **SS_CCAT_MAX** }

enum **_ELoggerErrorCodes** { **eSelectedDeviceNotFound** = ***, **eWriteToDeviceFailed** = ***, **eScreenCaptureFailed** = ***, **eScreenCaptureSaveTimeExpired** = ***, **eScreenCaptureStoreTimeExpired** = ***, **eNoLogToStore** = *** }

enum **_EEventLoggerSuccessCode** { **COPY_EVT_USB_SUCCESS** = ***, **CLEAR_EVENT_LOG_SUCCESS** = ***, **STATISTICAL_COUNTER_READ_SUCCESS** = *** }

enum eSSLoggerCANProtocolID { **eSSLoggerCANProtocolIDCANTrigger** = ***, eSSLoggerCANProtocolIDDTCTrigger }*eSSLoggerCANProtocolID*

enum eSSLoggerCANEvent { **eSSLoggerCANEventStart** = ***, eSSLoggerCANEventError, eSSLoggerCANEventFinished }*eSSLoggerCANEvent*

enum **SS_STORELOGS_OPE_TYPE** { **SS_STORELOGS__CWORD33_LOG** = ***, **SS_STORELOGS_ILLEGAL**, **SS_STORELOGS_ACCOFFON**, **SS_STORELOGS_SYS_ILLEGAL**, **SS_STORELOGS_ACCOFFON_PRESS**, **SS_STORELOGS_MAX** }

enum **_PwLVIStatus** { **ePwSrvLVI_Status_InActive**, **ePwSrvLVI_Status_Active** }

enum _SS_PowerServiceProtocol { SS_POWER_PRINT_CONNECTIONS = ***, SS_POWER_PRINT_STACK = ***, SS_POWER_WAKEUP_COMPLETE = ***, SS_POWER_SHUTDOWN_COMPLETE = ***, SS_POWER_VOLTAGE_STATE = ***, SS_POWER_LVI2_SHUTDOWN_COMPLETE = ***, SS_POWER_CRANK_STATE = ***, SS_POWER_WAKEUP_MODULES_CMPL_RSPN = ***, SS_POWER_SYSTEM_LAUNCH_COMPLETE = ***, SS_POWER_STATE_CHANGE_REQ = ***, SS_POWER_STATE_CHANGE_RESP = ***, SS_POWER_COMM_WAKEUP = ***, SS_POWER_SHUTDOWN_REQ = ***, SS_POWER_SHUTDOWN_RESP = ***, SS_POWER_CRNT_STATE_QUERY = ***, SS_POWER_CRNT_STATE_QUERY_RSPN = ***, SS_POWER_SYSTEM_MODE_INFO_REQ = ***, SS_POWER_SYSTEM_MODE_INFO_RESP = ***, SS_POWER_INITCOMP_REP = ***, SS_POWER_PUBLISH_POWER_POPUP_REQ, SS_POWER_PUBLISH_POWER_POPUP_RESP, SS_POWER_PUBLISH_SHUTDOWN_CONDITION_REQ, SS_POWER_PUBLISH_SHUTDOWN_CONDITION_RESP, SS_POWER_FWD_START_CONFIRMATION_MSG_REQ, SS_POWER_FWD_START_CONFIRMATION_MSG_RESP, SS_POWER_POWER_REQUEST_MSG, SS_POWER_SHUTDOWN_REQUEST_MSG, SS_POWER_USER_MODE_SET_RESP, SS_POWER_HEARTBEAT_REQ, SS_POWER_HEARTBEAT_RESP, SS_POWER_HARD_RESET_REQ }

Functions

E_CWORD33_Status _CWORD33_SystemConnectToPowerSerivce (HANDLE hApp)

E_CWORD33_Status _CWORD33_SystemConnectToHeartBeatService (HANDLE hApp)

E_CWORD33_Status _CWORD33_SystemConnectToSystemManagerService (HANDLE hApp)

E_CWORD33_Status **SS_ConvLastInfoToOrder** (SS_LAST_INFO_t &curLastMode, std::string &order, const char *p_cfgPath=NULL)

E_CWORD33_Status **SS_LoggerStoreLogs_deleteOldLogAbnrm** (const std::string &log_path, std::vector< std::string > &l_vector, const std::string &f_archive_destination, SI_32 max_num, UI_32 abnrm_total)

E_CWORD33_Status **StartRtUsbLogThread** (HANDLE hApp)

E_CWORD33_Status **StopRtUsbLogThread** (HANDLE hApp)

E_CWORD33_Status RegisterSMSessionAckCallback (E_CWORD33_Status(*CallbackPtr)(HANDLE))

E_CWORD33_Status SendBootModeSetRequestToSystemManager (ESMBootModeInfo hostBootMode)

E_CWORD33_Status SetDataResetModeToSystemManager (ESMDataResetModeInfo dataResetMode)

E_CWORD33_Status SetProgUpdateStateToSystemManager (SMProgUpdateState progUpdateState)

E_CWORD33_Status SendCpuResetRequestToSystemManager (ESMCpuResetReason l_eCpuResetReason, std::string f_messageStr="", std::string f_suffixStr="")

E_CWORD33_Status SetBootLoaderInfoRequestToSystemManager (ESMBootInfoType f_type, UI_32 f_value)

E_CWORD33_Status GetBootLoaderInfoRequestToSystemManager (void *p_info)

E_CWORD33_Status SetWakeupOrderToSystemManager (std::string f_orderName)
E_CWORD33_Status SetNextWakeupTypeToSystemManager (ESMNextWakeupType f_wakeupType)
E_CWORD33_Status SendVarCodeDataToSystemManager (UI_32 f_uiLength, void *f_data)
VOID Set_UseStopCompleteNotificationVs_CWORD33_OnStopFnc_StateVar (BOOL f_SetTrue)
E_CWORD33_Status SendUserInvokedLoggingRequestToSystemManager (eSMUserLogType f_userInvokedLogType, std::string f_messageStr="", std::string f_suffixStr="")
E_CWORD33_Status Send_CWORD33_OnStopResponseToSystemManager (E_CWORD33_Status f_eStatus)
E_CWORD33_Status Get_CWORD33_OnStartExtInfo (T_SS_SM_START_ExtDataStructType &f_info)
E_CWORD33_Status Get_CWORD33_OnStopExtInfo (T_SS_SM_STOP_ExtDataStructType &f_info)
void SendDebugDumpResponseToSystemManager(BOOL f_bFormatStrRequired, PCSTR f_cFormat,...) E_CWORD33_Status Send_CWORD33_EmmcLogsRequestToSystemManager(std E_CWORD33_Status SendClearLogsRequestToSystemManager (TSystemManagerClearLogsInfo *f_info)
E_CWORD33_Status AttachCallbackToSystemManager (HANDLE hApp, UI_32 iCmd, CbFuncPtr fpOnCmd)

AttachCallbackToSystemManager.

E_CWORD33_Status DetachCallbacksFrom_CWORD33_Dispatcher (HANDLE hApp, PCSTR pServiceName, const _CWORD33_ProtocolCallbackHandler *pMsgHandler, UI_32 uiHandlerCount, HANDLE hSession=NULL)

Set of complementary detach functions that accept same arguments as the _CWORD33_Attach_XYZ_ToDispatcher() functions do.

E_CWORD33_Status DetachParentCallbacksFrom_CWORD33_Dispatcher (HANDLE hChildApp, const _CWORD33_ProtocolCallbackHandler *pMsgHandler, UI_32 uiHandlerCount)

HANDLE GetSystemManagerSessionHandle (void)

GetSystemManagerSessionHandle.

E_CWORD33_Status SendShutdownToSystemManager (Pwr_ServiceSetInterface *pData)

SendShutdownToSystemManager.

E_CWORD33_Status SendSystemModeRequestToSystemManager (void)

SendSystemModeRequestToSystemManager.

E_CWORD33_Status SendInitCompReportToSystemManager (void)

SendInitCompReportToSystemManager.

E_CWORD33_Status SendWakeUpToSystemManager (wakeInfo *pData)

SendWakeUpToSystemManager.

E_CWORD33_Status SendStartupConfirmationToSystemManager (StartupConfirmationMsgStrut &l_startupConfirmationMsg)

SendStartupConfirmationToSystemManager.

E_CWORD33_Status SendSystemErrorToSystemManager (E_CWORD33_SystemError f_systemError)

SendSystemErrorToSystemManager.

E_CWORD33_Status Send_CWORD56_HeartBeatRequestToSystemManager (EPWR_HB_REQ_MSG_STRUCT f_HbReq)

Send_CWORD56_HeartBeatRequestToSystemManager.

E_CWORD33_Status SendBootMicroResetNotificationToSystemManager (eSMBootMicroResetReason ResetReason)

SendBootMicroResetNotificationToSystemManager.

E_CWORD33_Status SendDiagLoggingRequestToSystemManager (std::string f_copyDestPathStr)

SendDiagLogEventRequestToSystemManager.

E_CWORD33_Status SendCANLoggingRequestToSystemManager (void)

SendCANLoggingRequestToSystemManager.

E_CWORD33_Status SendDTCLoggingRequestToSystemManager (UI_32 f_dtc)

SendDTCLoggingRequestToSystemManager.

E_CWORD33_Status RegisterBootMicroLogRequestCb (HANDLE hApp, CbFuncPtr fpOnCmd)

RegisterBootMicroLogRequestCb.

E_CWORD33_Status SendBootMicroLogResponseToSystemManager (std::string f_logString)

SendBootMicroLogResponseToSystemManager.

E_CWORD33_Status SendLogStartRequestToSystemManager (EErrorEventType f_errorEventType)

SendLogStartRequestToSystemManager.

E_CWORD33_Status SendLogArtifactRequestToSystemManager (EArtifactId f_artifactId)

SendLogArtifactRequestToSystemManager.

E_CWORD33_Status SendLogCompleteRequestToSystemManager (E_CWORD33_Status f_eStatus=e_CWORD33_StatusOK)

SendLogCompleteRequestToSystemManager.

E_CWORD33_Status SendEelExportRequestToSystemManager (std::string f_path)

SendEelExportRequestToSystemManager.

E_CWORD33_Status OnSystemManagerDebugDump (HANDLE hApp)

DebugDump request callback function.

SS_String GetStr (BOOL f_enum)
SS_String GetStr (EErrorEventType f_enum)
SS_String GetStr (E_CWORD33_SystemError f_enum)
SS_String GetStr (EPWR_LHC_TYPE f_enum)
SS_String GetStr (EPWR_PROD_MODE_TYPE f_enum)
SS_String GetStr (EPWR_SC_COLD_START_REQ_TYPE f_enum)
SS_String GetStr (EPWR_SC__CWORD56__BOOT_MODE_TYPE f_enum)
SS_String GetStr (EPWR_SHUTDOWN_TRIGGER_TYPE f_enum)
SS_String GetStr (EPWR_STARTUP_STAGE_TYPE f_enum)
SS_String GetStr (EPWR_SYSTEM_MODE_INFO f_enum)
SS_String GetStr (EPWR_TRANSPORT_MODE_TYPE f_enum)
SS_String GetStr (EPWR_USER_MODE_CHANGE_REASON_TYPE f_enum)
SS_String GetStr (EPWR_USER_MODE_TYPE f_enum)
SS_String GetStr (EPWR_WAKEUP_FACTOR_TYPE f_enum)
SS_String GetStr (E_CWORD33_Status f_enum)
SS_String GetStr (ESMBootModeInfo f_enum)

SS_String GetStr (ESMCpuResetReason f_enum)
SS_String GetStr (SS_SystemManagerProtocol f_enum)

Variables

const UI_32 **MaxTestCaseName** = ***
const UI_32 **MaxRespMsg** = ***
const long **lProcess_VALIDATION_VALUE** = ***
const int **iProcess_DEFAULT_PROCESS_PRIORITY** = ***
const int **iProcess_MAXIMUM_PROCESS_PRIORITY** = ***
const long **iProcess_DEFAULT_PROCESS_FLAGS**
const int **iProcess_MAXIMUM_NUMBER_OF_PROCESS_ARGUMENTS** = ***
const int **iProcess_MAXIMUM_PROCESS_NAME_LENGTH** = ***

Detailed Description

Class Documentation

struct ARTIFACT_RESPONSE

Class Members:

EArtifactId	ArtifactId	
CHAR	FilePathAndName[MAX_PATH_LENGTH]	

struct THeartBeatSession

Class Members:

BOOL	fAvailable	
HANDLE	hService	
string	szServiceName	

struct _T_CWORD33_CANMileageInfo

CAN Diagnostic status data.

Class Members:

UI_8	DidA_ExtTest_Pres	
UI_8	EngRun_Stat	
UI_8	Odo__CWORD34__H	
UI_8	Odo__CWORD34__L	
UI_8	Odo_LSB_H	
UI_8	Odo_LSB_L	
UI_8	PN14_SupBat_Volt	

struct __CWORD62_DateAndTime

CAN current date and time.

CAN current date and time

Class Members:

UI_8	DateTime_Stat	Date time status
UI_8	DateTimeDay	Date
UI_8	DateTimeHour	Hour
UI_8	DateTimeMinute	Minute
UI_8	DateTimeMonth	Month
UI_8	DateTimeSecond	Second
UI_8	DateTimeYear	Year
UI_8	TimeFormat	Format

struct _SStatisticalCounter

Statistical counter.

Class Members:

UI_8	StatisticalCountBuffer[MAX_STATISTICAL_BUFFER]	Counter values for startup,normal and shutsown phases.
UI_16	u16NormalCounterLength	No of Counter from Normal phase.
UI_16	u16ShutDownCounterLength	No of Counter from Shut down phase.
UI_16	u16StartupCounterLength	No of Counter from startup phase.

struct _SEventLoggerCommonInfo

Event logger common information.

Class Members:

UI_8	BodyCAN_Stat:4	
UI_8	FOT_Temp:2	
UI_8	HeadUnitCAN_Stat:4	
UI_8	HMIInteraction:4	
UI_8	IGN_Status:4	
UI_8	Syst_CWORD61_oltage:6	

struct _SEventLoggerResetInfo

Event logger resets counter information.

Class Members:

UI_8	_CWORD102__ResetInfo	
UI_8	_CWORD56__ResetInfo	

struct _SEventLoggerEventInfo

Event logger *CWORD56* events information.

Class Members:

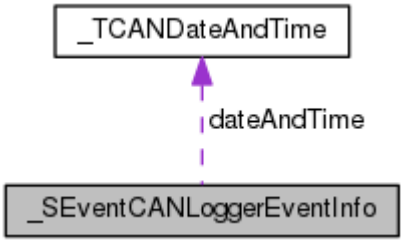
UI_8	EventData[4]	
UI_8	EventGroup	
UI_8	EventIdentifier	
UI_8	NumberOfEvents	

struct _SEventCANLoggerEventInfo

Event logger CAN events information.

Collaboration diagram for _SEventCANLoggerEventInfo:

IMAGE



Class Members:

STCanCurrentDateTime	dateAndTime	
BOOL	success	
UI_32	triggerNumber	
BOOL	valid	

struct _PwCompleteAck

Class Members:

CHAR	szServiceName[MAX_QUEUE_NAME_SIZE]	
UI_16	unSessionId	

struct _SS_PSCurrentState

Class Members:

CHAR	printRespmsg[MaxRespMsg]	
CHAR	testCaseId[MaxTestCaseName]	

Macro Definition Documentation

#define _CWORD33__MAKE_DEFAULT_CALLBACK(x)

Value:x.onInitilization = _CWORD33_OnInitialization; \
x.onDestroy = _CWORD33_OnDestroy; \
x.onStart = _CWORD33_OnStart; \
x.onStop = _CWORD33_OnStop; \
x.onDebugDump = _CWORD33_OnDebugDump; \
x.createStateMachine = _CWORD33_CreateStateMachine; \
x.ssFrameworkInterface = _CWORD33_SSFrameworkInterface; \
x.onPreStart = _CWORD33_OnDummy; \
x.onPreStop = _CWORD33_OnDummy; \
x.onBackgroundStart = _CWORD33_OnDummy; \
x.onBackgroundStop = _CWORD33_OnDummy;

#define NTFY_SSSystemMgrUserMode SERVICE_SYSMANAGER"/UserMode"

TRUE : UserMode is UserOn FALSE : UserMode is UserOff Notification for User Mode

#define SSDEBUGDUMP(args...) (SendDebugDumpResponseToSystemManager(TRUE, ## args))

Redirect either formatted, or unformatted response output to System Manager.

See also:

SendDebugDumpResponseToSystemManager

Typedef Documentation

typedef enum _SS_PowerServiceProtocol SS_PowerServiceProtocol

power service protocol > define all protocol messages in and out of power that are pertinent to the functionality offered by power.

Enumeration Type Documentation

enum SS_PowerServiceProtocol

power service protocol > define all protocol messages in and out of power that are pertinent to the functionality offered by power.

Enumerator

SS_POWER_PRINT_CONNECTIONS client to service

SS_POWER_PRINT_STACK client to service

SS_POWER_WAKEUP_COMPLETE service to client // used by power state machine

SS_POWER_SHUTDOWN_COMPLETE service to client // used by power state machine

SS_POWER_VOLTAGE_STATE client to service

SS_POWER_LVI2_SHUTDOWN_COMPLETE service to client

SS_POWER_CRANK_STATE client to service

SS_POWER_WAKEUP_MODULES_CMPL_RSPN service to shadow

SS_POWER_SYSTEM_LAUNCH_COMPLETE client to service

SS_POWER_STATE_CHANGE_REQ client to service : send wakeup to System Manager

SS_POWER_STATE_CHANGE_RESP Response to request of power state transition to power_supply_manager_shadow.

SS_POWER_COMM_WAKEUP client to service : Power On expected

SS_POWER_SHUTDOWN_REQ client to service : Comm Sleep from Shadow

SS_POWER_SHUTDOWN_RESP service to shadow : Shutdown complete from System Manager

SS_POWER_CRNT_STATE_QUERY client to service

SS_POWER_CRNT_STATE_QUERY_RSPN service to client

SS_POWER_SYSTEM_MODE_INFO_REQ Client (Shadow) to Service (Power Service)

SS_POWER_SYSTEM_MODE_INFO_RESP Service to client *****.

SS_POWER_INITCOMP_REP Client (Shadow) to Service (Power Service)

SS_POWER_PUBLISH_POWER_POPUP_REQ Shadow to Pwr Svc.

SS_POWER_PUBLISH_POWER_POPUP_RESP Pwr Svc to Shadow.

SS_POWER_PUBLISH_SHUTDOWN_CONDITION_REQ Shadow to Pwr Svc.

SS_POWER_PUBLISH_SHUTDOWN_CONDITION_RESP Pwr Svc to Shadow.

SS_POWER_FWD_START_CONFIRMATION_MSG_REQ Shadow to Pwr Svc.

SS_POWER_FWD_START_CONFIRMATION_MSG_RESP Pwr Svc to Shadow.

SS_POWER_POWER_REQUEST_MSG Shadow to Pwr Svc.

SS_POWER_SHUTDOWN_REQUEST_MSG Shadow to Pwr Svc.

SS_POWER_USER_MODE_SET_RESP Power Service to Shadow.

SS_POWER_HEARTBEAT_REQ Shadow to Power Service.

SS_POWER_HEARTBEAT_RESP Power Service to Shadow.

SS_POWER_HARD_RESET_REQ Power Service to Shadow.

enum SS_Sources

Enumerator

SS_SOURCE_MSC Any mass storage device (USB-MSC, SD, HDD etc.,)

SS_SOURCE_MTP *CWORD104* , MTP protocol devices(like *CWORD28* over USB)

SS_SOURCE__CWORD73__APP *CWORD73* based audio application (*CWORD74* over BT/USB) - *CWORD22* , *CWORD13* etc.,

SS_SOURCE__CWORD73__VIDEO_MODE *CWORD73* Video Mode (*CWORD57* playback)

SS_SOURCE_STREAMING Streaming over Wi-fi.

SS_SOURCE_BT_AUDIO Bluetooth Audio.

SS_SOURCE_BT_SPP Bluetooth SPP based audio application (*CWORD28* /BB etc.,)

SS_SOURCE_SPEECH Speech audio playback.

SS_SOURCE_USB will be deprecated soon

enum SS_Zones

Enumerator

SS_ZONE_DRIVER Zero/Driver zone.

enum E_AUDIO_PORT

Enumerator

SS_AUDIO_INPUT_PORT_1 Enum representing MCASP2.

SS_AUDIO_INPUT_PORT_2 Enum representing MCBSP.

SS_AUDIO_INPUT_PORT_3 Enum representing MCASP3.

SS_AUDIO_INPUT_PORT_4 Enum representing MCASP3.

SS_AUDIO_INPUT_PORT_5 Enum representing other port.

enum eSSLoggerCANEvent

eSSLoggerCANEvent

Note:

Events that are published to CAN for HK and CAN logging. Data structures attached to the single events can be found in below table

EventId	Return type
eSSLoggerCANEventStart	None
eSSLoggerCANEventError	STEventCANLoggerEventInfo
eSSLoggerCANEventFinished	STEventCANLoggerEventInfo

Enumerator

eSSLoggerCANEventError Logging has started for HK and CAN events.

eSSLoggerCANEventFinished Logging has aborted with an error.

Logging has finished succesfully

enum eSSLoggerCANProtocolID

eSSLoggerCANProtocolID

Note:

These IDs depict the possible events coming from CAN to logger_service

Enumerator

eSSLoggerCANProtocolIDDTCTrigger Start logging to USB.

Start logging of DTC to emmc

Function Documentation

E_CWORD33_Status _CWORD33_SystemConnectToHeartBeatService (HANDLE *hApp*)

Summary

Called from framework for every app to start connection to HeartBeat

Parameters:

in	*hApp*	HANDLE - App handle, Handle to message queue of

		HeartBeat Service.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Preconditions

Change of the internal state

Change of internal state according to the API does not occur.

Classification

Public

Type

None

See also:

_CWORD33_GetAppName _CWORD33_AttachCallbacksToDispatcher

E_CWORD33_Status _CWORD33_SystemConnectToPowerSerivce (HANDLE *hApp*)

Summary

Subscribe and attach call backs to notifications, to PowerSerivce!

Parameters:

in	*hApp*	HANDLE - App handle

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared

	memory, etc.)

Preconditions

Change of the internal state

Change of internal state according to the API does not occur.

Classification

Public

Type

None

See also:

_CWORD33_GetAppName _CWORD33_SubscribeNotificationsWithCallback

E_CWORD33_Status _CWORD33_SystemConnectToSystemManagerService (HANDLE *hApp*)

Summary

Subscribe and attach callbacks to System Manager Service!

Parameters:

in	*hApp*	HANDLE - App handle.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Preconditions

Change of the internal state

Change of internal state according to the API does not occur.

Classification

Public

Type

None

See also:

_CWORD33_GetAppName SystemManagerOpenSender

E_CWORD33_Status AttachCallbackToSystemManager (HANDLE *hApp*, UI_32 *iCmd*, CbFuncPtr *fpOnCmd*)

AttachCallbackToSystemManager.

Note:

This function is called by System Manager session clients to attach a callback function to the client application with System Manager configured as the specified sender of the specified protocol message.

Parameters:

in	*f_hApp*	HANDLE - Handle to message queue of the System Manager attached client.
in	*iCmd*	UI_32 iCmd - Command ID
in	*fpOnCmd*	CbFuncPtr fpOnCmd - Function pointer to be called in the form upon the invokation of the specified command ID. The callback function must have the following signature : void (*)(HANDLE).

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status Get_CWORD33_OnStartExtInfo (T_SS_SM_START_ExtDataStructType & *f_info*)

Summary

API to get the extended information of the start-up parameters.

Parameters:

out	*f_info*	T_SS_SM_START_ExtDataStructType - Startup parameters extended information

T_SS_SM_START_ExtDataStructType Structure

```
1 typedef struct {
2 {
3     BOOL            isProgUpdated;
4     EMRelaunchStatus relaunchStatus;
5     BOOL            isMapUpdated;
6     BOOL            isMapDiffUpdated;
7     uint8_t         reserved[];
8 }T_SS_SM_START_ExtDataStructType;
```

isProgUpdatedPresence or absence of a program update

TRUE : Yes program update
FALSE : No program update

relaunchStatusRelaunch state of self-service
e_SS_SM_RELAUNCH_STATUS_NONE : Relaunch non-occurrence (first start)
e_SS_SM_RELAUNCH_STATUS_SAFE : Normal Relaunch
e_SS_SM_RELAUNCH_STATUS_ERR : Abnormal Relaunch

isMapUpdatedPresence or absence of a map update
TRUE : Yes map update
FALSE : No map update

isMapDiffUpdatedThe presence or absence of the map difference updating
TRUE : Yes map difference update
FALSE : No map difference update

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusBadConnection	Failure to socket connection
e_CWORD33_StatusFail	Some sort of error occurred(Socket Connection/Memory Allocation/Event Control)
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

Get_CWORD33_OnStopExtInfo, _CWORD33_OnStart

E_CWORD33_Status Get_CWORD33_OnStopExtInfo (T_SS_SM_STOP_ExtDataStructType & *f_info*)

Summary

API to get the extended information of the end parameter.

Parameters:

out	*f_info*	T_SS_SM_STOP_ExtDataStructType - Exit extended parameter information

T_SS_SM_STOP_ExtDataStructType Structure

```
1 typedef struct {
2 {
3     BOOL          isProgUpdated;
4     uint8_t       reserved[];
5 }T_SS_SM_START_ExtDataStructType;
```

isProgUpdatedPresence or absence of a program update
TRUE : Yes program update
FALSE : No program update

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusBadConnection	Failure to socket connection
e_CWORD33_StatusFail	Some sort of error occurred(Socket Connection/Memory Allocation/Event Control)
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

Get_CWORD33_OnStartExtInfo, _CWORD33_OnStop

E_CWORD33_Status GetBootLoaderInfoRequestToSystemManager (void * *p_info*)

Summary

This function is called to get Boot-Loader Information.

Parameters:

out	*p_info*	void* - pointer for Boot-Loader Information

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusBadConnection	Socket connection error
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

None

SS_String GetStr (BOOL *f_enum*)

Summary

change member from BOOL type to string

Parameters:

in	*f_enum*	BOOL - the member what will convert farm BOOL type to string type

		1 typedef int BOOL; 2 #define TRUE (1==1) 3 #define true (1==1) 4 #define FALSE (0==1)

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EErrorEventType *f_enum*)

Summary

change member from EErrorEventType type to string

Parameters:

in	*f_enum*	EErrorEventType - the member what will convert farm EErrorEventType type to string type.error event type. 1 enum EErrorEventType { 2 eErrorEventTypeProcessCrash = ***, 3 eErrorEventTypeHeartBeatFailure, 4 eErrorEventTypeSystemLowMemory, 5 eErrorEventTypeReserved1, // Formerly SS_SM_EVENT_ERROR_HIGH_CPU_LOAD, now HMI enum place holder. 6 eErrorEventTypeUserInvokedUserForceReset, 7 eErrorEventTypeReserved2, 8 eErrorEventTypeReserved3, 9 eErrorEventTypeUserInvokedCollectAllLogs, 10 eErrorEventTypeBootMicroReset, 11 eErrorEventTypeProcessExit, 12 eErrorEventTypeUserInvokedCollectScreenShot, 13 eErrorEventTypeUserInvokedCollect_CWORD33_Logs, 14 eErrorEventTypeEelExport, 15 eErrorEventType_CWORD33_EmmcLogs, 16 eErrorEventTypeDiagEvent, 17 eErrorEventTypeCanEvent, 18 eErrorEventTypeDtcEvent, 19 eErrorEventTypeModConnFailed, 20 eErrorEventTypeStartRespFailed,

		21 eErrorEventTypeUserInvokedCollectDevLogs, 22 eErrorEventTypeModuleInvokedResetRequest, 23 eErrorEventTypeModuleInvokedCollectDebugLogs, 24 eErrorEventTypeUserInvokedClearLogs, 25 eErrorEventTypeUserInvokedCollectNaviLog, 26 eErrorEventTypeGroupRelaunch, 27 eErrorEventTypeMaxValue 28 };

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (E_CWORD33_SystemError *f_enum*)

Summary

change member from E_CWORD33_SystemError type to string

Parameters:

in	*f_enum*	E_CWORD33_SystemError - the member what will convert farm E_CWORD33_SystemError type to string type.The type of system error. typedef enum _E_CWORD33_SystemError { e_CWORD33_SystemErrorNone = ***, e_CWORD33_DSPHardwareReset = *** } E_CWORD33_SystemError;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_LHC_TYPE *f_enum*)

Summary

change member from EPWR_LHC_TYPE type to string

Parameters:

in	*f_enum*	EPWR_LHC_TYPE - the member what will convert farm EPWR_LHC_TYPE type to string type.Limp home cutoff protocol type. typedef enum { epslhcINVALID = ***, epslhcDISABLED = ***, epslhcENABLED = *** } ePwrServiceLHCType, EPWR_LHC_TYPE;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_PROD_MODE_TYPE *f_enum*)

Summary

change member from EPWR_PROD_MODE_TYPE type to string

Parameters:

in	*f_enum*	EPWR_PROD_MODE_TYPE - the member what will convert farm EPWR_PROD_MODE_TYPE type to string

		type.The type of production mode protocol. typedef enum { epspmINVALID = ***, epspmDISABLED = ***, epspmENABLED = *** } ePwrServiceProdModeType, EPWR_PROD_MODE_TYPE;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_SC_COLD_START_REQ_TYPE *f_enum*)

Summary

change member from EPWR_SC_COLD_START_REQ_TYPE type to string

Parameters:

in	*f_enum*	EPWR_SC_COLD_START_REQ_TYPE - the member what will convert farm E_CWORD33_SystemError type to string type.The type of cold start. typedef enum { epsscrtINVALID = ***, epsscrtNOT_REQUIRED = ***, epsscrtREQUIRED = *** } EPWR_SC_COLD_START_REQ_TYPE;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_SC__CWORD56__BOOT_MODE_TYPE *f_enum*)

Summary

change member from EPWR_SC__CWORD56__BOOT_MODE_TYPE type to string

Parameters:

in	*f_enum*	EPWR_SC__CWORD56__BOOT_MODE_TYPE - the member what will convert farm EPWR_SC__CWORD56__BOOT_MODE_TYPE type to string type.The type of startup confirmation. typedef enum { eps_CWORD56_bmINVALID = ***, eps_CWORD56_bmAPPLICATION_MODE = ***, eps_CWORD56_bmPROGRAMMING_MODE = *** } EPWR_SC__CWORD56__BOOT_MODE_TYPE;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_SHUTDOWN_TRIGGER_TYPE *f_enum*)

Summary

change member from EPWR_SHUTDOWN_TRIGGER_TYPE type to string

Parameters:

in	*f_enum*	EPWR_SHUTDOWN_TRIGGER_TYPE - the member what will convert farm EPWR_SHUTDOWN_TRIGGER_TYPE type to string

		type.The type of shutdown trigger. typedef enum { epssdmsdtINVALID = ***, epssdmsdtTESTACC_OFF = ***, epssdmsdtON_KEY = ***, epssdmsdtIGN_LOCK = ***, epssdmsdtPWR_SAVE = ***, epssdmsdtTMP_STARTUP = ***, epssdmsdtDIAG_DEACTIVATION = ***, epssdmsdtABNORMAL_VOLTAGE = ***, epssdmsdtABNORMAL_TEMP = ***, epssdmsdtBATTERYCUTOFF = ***, epssdmsdtLIMPHOME = ***, epssdmsdtHU_CAN_ERROR = ***, epssdmsdtBODY_CAN_ERROR = ***, epssdmsdtTRANSPORT_MODE = ***, epssdmsdtPRODUCTION_MODE = ***, epssdmsdtIGN_OFF = ***, epssdmsdtGENERIC_ERROR_RESET = ***, epssdmsdtFATAL_ERROR_RESET = ***, epssdmsdtUSER_DATA_RESET = ***, epssdmsdtFACTORY_DATA_RESET = ***, epssdmsdtFAST_SLEEP_MODE = ***, epssdmsdtNORMAL_RESET = ***, epssdmsdtPROGUPDATE_RESET = ***, } ePowerSrvPwrShutdownTriggerType, EPWR_SHUTDOWN_TRIGGER_TYPE;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_STARTUP_STAGE_TYPE *f_enum*)

Summary

change member from EPWR_STARTUP_STAGE_TYPE type to string

Parameters:

in	*f_enum*	EPWR_STARTUP_STAGE_TYPE - the member what will convert farm EPWR_STARTUP_STAGE_TYPE type to string type.The type of startup stage. typedef enum { epssusfINVALID = ***, epssusSYSTEM_SERVICES_STARTED = ***, epssusALL_SERVICES_LAUNCHED = *** }ePowerSrvStartupStageType, EPWR_STARTUP_STAGE_TYPE;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_SYSTEM_MODE_INFO *f_enum*)

Summary

change member from EPWR_SYSTEM_MODE_INFO type to string

Parameters:

in	*f_enum*	EPWR_SYSTEM_MODE_INFO - the member what will convert farm EPWR_SYSTEM_MODE_INFO type to string type.The type of system mode info. typedef enum { epssinfINVALID = ***, epssinfNORMAL = ***, epssinfPROGRAMMING = ***, }ePowerSrvSystemModeInfo, EPWR_SYSTEM_MODE_INFO;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_TRANSPORT_MODE_TYPE *f_enum*)

Summary

change member from EPWR_TRANSPORT_MODE_TYPE type to string

Parameters:

in	*f_enum*	EPWR_TRANSPORT_MODE_TYPE - the member what will convert farm EPWR_TRANSPORT_MODE_TYPE type to string type.The type of transport mode protocol. typedef enum { epstmINVALID = ***, epstmDISABLED = ***, epstmENABLED = *** } ePwrServiceTransportModeType, EPWR_TRANSPORT_MODE_TYPE;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_USER_MODE_CHANGE_REASON_TYPE *f_enum*)

Summary

change member from EPWR_USER_MODE_CHANGE_REASON_TYPE type to string

Parameters:

in	*f_enum*	EPWR_USER_MODE_CHANGE_REASON_TYPE - the member what will convert farm EPWR_USER_MODE_CHANGE_REASON_TYPE type to string type.The type of user mode change reason.

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_USER_MODE_TYPE *f_enum*)

Summary

change member from EPWR_USER_MODE_TYPE type to string

Parameters:

in	*f_enum*	EPWR_USER_MODE_TYPE - the member what will convert farm EPWR_USER_MODE_TYPE type to string type.The type of user mode. typedef enum { epsumINVALID = ***, epsumOFF = ***, epsumON = ***, } ePwrServiceUserModeType, EPWR_USER_MODE_TYPE;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (EPWR_WAKEUP_FACTOR_TYPE *f_enum*)

Summary

change member from EPWR_WAKEUP_FACTOR_TYPE type to string

Parameters:

in	*f_enum*	EPWR_WAKEUP_FACTOR_TYPE - the member what will convert farm EPWR_WAKEUP_FACTOR_TYPE type to string type.The type of startup reason. typedef enum { epswfINVALID = ***, epswfTESTACC = ***, epswfON_KEY = ***, epswfIGN_ACC = ***, epswfDR_OPEN_CLOSE = ***, epswfDX_ACTIVATION = ***, epswfETHERNET = ***, epswfPASS_ACTIVATION = ***, epswfSPVACTIVATION = ***, epswfUSER_DATA_RESET = *** } ePowerSrvWakeupFactors, EPWR_WAKEUP_FACTOR_TYPE;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (E_CWORD33_Status *f_enum*)

Summary

change member from E_CWORD33_Status type to string

Parameters:

in	*f_enum*	E_CWORD33_Status - the member what will convert farm E_CWORD33_Status type to string type.Status and return types. typedef enum e_CWORD33_Status { e_CWORD33_StatusEmptyMediaList = -10, ///< Empty media list e_CWORD33_StatusSessionLimitMaxedOut = -9, ///< Maximum session limit reached e_CWORD33_StatusDbRecNotFound = -8, ///< Database record not found e_CWORD33_StatusDbResultError = -7, ///< Database result error e_CWORD33_StatusDbExecuteFail = -6, ///< Database execute fail e_CWORD33_StatusSemCloseFail = -5, ///< Semaphore close failed e_CWORD33_StatusSemUnlockFail = -4, ///< Semaphore unlock failed e_CWORD33_StatusSemLockFail = -3, ///< Semaphore lock failed e_CWORD33_StatusFail = -2, ///< Failed e_CWORD33_StatusErrOther = -1, ///< Unknown error e_CWORD78_StatusOK = ***, ///< Success / Pass / OK e_CWORD33_StatusInvldBuf = ***, ///< Invalid buffer e_CWORD33_StatusInvldHandle = ***, ///< Invalid handle e_CWORD33_StatusInvldHndlType = ***, ///< Invalid handle type e_CWORD33_StatusInvldQName = ***, ///< Invalid messasge queue name e_CWORD33_StatusMsgQFull = ***, ///< Message queue full e_CWORD33_StatusInvldNotification = ***,///< The Notification event not present e_CWORD33_StatusInvldParam = ***, ///< Invalid parameter e_CWORD33_StatusInvldBufSize = ***, ///< Buf size too small e_CWORD33_StatusInvldID = ***, ///< Unrecognized ID e_CWORD33_StatusCannotRelease = ***, ///< Cannot release resource e_CWORD33_StatusBadConnection = ***, ///< Could not locate resource e_CWORD33_StatusExit = ***, ///< Normal application termination e_CWORD33_StatusNotImplemented = ***, ///< incomplete feature e_CWORD33_StatusThreadBusy = ***, ///< Joined thread is already being joined e_CWORD33_StatusThreadSelfJoin = ***, ///< Thread is joining itself e_CWORD78_StatusThreadInvalidVal = ***, ///< Invalid value passed e_CWORD33_StatusThreadNotExist = ***, ///< The thread does not exist e_CWORD33_StatusFault = ***, ///< A fault occurred while attempting to make call

		e_CWORD33_StatusServNotFound = ***, ///< Service not present in serv dir e_CWORD33_StatusServerInUse = ***, ///< Service already processing 1 client request e_CWORD33_StatusDbIndexing = ***, ///< Database Indexing in progress e_CWORD33_StatusNullPointer = ***, e_CWORD33_StatusMsgNotProcessed = ***, e_CWORD33_StatusFileLoadSuccess = ***, ///< File Load Success e_CWORD33_StatusFileLoadError = ***, ///< File Load Error e_CWORD33_StatusAccessError = ***, ///< Error when accessing resource e_CWORD33_StatusDuplicate = ***, ///< Duplicate entry e_CWORD33_StatusMsgQEmpty = ***, ///< Message queue empty e_CWORD33_StatusThreadAlreadyRunning = ***, e_CWORD33_StatusErrNoEBADF = ***, ///< Bad file descriptor e_CWORD33_StatusErrNoEAGAIN = ***, ///< Resource unavailable, try again e_CWORD33_StatusErrNoEINTR = ***, ///< Interrupted system call e_CWORD33_StatusSessionErr = ***, ///< Error in session handling e_CWORD33_StatusDBCorrupt = ***, ///< Database corrupt e_CWORD33_StatusDBFileNotFound = *** ///< Database file not found } E_CWORD33_Status, *PE_CWORD33_Status;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (ESMBootModeInfo *f_enum*)

Summary

change member from ESMBootModeInfo type to string

Parameters:

in	*f_enum*	ESMBootModeInfo - the member what will convert farm ESMBootModeInfo type to string type.The type of boot mode information. typedef enum { e_SS_SM_BOOT_MODE_INVALID = -1 , e_SS_SM_BOOT_MODE_APPLICATION , e_SS_SM_BOOT_MODE_PROGRAMMING }ESMBootModeInfo;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (ESMCpuResetReason *f_enum*)

Summary

change member from ESMCpuResetReason type to string

Parameters:

in	*f_enum*	ESMCpuResetReason - the member what will convert farm ESMCpuResetReason type to string type.The type of CPU reset reason. typedef enum { e_SS_SM_CPU_RESET_REASON_INVALID = -1 , e_SS_SM_CPU_RESET_REASON_NORMAL = *** , e_SS_SM_CPU_RESET_REASON_DATA_RESET , e_SS_SM_CPU_RESET_REASON_USER_FORCE_RESET , e_SS_SM_CPU_RESET_REASON_GENERIC , e_SS_SM_CPU_RESET_REASON_GENERIC_ERR = e_SS_SM_CPU_RESET_REASON_GENERIC , e_SS_SM_CPU_RESET_REASON_DSP_ERR , e_SS_SM_CPU_RESET_REASON_IMMRESET_NORMAL , e_SS_SM_CPU_RESET_REASON_CRITICAL_ERR }ESMCpuResetReason;

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

SS_String GetStr (SS_SystemManagerProtocol *f_enum*)

Summary

change member from SS_SystemManagerProtocol type to string

Parameters:

in	*f_enum*	SS_SystemManagerProtocol - the member what will convert farm SS_SystemManagerProtocol type to string type.The type of system manager protocol.

Return values:

SS_String	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

HANDLE GetSystemManagerSessionHandle (void)

GetSystemManagerSessionHandle.

Note:

This function returns the session handle with System Manager

Returns:

HANDLE - A session handle with System Manager

E_CWORD33_Status OnSystemManagerDebugDump (HANDLE *hApp*)

DebugDump request callback function.

Note:

This function is called when System Manager requests the receiving application to generate their debugdump.

Parameters:

hApp	

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status RegisterBootMicroLogRequestCb (HANDLE *hApp*, CbFuncPtr *fpOnCmd*)

RegisterBootMicroLogRequestCb.

This function is called by Logging Shadow to register a callback function for the boot micro log request that may be generated by System Manager when an error event occurs.

Parameters:

in	*hApp*	_CWORD3_002 Framework Application Handle
in	*fpOnCmd*	Call back function with the signature: E_CWORD33_Status fnct(HANDLE hApp);

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status RegisterSMSessionAckCallback (E_CWORD33_Status(*)(HANDLE) *CallbackPtr*)

Summary

API for setting the _CWORD33_OnStop response

Parameters:

in	*CallbackPtr*	E_CWORD33_Status - Pointer to a callback function

TRUE : Not carried out in the automatic response.
FALSE : Carry out the response in automatic. (Default)

Return values:

VOID	None

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

None

E_CWORD33_Status Send_CWORD33_OnStopResponseToSystemManager (E_CWORD33_Status *f_eStatus*)

Summary

API to send a _CWORD33_OnStop response

Parameters:

in	*f_eStatus*	E_CWORD33_Status - End status

e_CWORD33_StatusOK : End processing completion
e_CWORD33_StatusFail : End processing uncompleted

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget only

See also:

None

E_CWORD33_Status Send_CWORD56_HeartBeatRequestToSystemManager (EPWR_HB_REQ_MSG_STRUCT *f_HbReq*)

Send_CWORD56_HeartBeatRequestToSystemManager.

Note:

This function forwards a *CWORD56* heartbeat request received by power services from the power shadow to System Manager.

Parameters:

in		

E_CWORD33_Status SendBootMicroLogResponseToSystemManager (std::string *f_logString*)

SendBootMicroLogResponseToSystemManager.

This function is called by Logging Shadow to provide a string representation of the binary boot-micro log data.

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendBootMicroResetNotificationToSystemManager (eSMBootMicroResetReason *ResetReason*)

SendBootMicroResetNotificationToSystemManager.

Parameters:

ResetReason	Enumerated boot micro reset reason.

This function is called by Logging Shadow to inform System Manager of a boot micro reset condition.

E_CWORD33_Status SendBootModeSetRequestToSystemManager (ESMBootModeInfo *hostBootMode*)

Summary

API to record the mode of the next boot in a non-volatile area.

Parameters:

in	*hostBootMode*	ESMBootModeInfo - BOOT mode information

e_SS_SM_BOOT_MODE_APPLICATION : Normal start
e_SS_SM_BOOT_MODE_PROGRAMMING : Version up start

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method only

See also:

None

E_CWORD33_Status SendCANLoggingRequestToSystemManager (void)

SendCANLoggingRequestToSystemManager.

Note:

This function is called by logger to initiate log artifact collection and storage when signaled via CAN.

Returns:

Status E_CWORD33_Status - success or error

void SendDebugDumpResponseToSystemManager (BOOL f_bFormatStrRequired, PCSTR f_cFormat, ...) E_CWORD33_Status Send_CWORD33_EmmcLogsRequestToSystemManager(std E_CWORD33_Status SendClearLogsRequestToSystemManager (TSystemManagerClearLogsInfo * *f_info*)

Summary

Send a debug dump response to system_manager

Parameters:

in	*f_bFormatStrRequired*	BOOL Specify whether to dump the application name or not
in	*f_cFormat*	PCSTR Display format for the dump (the same character strings format as the printf format)
in	*arg_list*	... Dump data in line with f_cFormat format

f_bFormatStrRequired
true Dump the application name
false Do not dump the application name

Return values:

None	

Precondition

None

Change in the internal status

The change in the internal status does not occur by this API.

Classification

Public

Type

Fire and Forget only

See also:

Summary

API to request a copy to the specified path of the LOG file of the built-in non-volatile area.

Parameters:

in	*f_path*	std::string - LOG destination file PATH

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldBuf	Invalid buffer
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method only

See also:

SendClearLogsRequestToSystemManager

Summary

API to clear the stored in the built-in non-volatile area log.

Parameters:

in	*f_info*	TSystemManagerClearLogsInfo - A pointer to the LOG clear information

TSystemManagerClearLogsInfo Structure

```
1 typedef struct T_SystemManagerClearLogsInfo{
2 {
3     void *rsv;
4 }TSystemManagerClearLogsInfo;
```

rsv Reserve area

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle

e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Sync

See also:

None

E_CWORD33_Status SendCpuResetRequestToSystemManager (ESMCpuResetReason *l_eCpuResetReason*, std::string *f_messageStr* = "", std::string *f_suffixStr* = "")

Summary

API to request LOG saved to the internal non-volatile area

Parameters:

in	*l_eCpuResetReason*	ESMCpuResetReason - Reset factor
in	*f_messageStr*	std::string - Abnormal message string
in	*f_suffixStr*	std::string - Suffix string of abnormal LOG file

enum ESMCpuResetReason Variable
e_SS_SM_CPU_RESET_REASON_NORMAL : Normal Reset
e_SS_SM_CPU_RESET_REASON_DATA_RESET : Data Reset
e_SS_SM_CPU_RESET_REASON_USER_FORCE_RESET : User forced reset
e_SS_SM_CPU_RESET_REASON_GENERIC_ERR : Generic Error reset
e_SS_SM_CPU_RESET_REASON_DSP_ERR : DSP Error
e_SS_SM_CPU_RESET_REASON_IMMRESET_NORMAL : Reset immediately

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle

e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

None

E_CWORD33_Status SendDiagLoggingRequestToSystemManager (std::string *f_copyDestPathStr*)

SendDiagLogEventRequestToSystemManager.

Note:

This function is called by logger to initiate log artifact collection and storage on behalf of diagnostic services.

Parameters:

f_copyDestPathStr	

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendDTCLoggingRequestToSystemManager (UI_32 *f_dtc*)

SendDTCLoggingRequestToSystemManager.

Note:

This function is called by logger to initiate log artifact collection and storage when signaled via diagnostic command.

Parameters:

f_dtc	Diagnostic trouble code.

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendEelExportRequestToSystemManager (std::string *f_path*)

SendEelExportRequestToSystemManager.

Note:

This function forwards the Device Detection EEL_export trigger received by SS Logger to System Manager.

Parameters:

f_path	Destination path for the trigger artifacts.

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendInitCompReportToSystemManager (void)

SendInitCompReportToSystemManager.

Note:

This function is called to send init comp report to System Manager

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendLogArtifactRequestToSystemManager (EArtifactId *f_artifactId*)

SendLogArtifactRequestToSystemManager.

Note:

This function is call by SS_Logger to request SM to acquire and

Parameters:

in	*f_artifactId,enumerated*	artifact number.

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendLogCompleteRequestToSystemManager (E_CWORD33_Status *f_eStatus* = e_CWORD33_StatusOK)

SendLogCompleteRequestToSystemManager.

Note:

This function is call by SS_Logger when the logging process has completed and no additional request for artifacts will be generated.

Parameters:

none.	

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendLogStartRequestToSystemManager (EErrorEventType *f_errorEventType*)

SendLogStartRequestToSystemManager.

Note:

This function is call by SS_Logger when the logging process has completed and no additional request for artifacts will be generated.

Parameters:

none.	

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendShutdownToSystemManager (Pwr_ServiceSetInterface * *pData*)

SendShutdownToSystemManager.

Note:

This function is called to send Shutdown command to System Manager

Parameters:

in	*pData*	Pwr_ServiceSetInterface *pData - Pointer to Shutdown Command data

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendStartupConfirmationToSystemManager (StartupConfirmationMsgStrut & *l_startupConfirmationMsg*)

SendStartupConfirmationToSystemManager.

Note:

This function is called to send the startup confirmation information to System Manager

Parameters:

in	*l_startupConfirmationMsg*	StartupConfirmationMsgStrut &l_startupConfirmationMsg - startup confirmation data

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendSystemErrorToSystemManager (E_CWORD33_SystemError *f_systemError*)

SendSystemErrorToSystemManager.

Note:

This function is called by an application via Native Services to send notice of a critical error to System Manager when a non recoverable run-time error is encountered.

Parameters:

in		

E_CWORD33_Status SendSystemModeRequestToSystemManager (void)

SendSystemModeRequestToSystemManager.

Note:

This function is called to send system mode request to System Manager

Returns:

Status E_CWORD33_Status - success or error

E_CWORD33_Status SendUserInvokedLoggingRequestToSystemManager (eSMUserLogType *f_userInvokedLogType*, std::string *f_messageStr* = "", std::string *f_suffixStr* = "")

Summary

API to request LOG saved to the internal non-volatile area

Parameters:

in	*f_userInvokedLogType*	eSMUserLogType - User invoked log type.
in	*f_messageStr*	std::string - Abnormal message string
in	*f_suffixStr*	std::string - Suffix string of abnormal LOG file

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

SendClearLogsRequestToSystemManager

E_CWORD33_Status SendVarCodeDataToSystemManager (UI_32 *f_uiLength*, void * *f_data*)

Summary

In debugging, API to log the character information of variable code.

Parameters:

in	*f_uiLength*	UI_32 - Data length of f_data
in	*f_data*	void - Variable code(char type)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldBuf	Invalid buffer
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle

e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget only

See also:

SendClearLogsRequestToSystemManager

E_CWORD33_Status SendWakeUpToSystemManager (wakeInfo * *pData*)

SendWakeUpToSystemManager.

Note:

This function is called to send the Wakeup command to System Manager

Parameters:

in	*pData*	wakeInfo *pData - Pointer to Wakeup Command data

Returns:

Status E_CWORD33_Status - success or error

VOID Set_UseStopCompleteNotificationVs_CWORD33_OnStopFnc_StateVar (BOOL *f_SetTrue*)

Summary

API for setting the _CWORD33_OnStop response

Parameters:

in	*f_setTrue*	BOOL - Response presence or absence of specified

TRUE : Not carried out in the automatic response.

FALSE : Carry out the response in automatic. (Default)

Return values:

VOID	None

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

None

E_CWORD33_Status SetBootLoaderInfoRequestToSystemManager (ESMBootInfoType *f_type*, UI_32 *f_value*)

Summary

API to perform a write request to the specified field of shared memory with BootLoader.

Parameters:

in	*f_type*	ESMBootInfoType - Configuration information type
in	*f_value*	UI_32 - Set value

enum ESMBootInfoType Variable
e_SS_SM_BOOT_INFO_MDUPDATE : Writing to LBM_RAM_t.mdUpdate
e_SS_SM_BOOT_INFO_SDUPDATE : Writing to LBM_RAM_t.sdUpdate

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusBadConnection	Socket connection error
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

None

E_CWORD33_Status SetDataResetModeToSystemManager (ESMDataResetModeInfo *dataResetMode*)

Summary

API to record the data reset mode of the next boot in a non-volatile area.

Parameters:

in	*dataResetMode*	ESMDataResetModeInfo - Data reset mode information

e_SS_SM_DATA_RESET_MODE_NONE : Do not initialize
e_SS_SM_DATA_RESET_MODE_USER : User data initialization
e_SS_SM_DATA_RESET_MODE_FACTORY : Factory initialization

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusBadConnection	Socket connection error
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget Method

See also:

SetProgUpdateStateToSystemManager

E_CWORD33_Status SetNextWakeupTypeToSystemManager (ESMNextWakeupType *f_wakeupType*)

Summary

Set the next wakeup type.

Parameters:

in	*f_wakeupType*	ESMNextWakeupType - Wakeup type

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusFail	Some sort of error occurred

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

None

E_CWORD33_Status SetProgUpdateStateToSystemManager (SMProgUpdateState *progUpdateState*)

Summary

API to record the program update state in the non-volatile information.

Parameters:

in	*updateState*	SMProgUpdateState - Program update status

SS_SM_PROG_UPDATE_STATE_UPDATED : Program update
SS_SM_PROG_UPDATE_STATE_MAP_UPDATED : Map update
SS_SM_PROG_UPDATE_STATE_MAPDIFF_UPDATED : Map difference updating

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusBadConnection	Socket connection error
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

SetDataResetModeToSystemManager, Get_CWORD33_OnStartExtInfo, Get_CWORD33_OnStopExtInfo

E_CWORD33_Status SetWakeupOrderToSystemManager (std::string *f_orderName*)

Summary

API to save the boot order.

Parameters:

in	*f_orderName*	std::string - Boot order defined name

Boot order defined name, it is 0-32byte.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusBadConnection	Socket connection error
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

Dispatcher for the System Manager is generated, and Availability of the service is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

None

Variable Documentation

const long iProcess_DEFAULT_PROCESS_FLAGS

Initial value:= POSIX_SPAWN_SETSCHEDULER
| POSIX_SPAWN_SETSCHEDPARAM

Task_manager

struct _T_PRIM_PRM
struct T_PRIM_MONITOR_PRM
struct PRIM_SHAREDATA_TBL
struct PRIM_FORK_TBL
struct PRIM_EXFUNC_TBL
struct PRIM_BACKUPCHK_TBL
struct PRIM_ACCOFF_TBL

struct TSKM_GSTEP_REQ_INFO_t
struct TSKM_GSTEP_t
struct TSKM_GSTEP_CTX_t
struct TSKM_NV_HEADER_t
struct TSKM_NV_FOOTER_t
struct TSKM_NV_BODY_t
struct TSKM_NV_INFO_t
struct TSKM_EV_PRI_REQ_WAKEUP_PRM_t
struct TSKM_EV_PRI_REQ_DOWN_PRM_t
struct TSKM_EV_PRI_REP_WAKEUP_COMP_PRM_t
struct TSKM_EV_PRI_REP_DOWN_COMP_PRM_t
struct TSKM_EV_PRI_REP_CONNECT_PRM_t
struct TSKM_EV_PRI_RES_WAKEUP_PRM_t
struct TSKM_EV_PRI_RES_DOWN_PRM_t
struct TSKM_EV_PRI_EX_RES_DEBUGDUMP_PRM_t
struct TSKM_EV_LCL_CHG_SVC_STATE_PRM_t
struct _TSKM_EVENT_INFO_t
union _TSKM_EVENT_INFO_t.prm
struct TSKM_SVC_ATTR_t
struct TSKM_SVC_CTX_t
struct TSKM_SVCS_CTX_t
struct TSKM_SVC_CTL_t
struct TSKM_SVC_INFO_t
struct TSKM_LOGGING_INFO_t
struct TSKM_ERROR_REBOOT_t
struct TSKM_DATAINIT_t
struct TSKM_WAKEUP_ORDER_t

Macros

#define **INI_SUCCESS** 0 /* */
#define **INI_FALSE** -1 /* */
#define **INI_FD_MAX** 3
#define **INI_INITCOMP_NONE** 0x0000000000000000LLU
#define **INI_INITCOMP_ON_START** 0x0000000000000001LLU /* ON_START */
#define **INI_INITCOMP_NVM_ACCESS** 0x0000000000000002LLU /* NVM ACCESS */
#define **INI_INITCOMP_TEST0** 0x1000000000000000LLU /* TEST */
#define **INI_INITCOMP_TEST1** 0x2000000000000000LLU /* TEST */
#define **INI_INITCOMP_TEST2** 0x4000000000000000LLU /* TEST */
#define **INI_INITCOMP_TEST3** 0x8000000000000000LLU /* TEST */
#define **INI_TERMCOMP_NONE** 0x0000000000000000LLU
#define **INI_TERMCOMP_ACTIVITYMGR** 0x0000000000000001LLU /* ActivityManager */
#define **INI_TERMCOMP_RESIDENT** 0x0000000000000002LLU /* SVC */
#define **INI_TERMCOMP_TRANSIENT** 0x0000000000000004LLU /* SVC */
#define **INI_TERMCOMP_TEST0** 0x1000000000000000LLU /* TEST */
#define **INI_TERMCOMP_TEST1** 0x2000000000000000LLU /* TEST */
#define **INI_TERMCOMP_TEST2** 0x4000000000000000LLU /* TEST */
#define **INI_TERMCOMP_TEST3** 0x8000000000000000LLU /* TEST */
#define **INI_DEBUGDUMP**(args...) _INI_DEBUGDUMP(TRUE, ## args)

```
#define INI_DEBUGDUMP_RAW(args...)  _INI_DEBUGDUMP(FALSE, ## args)
#define SRAM_INIT  (int)0       /*            */
#define SRAM_CHK  (int)1       /*               */
#define SRAM_MAPVER  (int)2       /* Version             */
#define SRAM_MAPVER_CHK  (int)3    /* Version +  */
#define SRAM_SYSTEM_INIT  (int)4   /*  #37 */
#define SRAM_CONFIG_INIT  (int)5   /*  #37 */
#define SRAM_CHK_OK  (int)0     /* SRAM  */
#define SRAM_CHK_CHG  (int)1     /* SRAM */
#define SRAM_CHK_NG  (int)-1     /* SRAM                */
#define PRIM_SHM_NAME_MAX  32            /*  */
#define SYS_RESET  (int32)1     /*    */
#define SYS_ONLY_RESET  (int)2       /*                  #9# */
#define TSKM_ID  RPC_ID
#define TSKM_Init(p_tskmId)  RPC_START_CLIENT(p_tskmId)
#define TSKM_End(tskmId)  RPC_end(tskmId)
#define TSKM_DATA_INIT_FUNC(ServiceName,  ArgName)  E_CWORD33_Status tskm_ ##
    ServiceName ## _data_init(T_SS_SM_START_DataStructType* ArgName)
#define TSKM_CFG_WAIT_SHUTDOWN  10000
#define TSKM_CFG_TOUCH_TIMEOUT  2
#define TSKM_ST_ACCOFF  0x01000000U
#define TSKM_ST_ACCON  0x02000000U
#define TSKM_ST_WAKEUP  0x02010000U
#define TSKM_ST_RUNNING  0x02020000U
#define TSKM_ST_DOWN  0x02040000U
#define TSKM_STATE_MASK0  0xFF000000U
#define TSKM_STATE_MASK1  0xFFFF0000U
#define TSKM_STATE_MASK2  0xFFFFFF00U
#define TSKM_STATE_MASK3  0xFFFFFFFFU
#define TSKM_GSTEP_NONE  0x00000000U
#define TSKM_GSTEP_BUPCHK  0x00000001U
#define TSKM_LSTEP_LAST  0xFFFFFFF0U
#define TSKM_LSTEP_SHM  0xFFFFFFF1U
#define TSKM_LSTEP_BUPCHK  0xFFFFFFF2U
#define TSKM_LSTEP_ALL  0xFFFFFFF3U
#define TSKM_NV_STRUCT_VERSION  "v000"
#define TSKM_SVC_RESERVE_MAX  (8)
#define TSKM_NV_SIZE_ALL  (512)
#define TSKM_NV_SIZE_HEADER  (32)
#define TSKM_NV_SIZE_FOOTER  (32)
#define TSKM_NV_SIZE_BODY  (TSKM_NV_SIZE_ALL - TSKM_NV_SIZE_HEADER -
    TSKM_NV_SIZE_FOOTER)
#define TSKM_EV_BOOTINFO_SIZE  sizeof(T_SS_SM_START_DataStructType)
#define TSKM_EV_EXTBOOTINFO_SIZE  sizeof(T_SS_SM_START_ExtDataStructType)
#define TSKM_EV_DEBUGDUMP_SIZE  4096
#define TSKM_SVC_WAIT_REQ_MAX  (8)
#define TSKM_SVCID_NONE  0x00000000U
#define TSKM_SVCID_TE_RESIDENT  0x00000001U
```

#define **TSKM_SVCID_TE_TRANSIENT** 0x00000002U
#define **TSKM_SVCID_ACTIVITYMGR** 0x00000003U
#define **TSKM_SVCID_SYSVUP** 0x00000004U
#define **TSKM_SVCID__CWORD52_VUP** 0x00000005U
#define **TSKM_SVCID_ROOTFSVUP** 0x00000006U
#define **TSKM_SVCID_VUPPROGUI** 0x00000007U
#define **TSKM_SVCID_NORVUP** 0x00000008U
#define **TSKM_SVCID__CWORD58_** 0x00000009U
#define **TSKM_SVCID_SYSUPDATE** 0x0000000AU
#define **TSKM_SVCID_NANDUPDATE** 0x0000000BU
#define **TSKM_SVCID_BTPHONESRV** 0x0000000CU
#define **TSKM_SVCID_BTPBKSRV** 0x0000000DU
#define **TSKM_SVCID_BTMSGSRV** 0x0000000EU
#define **TSKM_SVCID_DTVVUPSRV** 0x0000000FU
#define **TSKM_SVCID_XMVUPSRV** 0x00000020U
#define **TSKM_ERR_t** int32_t
#define **TSKM_E_OK** 0
#define **TSKM_E_PAR** 1
#define **TSKM_E_STATE** 2
#define **TSKM_E_PERMIT** 3
#define **TSKM_E_ALREADY** 4
#define **TSKM_E_RETRY** 5
#define **TSKM_E_NG** 6
#define **TSKM_BOOL_t** int32_t
#define **TSKM_TRUE** 1
#define **TSKM_FALSE** 0
#define **TSKM_RSV_t** int
#define **TSKM_LOGGING_MSG_STR_SIZE** 64
#define **TSKM_ORDER_NAME_MAX** 32

Typedefs

typedef struct _T_PRIM_PRM **T_PRIM_PRM**
typedef unsigned int **PRI_DWORD**
typedef uint32_t **TSKM_STATE_t**
typedef uint32_t **TSKM_GLOBAL_STEP_t**
typedef uint32_t **TSKM_LOCAL_STEP_t**
typedef struct _TSKM_EVENT_INFO_t **TSKM_EVENT_INFO_t**
typedef uint32_t **TSKM_SVCID_t**

Enumerations

enum **_PRIM_FORK_STEP** { **PRIM_STEPFORK_FIRST** = ***, **PRIM_STEPFORK_SECOND**, **PRIM_STEPFORK_THIRD**, **PRIM_STEPFORK_FOURTH**, **PRIM_STEPFORK_FIFTH**, **PRIM_STEPFORK_SIXTH**, **PRIM_STEPFORK_SEVENTH**, **PRIM_STEPFORK_EIGHTH**, **PRIM_STEPFORK_NINTH**, **PRIM_STEPFORK_TENTH**, **PRIM_STEPFORK_MAX** }
enum **_PRIM_ACCOFF_STEP** { **PRIM_ACCOFF_FIRST** = ***, **PRIM_ACCOFF_SECOND**, **PRIM_ACCOFF_THIRD**, **PRIM_ACCOFF_FOURTH**, **PRIM_ACCOFF_FIFTH**, **PRIM_ACCOFF_MAX** }

enum **TSKM_EVENT_t** { **TSKM_EV_NOP**, **TSKM_EV_PRI_REQ_WAKEUP**,
TSKM_EV_PRI_REQ_DOWN, **TSKM_EV_PRI_REQ_TOUCH**,
TSKM_EV_PRI_REQ_DEBUGDUMP, **TSKM_EV_PRI_REP_LOWMEM**,
TSKM_EV_PRI_REP_WAKEUP_COMP, **TSKM_EV_PRI_REP_DOWN_COMP**,
TSKM_EV_PRI_REP_CONNECT, **TSKM_EV_PRI_REP_DISCONNECT**,
TSKM_EV_PRI_RES_WAKEUP, **TSKM_EV_PRI_RES_DOWN**,
TSKM_EV_PRI_RES_DEBUGDUMP, **TSKM_EV_PRI_REQ_EXIT**, **TSKM_EV_SVC_REP_TERM**,
TSKM_EV_API_REQ_REBOOT, **TSKM_EV_API_REQ_SVC_CTL**,
TSKM_EV_API_REQ_SVC_DISABLE, **TSKM_EV_LCL_REQ_STOP**,
TSKM_EV_LCL_CHG_SVC_STATE, **TSKM_EV_LCL_REQ_SDUMP**,
TSKM_EV_LCL_REP_TIMEOUT, **TSKM_EV_LCL_REP_POLLING**,
TSKM_EV_LCL_REP_LOWMEM, **TSKM_EV_LCL_REQ_TRANS_STEP** }
enum **TSKM_PROTCOL_t** { **TSKM_DATAINIT_REQ** = ***, **TSKM_DATAINIT_RESP**,
TSKM_TRANS_STEP_REQ }
enum **TSKM_SVC_TYPE_t** { **TSKM_SVC_TYPE_NATIVE**, **TSKM_SVC_TYPE_UNKNONW** }
enum **TSKM_SVC_POLICY_t** { **TSKM_SVC_POLICY_TSS**, **TSKM_SVC_POLICY_RR**,
TSKM_SVC_POLICY_FIFO }
enum **TSKM_SVC_LC_t** { **TSKM_SVC_LC_ALWAYS**, **TSKM_SVC_LC_ALWAYS_RECOVERABLE**,
TSKM_SVC_LC_DYNAMIC }
enum **TSKM_SVC_ASSIGN_CPU_t** { **TSKM_SVC_ASSIGN_CPU_AUTO** = ***,
TSKM_SVC_ASSIGN_CPU_0 = ***, **TSKM_SVC_ASSIGN_CPU_1** = *** }
enum **TSKM_SVC_STATE_t** { **TSKM_SVC_DORMANT**, **TSKM_SVC_WAITCONNECT**,
TSKM_SVC_WAKEUP, **TSKM_SVC_RUNNING**, **TSKM_SVC_DOWN**, **TSKM_SVC_FINDOWN**,
TSKM_SVC_DISABLE }
enum **TSKM_SVC_WAIT_STATE_t** { **TSKM_SVC_WAIT_NONE**, **TSKM_SVC_WAIT_TRANSIENT**,
TSKM_SVC_WAIT_BOTH }
enum **TSKM_SVC_CMD_t** { **TSKM_SVC_CMD_NOP**, **TSKM_SVC_CMD_EXEC**,
TSKM_SVC_CMD_ENABLE, **TSKM_SVC_CMD_DISABLE**, **TSKM_SVC_CMD_RESERVE** }
enum **TSKM_BOOT_t** { **TSKM_BOOT_NORMAL**, **TSKM_BOOT_FACTRESET**,
TSKM_BOOT_USERRESET, **TSKM_BOOT_VERSIONUP** }
enum **TSKM_LOGGING_TYPE_NORMAL_t** { **TSKM_LOGGING_TYPE_MODULE_LOGS**,
TSKM_LOGGING_TYPE_GRP_RELAUNCH }
enum **TSKM_ERROR_REBOOT_TYPE_t** { **TSKM_ERROR_REBOOT_NORMAL** }
enum **TSKM_DATAINIT_TYPE_t** { **TSKM_DATAINIT_TYPE_USER** }

Functions

int INI_Main (T_PRIM_PRM *p_prm, int argc, char *argv[])
void INI_ExitStart (void *rsv)
void INI_ExitDone (int status)
void * INI_GetPrivate ()
void * INI_GetHandle ()
int32_t INI_SetMonitorTimeout (uint32_t timeout)
int32_t INI_StepForkComp (uint64_t compId)
int32_t INI_AccOffComp (uint64_t compId)
int32_t INI_GetBootInfo (T_SS_SM_START_DataStructType *info)
int32_t INI_GetExtBootInfo (T_SS_SM_START_ExtDataStructType *info)
int32_t INI_Init (T_PRIM_PRM *p_prm, int argc, char *argv[], int *fdNum, int fdlist[INI_FD_MAX])
BOOL INI_Handler (fd_set *p_fds)

void INI_Term (void)
int32_t INI_SetMonitorState (T_PRIM_MONITOR_PRM *p_prm)
void **_INI_DEBUGDUMP** (BOOL bIsNeedSvcName, PCSTR f_cFormat,...)
E_CWORD33_Status _CWORD33_OnTouch (HANDLE hApp)
TSKM_ERR_t TSKM_SvcCtl (TSKM_SVCID_t svcId, const TSKM_SVC_CTL_t *ctl)
TSKM_ERR_t TSKM_SvcGetInfo (TSKM_SVCID_t svcId, TSKM_SVC_INFO_t *svcInfo)
TSKM_ERR_t TSKM_ErrorReboot (const TSKM_ERROR_REBOOT_t *p_info)
TSKM_ERR_t TSKM_Reboot (const TSKM_RSV_t *p_rsv)
TSKM_ERR_t TSKM_Logging (const TSKM_LOGGING_INFO_t *p_info)
TSKM_ERR_t TSKM_DataInit (HANDLE hApp, const TSKM_DATAINIT_t *p_info)
TSKM_ERR_t TSKM_SetWakeupOrder (const TSKM_WAKEUP_ORDER_t *p_order)
TSKM_ERR_t TSKM_GetExtBootInfo (T_SS_SM_START_ExtDataStructType *p_info)
TSKM_ERR_t tskm_svcsEventHandle (TSKM_SVCS_CTX_t *p_svcs, const TSKM_EVENT_INFO_t *p_inEv, TSKM_EVENT_INFO_t *p_outEv)
TSKM_SVC_CTX_t * tskm_svcsGetSvcBySvcId (TSKM_SVCS_CTX_t *p_svcs, TSKM_SVCID_t svcId)
TSKM_SVC_CTX_t * tskm_svcsGetSvcByPid (TSKM_SVCS_CTX_t *p_svcs, pid_t pid)
TSKM_BOOL_t tskm_svcsIsWaiting (TSKM_SVCS_CTX_t *p_svcs)
TSKM_SVC_WAIT_STATE_t tskm_svcsGetSvcTermWaitState (TSKM_SVCS_CTX_t *p_svcs)
TSKM_ERR_t tskm_svcsSetBootInfo (TSKM_SVCS_CTX_t *p_svcs, T_SS_SM_START_DataStructType *p_info, T_SS_SM_START_ExtDataStructType *p_exInfo)
TSKM_ERR_t tskm_svcsAvtiveSvcTerm (TSKM_SVCS_CTX_t *p_svcs)
TSKM_ERR_t tskm_svcsCallDebugDump (TSKM_SVCS_CTX_t *p_svcs)
TSKM_ERR_t tskm_svcsCallLowMem (TSKM_SVCS_CTX_t *p_svcs)
TSKM_ERR_t tskm_svcEventHandle (TSKM_SVC_CTX_t *p_svc, TSKM_EVENT_INFO_t *p_ev)
TSKM_ERR_t tskm_svcExec (TSKM_SVC_CTX_t *p_svc)
TSKM_ERR_t tskm_svcWakeupRequest (TSKM_SVC_CTX_t *p_svc, TSKM_GSTEP_REQ_INFO_t *p_req)
TSKM_ERR_t tskm_svcDownRequest (TSKM_SVC_CTX_t *p_svc, TSKM_GSTEP_REQ_INFO_t *p_req)
TSKM_ERR_t tskm_svcDisableRequest (TSKM_SVC_CTX_t *p_svc)
TSKM_ERR_t tskm_svcEnableRequest (TSKM_SVC_CTX_t *p_svc)
TSKM_BOOL_t tskm_svcIsCommnicatable (TSKM_SVC_CTX_t *p_svc)
int tskm_initServiceList (TSKM_SVCS_CTX_t *p_svcs, int iFd)
void tskm_initWakeupCtx (TSKM_GSTEP_CTX_t *p_wakeup, BOOL isVupMode)
void tskm_initDownCtx (TSKM_GSTEP_CTX_t *p_down, BOOL isVupMode)

Detailed Description

Class Documentation

struct T_PRIM_MONITOR_PRM

Class Members:

BOOL	bIsRun	
uint32_t	timeout	

struct PRIM_SHAREDATA_TBL

Class Members:

char	shmName[PRIM_SHM_NAME_MAX]	
int32_t	size	

struct PRIM_ACCOFF_TBL

Class Members:

uint32_t	localStep	
uint16_t	pno	
uint8_t	rsv[2]	

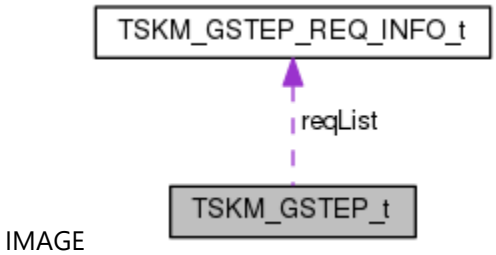
struct TSKM_GSTEP_REQ_INFO_t

Class Members:

TSKM_LOCAL_STEP_t	localStep	
TSKM_SVCID_t	svcId	

struct TSKM_GSTEP_t

Collaboration diagram for TSKM_GSTEP_t:



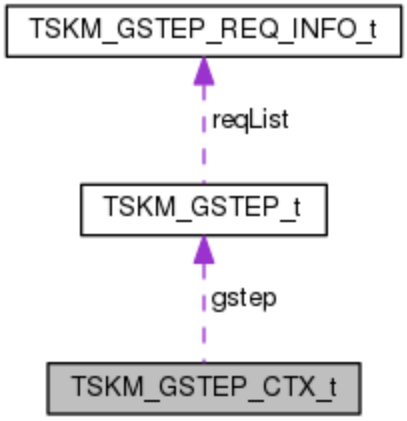
IMAGE

Class Members:

TSKM_SVCID_t *	execSvcIdList	
uint32_t	execSvcNum	
uint32_t	gstepId	
uint64_t	nextTransCond	
TSKM_GSTEP_REQ_INFO_t *	reqList	
uint32_t	reqNum	

struct TSKM_GSTEP_CTX_t

Collaboration diagram for TSKM_GSTEP_CTX_t:



IMAGE

Class Members:

uint64_t	compState	
TSKM_GSTEP_t *	gstep	
uint32_t	gstepIdx	
uint32_t	gstepNum	

struct TSKM_NV_HEADER_t

Class Members:

char	version[5]	

struct TSKM_NV_FOOTER_t

Class Members:

uint32_t	checkSum	

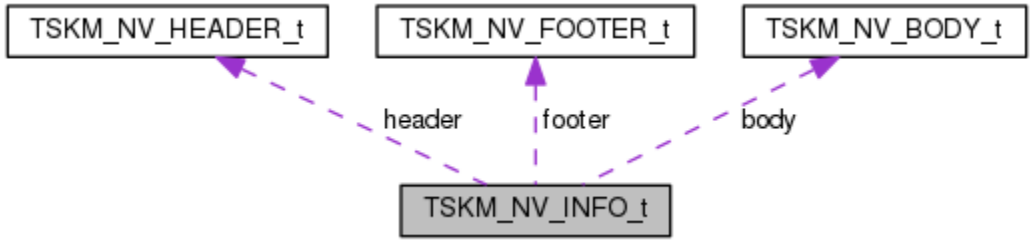
struct TSKM_NV_BODY_t

Class Members:

uint8_t	rsvSvcNum	
TSKM_SVCID_t	rsvSvcs[TSKM_SVC_RESERVE_MAX]	

struct TSKM_NV_INFO_t

Collaboration diagram for TSKM_NV_INFO_t:



IMAGE

Class Members:

TSKM_NV_BODY_t	body	
TSKM_NV_FOOTER_t	footer	
TSKM_NV_HEADER_t	header	
uint8_t	rsvBody[TSKM_NV_SIZE_BODY-sizeof(TSKM_NV_BODY_t)]	
uint8_t	rsvFooter[TSKM_NV_SIZE_FOOTER-sizeof(TSKM_NV_FOOTER_t)]	
uint8_t	rsvHeader[TSKM_NV_SIZE_HEADER-sizeof(TSKM_NV_HEADER_t)]	

struct TSKM_EV_PRI_REQ_WAKEUP_PRM_t

Class Members:

char	bootInfo[TSKM_EV_BOOTINFO_SIZE]	
char	extBootInfo[TSKM_EV_EXTBOOTINFO_SIZE]	
TSKM_BOOL_t	isDynamic	
TSKM_LOCAL_STEP_t	localStep	
TSKM_SVCID_t	svcId	

struct TSKM_EV_PRI_REQ_DOWN_PRM_t

Class Members:

TSKM_LOCAL_STEP_t	localStep	

struct TSKM_EV_PRI_REP_WAKEUP_COMP_PRM_t

Class Members:

uint64_t	compId	

struct TSKM_EV_PRI_REP_DOWN_COMP_PRM_t

Class Members:

uint64_t	compId	

struct TSKM_EV_PRI_REP_CONNECT_PRM_t

Class Members:

int	connFd	

struct TSKM_EV_PRI_RES_WAKEUP_PRM_t

Class Members:

TSKM_BOOL_t	isLast	
TSKM_BOOL_t	isShmDone	
TSKM_BOOL_t	isStepDone	

struct TSKM_EV_PRI_RES_DOWN_PRM_t

Class Members:

TSKM_BOOL_t	isLast	

struct TSKM_EV_PRI_EX_RES_DEBUGDUMP_PRM_t

Class Members:

char	dumpMsg[TSKM_EV_DEBUGDUMP_SIZE]	

struct TSKM_EV_LCL_CHG_SVC_STATE_PRM_t

Class Members:

TSKM_SVCID_t	svcId	

union _TSKM_EVENT_INFO_t.prm

Class Members:

TSKM_EV_LCL_CHG_SVC_STATE_PRM_t	chgSvc	
TSKM_EV_PRI_REP_CONNECT_PRM_t	repConnect	
TSKM_EV_PRI_REP_DOWN_COMP_PRM_t	repDownComp	
TSKM_EV_PRI_REP_WAKEUP_COMP_PRM_t	repWakeupComp	
TSKM_EV_PRI_REQ_DOWN_PRM_t	reqDown	
TSKM_EV_PRI_REQ_WAKEUP_PRM_t	reqWakeup	
TSKM_EV_PRI_RES_DOWN_PRM_t	resDown	
TSKM_EV_PRI_RES_WAKEUP_PRM_t	resWakeup	

struct TSKM_SVC_ATTR_t

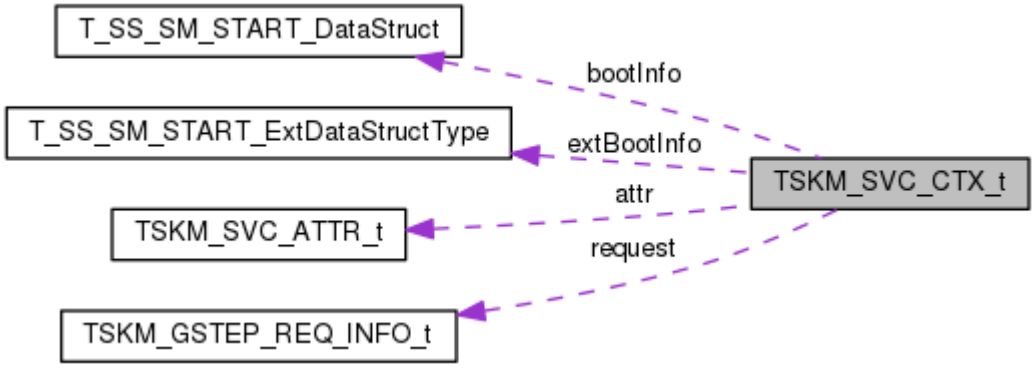
Class Members:

char **	args	
TSKM_SVC_ASSIGN_CPU_t	cpuAssign	
uint32_t	cpuLimit	
TSKM_SVC_LC_t	lifeCycle	
uint32_t	memLimit	
const char *	name	
const char *	path	
TSKM_SVC_POLICY_t	policy	
uint32_t	prio	

uint32_t	retryCnt	
uint32_t	runtimeLimit	
TSKM_BOOL_t	shotdownWait	
gid_t *	subgidList	
uint32_t	subgidNum	
TSKM_SVCID_t	svcId	
TSKM_SVC_TYPE_t	type	
const char *	user	

struct TSKM_SVC_CTX_t

Collaboration diagram for TSKM_SVC_CTX_t:



IMAGE

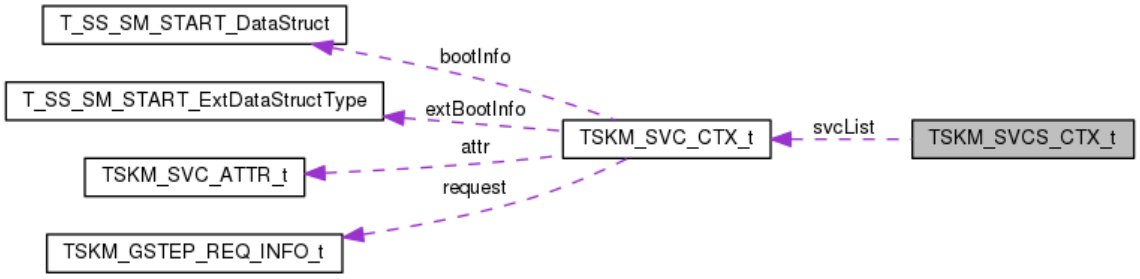
Class Members:

TSKM_SVC_ATTR_t *	attr	
T_SS_SM_START_DataStructType	bootInfo	
int	connFd	
uint32_t	errTermCnt	
T_SS_SM_START_ExtDataStructType	extBootInfo	
int	iFd	
TSKM_BOOL_t	isAvailable	
TSKM_BOOL_t	isShmDone	
TSKM_BOOL_t	isStepDone	
pid_t	pid	
TSKM_GSTEP_REQ_INFO_t	request[TSKM_SVC_WAIT_REQ_MAX]	
TSKM_SVC_STATE_t	state	
uint32_t	waitReqCnt	
uint32_t	waitResCnt	
uint32_t	watchCnt	

struct TSKM_SVCS_CTX_t

Collaboration diagram for TSKM_SVCS_CTX_t:

IMAGE



Class Members:

TSKM_SVC_CTX_t *	svcList	
uint32_t	svcNum	

struct TSKM_SVC_CTL_t

Class Members:

TSKM_SVC_CMD_t	cmd	

struct TSKM_SVC_INFO_t

Class Members:

TSKM_BOOL_t	isExecDisable	
TSKM_SVCID_t	svcId	

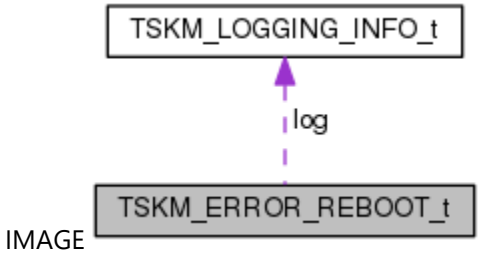
struct TSKM_LOGGING_INFO_t

Class Members:

char	messageStr[TSKM_LOGGING_MSG_STR_SIZE]	
TSKM_LOGGING_TYPE_NORMAL_t	type	

struct TSKM_ERROR_REBOOT_t

Collaboration diagram for TSKM_ERROR_REBOOT_t:



IMAGE

Class Members:

TSKM_LOGGING_INFO_t	log	
TSKM_ERROR_REBOOT_TYPE_t	type	

struct TSKM_WAKEUP_ORDER_t

Class Members:

char	orderName[TSKM_ORDER_NAME_MAX]	

Macro Definition Documentation

#define TSKM_DATA_INIT_FUNC(ServiceName, ArgName) E_CWORD33_Status tskm_ ## ServiceName ## _data_init(T_SS_SM_START_DataStructType* ArgName)

Summary

Defines callback functions to initialize data.

Parameters:

in	*serviceName*	const char* - Service name
in	*argName*	const T_SS_SM_START_DataStructType* - Argument name 1 typedef struct T_SS_SM_START_DataStruct{ 2 EPWR_WAKEUP_FACTOR_TYPE startupReason; 3 BOOL isUserModeOn; 4 ESMDataResetModeInfo dataResetMode; 5 EPWR_SC_SECURITY_STATUS securityStatus; 6 EPWR_SC_WAKEUP_TYPE wakeupType; 7 ESMDramBackupStatus dramBackupStatus; 8 ESMResetStatus resetStatus; 9 UI_32 resetCount; 10 } T_SS_SM_START_DataStructType;

Return values:

e_CWORD33_StatusOK	Normal
e_CWORD33_StatusFail	Check error

Preconditions

None

Change of the internal state

The internal state is not changed.

Classification

Public

See also:

None

#define TSKM_End(tskmId) RPC_end(tskmId)

Summary:

Releases the resource created at TSKM_Init.

Parameters:

in	*ID*	UINT32 - ID gotten at TSKM_Init()

Return values:

None	

Preconditions:

TSKM_Init must be used and initialized

Changes of the internal stat:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

TSKM_Init

#define TSKM_Init(p_tskmId) RPC_START_CLIENT(p_tskmId)

Summary:

Creates the resource to communicate with Task_Manager.

Parameters:

out	**pID*	UINT32 - Unique ID of the automatically allocated program

Return values:

None	

Preconditions:

None

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

TSKM_End

Function Documentation

E_CWORD33_Status _CWORD33_OnTouch (HANDLE *hApp*)

Summary:

Callback to notify process use

Parameters:

in	*hApp*	HANDLE - Application handle

Return values:

e_CWORD33_StatusOK	Succeeded
Other-than-e_CWORD33_StatusOK	Failed (Error type depends on the implementation at user side)

Preconditions:

Must set inside running information structure in calling INI_Main or INI_Init

Changes of the internal state:

The internal state is not changed

Classification:

Public

See also:

None

int32_t INI_AccOffComp (uint64_t *compId*)

Summary:

Notifies end event completion notification to TaskManager.

Parameters:

in	*compId*	uint64_t - The comparison ID for phase running (Designates INI_TERMCOMP_[a-zA-Z_]+ defined at INI_API.hpp)

The ID described at INI_API.hpp

1 - INI_TERMCOMP_NONE : None
2 - INI_TERMCOMP_ACTIVITYMGR : For ActivityManager end

3 - INI_TERMCOMP_RESIDENT : For resident service end
4 - INI_TERMCOMP_TRANSIENT : For non-resident service end

Return values:

INI_SUCCESS	Succeeded
INI_FALSE	Failed

Preconditions:

INI_Main or INI_Init must be called

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

Supplement:

next_trans_condition is the element to designate transition conditions to next phase. The transition to next phase is done when the following two conditions are met.
When the description of next_trans_condition element is omitted, transition is done only by condition 1.

1. All of the phase end request processing set by request element is completed. (The phase end request processing is executed by primary library asynchronously)
2. The completion notification of the event designated by next_trans_condition element is completed. Waiting conditions for event completion notification INI_INITCOMP_[A-Z]+ stipulated at INI_API.hpp is set in cond property designated at next_trans_condition element.

AND condition can be set by being INI_INITCOMP_AAAA| INI_INITCOMP_BBBB when conditions are some.
Event completion notification can be issued by using INI_ AccOffComp.

See also:

INI_Main, INI_Init

void INI_ExitDone (int *status*)

Summary:

Notifies the completion of process end processing.

Parameters:

in	*status*	int - Unused (This is for future extension. 0 must be designated.)

Return values:

None	

Preconditions:

None

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Main, INI_Init, INI_ExitStart

void INI_ExitStart (void * *rsv*)

Summary:

Starts the end processing of process.

Parameters:

in	*rsv*	void* - The pointer to reservation parameters (NULL must be appointed)

Return values:

None	

Preconditions:

None

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

Supplemtent:

local_step is the information defined in default_wakeup.xml/default_shutdown.xml as the definition of running order / end order.
local_step defined in default_shutdown.xml has the following types.
1-9 : Designates local phase numbers (the phase numbers of end processing)
last : This designates local phase numbers to be last. When there is the local phase processing not to be executed, this executes all.
all : The same as last.

See also:

INI_Main, INI_Init, INI_ExitDone

int32_t INI_GetBootInfo (T_SS_SM_START_DataStructType * *info*)

Summary:

Gets boot information.

Parameters:

in	*info*	T_SS_SM_START_DataStructType * - The pointer to boot information structure

T_SS_SM_START_DataStructType structure

```
typedef struct T_SS_SM_START_DataStruct{
    EPWR_WAKEUP_FACTOR_TYPE   startupReason;    /* Startup reason               */
    BOOL                      isUserModeOn;     /* User mode ON/OFF             */
    ESMDataResetModeInfo      dataResetMode;    /* Data reset mode              */
    EPWR_SC_SECURITY_STATUS   securityStatus;   /* Security state               */
    EPWR_SC_WAKEUP_TYPE       wakeupType;       /* Battery disconnection state  */
    ESMDramBackupStatus       dramBackupStatus; /* Backup memory area state */
    ESMResetStatus            resetStatus;      /* Reset state                  */
    UI_32                     resetCount;       /* Abnormal reboot number       */
} T_SS_SM_START_DataStructType;

Startup reason type:
    - epswfIGN_ACC     : Ignition ACC (Currently, Fixed)
User mode ON/OFF:
    - TRUE             : User mode ON     (Currently, Fixed)
Data reset mode:
    - e_SS_SM_DATA_RESET_NONE: Not-initialized
    - e_SS_SM_DATA_RESET_USER: User data initialization
    - e_SS_SM_DATA_RESET_FACTORY: Factory initialization
Security state:
    - epsssUNLOCK      : Unlock state
    - epsssLOCK        : Lock state
Battery disconnection state:
    - epsstWARMSTART   : Battery disconnection does not occur
    - epsstCOLDSTART   : Battery disconnection occurs
Backup memory area state:
    - e_SS_SM_DRAM_BACKUP_OK: Backup OK of Backup memory area
    - e_SS_SM_DRAM_BACKUP_NG: Backup NG of Backup memory area
Reset state:
    - e_SS_SM_RESET_STATUS_UNSET: Information unsetting(Not decided yet)
    - e_SS_SM_RESET_STATUS_NONE : Normal
    - e_SS_SM_RESET_STATUS_NG   : Abnormality occurred (Abnormal RESET or booting after service abnormal end)
    - e_SS_SM_RESET_STATUS_IMMEDIATE : Immediate RESET occurred
```

Return values:

INI_SUCCESS	Succeeded
INI_FALSE	Failed

Preconditions:

INI_Main or INI_Init must be called. (Process running must be completed)

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Main, INI_Init

int32_t INI_GetExtBootInfo (T_SS_SM_START_ExtDataStructType * *info*)

Summary:

Gets boot extended information.

Parameters:

in	*info*	T_SS_SM_START_ExtDataStructType * - The pointer to boot extended information structure

T_SS_SM_START_ExtDataStructType structure

```
#define SS_SM_START_EXT_INFO_SIZE 64
#define SS_SM_START_EXT_BODY_SIZE ( sizeof(BOOL)
                                  + sizeof(EMRelaunchStatus)
                                  + sizeof(BOOL)
                                  + sizeof(BOOL)
                                 )
define SS_SM_START_EXT_RSV_SIZE  SS_SM_START_EXT_INFO_SIZE
                               - SS_SM_START_EXT_BODY_SIZE
typedef struct {
    BOOL                  isProgUpdated;     /* Program update information       */
   EMRelaunchStatus relaunchStatus;           /* Own-service Relaunch state */
   BOOL           isMapUpdated;              /* Map update information             */
   BOOL           isMapDiffUpdated;          /* Map difference update information        */
    uint8_t         reserved[SS_SM_START_EXT_RSV_SIZE]; /* Reserve area     */
} T_SS_SM_START_ExtDataStructType;

Program update information
    - TRUE  : Program update
    - FALSE : No Program update
Own-service Relaunch state:
    - e_SS_SM_RELAUNCH_STATUS_NONE : Relaunch does not occur (Initial boot)
    - e_SS_SM_RELAUNCH_STATUS_SAFE : Normal Relaunch
    - e_SS_SM_RELAUNCH_STATUS_ERR  : Abnormal Relaunch
Map update information:
    - TRUE  : Map update
    - FALSE : No Map update
Map difference update information
```

28 - TRUE : Map difference update
29 - FALSE : No Map difference update

Return values:

INI_SUCCESS	Succeeded
INI_FALSE	Failed

Preconditions:

INI_Main or INI_Init must be called. (Process running must be completed.)

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Main, INI_Init

void* INI_GetHandle ()

Summary:

Gets the application handle of NSFW.

Parameters:

None	

Return values:

void(HANDLE)*	The application handle of NSFW

Preconditions:

INI_Main or INI_Init must be called and the application handle of NSFW must be generated.

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Main, INI_Init

void* INI_GetPrivate ()

Summary:

Gets private data.

Parameters:

None	

Return values:

*void**	The pointer to user private data

Preconditions:

INI_Main or INI_Init must be called and user private data must be set.

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Main, INI_Init

BOOL INI_Handler (fd_set * *p_fds*)

Summary:

Processes the reception event of primary library.
(This is used when implementing main thread at service side)

Parameters:

in	*p_fds*	fd_set* - fd set varied and detected by select()

Return values:

TRUE	Service running (Continuing)
FALSE	The process end processing of primary library is completed

Preconditions:

Main thread must be run at INI_Init()

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Init

int32_t INI_Init (T_PRIM_PRM * *p_prm*, int *argc*, char * *argv*[], int * *fdNum*, int *fdlist*[INI_FD_MAX])

Summary:

Initializes primary library.
(This is used when implementing main thread at service side)

Parameters:

in	*p_prm*	T_PRIM_PRM - The pointer to parameter table of primary library
in	*argc*	int - The number of command parameters (The parameters passed to main functions must be set without processing)
in	*argv*	char* - Command parameters (The parameters passed to main functions must be set without processing)
out	*fdNum*	int * - The number of element stored in fdlist (INI_FD_MAX:3)
out	*fdlist[INI_FD_MAX]*	int - The array of FD (File Descriptor) for event reception which primary library waits

Shared memory creation table
In process running, register the shared memory to create.
This creates the shared memory to be registered in process running and releases it in process end.
This must terminate by adding {\0} into member shmName of table termination.
It is able to access to the shared memory to have been created by shm_open.

```
1 typedef struct {
2    char      shmName[32];              /* Shared memory name          */
3    uint32_t  size;                  /* Shared memory size (BYTE) */
4 }PRIM_SHAREDATA_TBL;
```

Running extended function table / End extended function table
Registers callback functions to be called in process start and exit.
This must terminate by adding NULL into member func of table termination.
Parameters prm of callback functions are User private information and the pointer designated by prm is passed as input parameters [in].
Local phase numbers callback func by the designated phase numbers.

```
1 typedef struct {
2    void    (* func)( void* prm);          /* Callback functions for function extension          */
3    void*    prm;                         /* The pointer to the parameters of Callback function func  */
4    uint8_t  localStep;                     /* Local phase numbers                        */
5    uint8_t  rsv[3];                       /* Reserve for alignment              */
6 }PRIM_EXFUNC_TBL;
```

Parameter table of primary library

For _CWORD33_OnInitialization/_CWORD33_OnDestroy/_CWORD33_OnDebugDump/_CWORD33_OnTouch/_CWORD33_OnLowMemory, check the corresponding API specification.

It is able to get User private data by INI_GetPrivate().

```
1 typedef struct _T_PRIM_PRM{
2    PCSTR                    name;          /* Application name                        */
3    const PRIM_SHAREDATA_TBL*  shmTbl;         /* The pointer to shared memory creation table       */
4    const PRIM_EXFUNC_TBL*     wakeupExFuncTbl; /* The pointer to running extended function table
*/
5    const PRIM_EXFUNC_TBL*     downExFuncTbl;   /* The pointer to end extended function table
*/
6    E_CWORD33_Status (*onInit)(HANDLE hApp);         /* _CWORD33_OnInitialization callback function
setting */
7    E_CWORD33_Status (*onDestory)(HANDLE hApp);      /* _CWORD33_OnDestroy callback function
setting       */
8    E_CWORD33_Status (*onDebugDump)(HANDLE hApp);    /* _CWORD33_OnDebugDump Callback
function setting     */
9    E_CWORD33_Status (*onTouch)(HANDLE hApp);        /* _CWORD33_OnTouch callback function
setting         */
10   E_CWORD33_Status (*onLowMem)(HANDLE hApp);       /* _CWORD33_OnLowMemory callback
function setting     */
11    void* priv;                          /* User private data */
12 }T_PRIM_PRM;
```

Return values:

INI_SUCCESS	Succeeded
INI_FALSE	Failed

Preconditions:

None

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Main

int INI_Main (T_PRIM_PRM * *p_prm*, int *argc*, char * *argv*[])

Summary:

Runs the main thread of primary library.

Parameters:

in	*p_prm*	T_PRIM_PRM - The pointer to the parameter table of primary library
in	*argc*	int - The number of command parameters (The parameters passed to main functions must be set without processing.)
in	*argv*	char* - Command parameters (The parameters passed to main functions must be set without processing.)

Shared memory creation table

In process running, register the shared memory to create.

This creates the shared memory to be registered in process running and releases it in process end.

This must terminate by adding {\0} into member shmName of table termination.

It is able to access to the shared memory to have been created by shm_open.

```
1 typedef struct {
2    char      shmName[32];              /* Shared memory name          */
3    uint32_t  size;                  /* Shared memory size (BYTE) */
4 }PRIM_SHAREDATA_TBL;
```

Running extended function table / End extended function table

Registers callback functions to be called in process start and exit.

This must terminate by adding NULL into member func of table termination.

Parameters prm of callback functions are User private information and the pointer designated by prm is passed as input parameters [in].

Local phase numbers callback func by the designated phase numbers.

```
1 typedef struct {
2    void    (* func)( void* prm);          /* Callback functions for function extension         */
3    void*   prm;                    /* The pointer to the parameters of Callback function func  */
4    uint8_t  localStep;               /* Local phase numbers                  */
5    uint8_t  rsv[3];                /* Reserve for alignment            */
6 }PRIM_EXFUNC_TBL;
```

Parameter table of primary library
For
_CWORD33_OnInitialization/_CWORD33_OnDestroy/_CWORD33_OnDebugDump/_CWORD33_OnTouch/_CWORD33_OnLowMemory, check the corresponding API specification.
It is able to get User private data by INI_GetPrivate().

```
typedef struct _T_PRIM_PRM{
    PCSTR                   name;           /* Application name                    */
    const PRIM_SHAREDATA_TBL*  shmTbl;          /* The pointer to shared memory creation table       */
    const PRIM_EXFUNC_TBL*     wakeupExFuncTbl; /* The pointer to running extended function table
*/
    const PRIM_EXFUNC_TBL*     downExFuncTbl;   /* The pointer to end extended function table
*/
    E_CWORD33_Status (*onInit)(HANDLE hApp);         /* _CWORD33_OnInitialization callback function
setting */
    E_CWORD33_Status (*onDestory)(HANDLE hApp);      /* _CWORD33_OnDestroy callback function
setting       */
    E_CWORD33_Status (*onDebugDump)(HANDLE hApp);    /* _CWORD33_OnDebugDump Callback
function setting    */
    E_CWORD33_Status (*onTouch)(HANDLE hApp);        /* _CWORD33_OnTouch callback function
setting         */
    E_CWORD33_Status (*onLowMem)(HANDLE hApp);       /* _CWORD33_OnLowMemory callback
function setting     */
    void* priv;                         /* User private data */
}T_PRIM_PRM;
```

Return values:

INI_SUCCESS	Normal end
INI_FALSE	Abnormal end

Preconditions:

None

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Method only

See also:

INI_Init

int32_t INI_SetMonitorState (T_PRIM_MONITOR_PRM * *p_prm*)

Summary:

Sets abnormality monitoring state.
(This is used when implementing main thread at service side)

Parameters:

in	*p_prm*	T_PRIM_MONITOR_PRM - The pointer to the parameters structure for monitoring

T_PRIM_MONITOR_PRM structure

```
1 typedef struct {
2    BOOL       blsRun;    /* Monitoring state (TRUE:RUNFALSE:SLEEP) */
3    uint32_t   timeout;   /* Monitoring timeout time (sec)       */
4 }PRIM_SHAREDATA_TBL;
```

Return values:

INI_SUCCESS	Succeeded
INI_FALSE	Failed

Preconditions:

Main thread must be run at INI_ Init ()

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Init

int32_t INI_SetMonitorTimeout (uint32_t *timeout*)

Summary:

Requests the change for abnormality monitoring timeout time to Task Manager.
(This is used when main thread is not implemented at service side)

Parameters:

in	*timeout*	uint32_t - Abnormality monitoring timeout time (sec)(When 0 is designated, the service must be out of abnormality monitoring object.)

Return values:

INI_SUCCESS	Succeeded

Preconditions:

Main thread must be run at INI_Main()

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Main

int32_t INI_StepForkComp (uint64_t *compId*)

Summary:

Notifies the completion of running event to TaskManager.

Parameters:

in	*compId*	uint64_t - The comparison ID for phase running (Designates INI_INITCOMP_[a-zA-Z_]+ to define at INI_API.hpp)

The ID descripted at INI_API.hpp

1 - INI_INITCOMP_NONE: None
2 - INI_INITCOMP_ON_START: For ON Start
3 - INI_INITCOMP_NVM_ACCESS: For NVM access

Return values:

INI_SUCCESS	Succeeded
INI_FALSE	Failed

Preconditions:

INI_Main or INI_Init must be called.

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

Supplement:

next_trans_condition is the element to designate transition conditions to next phase. The transition to next phase is done when the following two conditions are met.
When the description of next_trans_condition element is omitted, transition is done only by condition 1.

1. All of the phase running request processing set by request element is completed.
   (The phase running request processing is executed by primary library asynchronously)
2. The completion notification of the event designated by next_trans_condition element is completed.

Waiting conditions for event completion notification INI_INITCOMP_[A-Z]+ stipulated at INI_API.hpp is set in cond property designated at next_trans_condition element.

AND condition can be set by being INI_INITCOMP_AAAA| INI_INITCOMP_BBBB when conditions are some.

Event completion notification can be issued by using INI_StepForkComp.

See also:

INI_Main, INI_Init

void INI_Term (void)

Summary:

Releases the resource of primary library.
(This is used when implementing main thread at service side)

Parameters:

None	

Return values:

None	

Preconditions:

Main thread must be run at INI_Init()

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

INI_Init, INI_Handler

TSKM_ERR_t TSKM_DataInit (HANDLE *hApp*, const TSKM_DATAINIT_t * *p_info*)

Summary:

Does the initialization process of service data.

Parameters:

in	*hApp*	HANDLE - Application handle
in	*p_info*	TSKM_DATAINIT_t * - Data initialization information

		1 typedef struct { 2 TSKM_DATAINIT_TYPE_t type; 3 E_CWORD33_Status (*onCompInit)(HANDLE hApp); 4 } TSKM_DATAINIT_t; 5 type : Data initialization type 6 TSKM_DATAINIT_TYPE_USER : User data initialization 7 onCompInit : Initialization complete Callback

Return values:

TSKM_E_OK	Succeeded
TSKM_E_PAR	Parameter error
TSKM_E_NG	Error

Preconditions:

TSKM_Init must be used and initialized

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Method only

See also:

TSKM_DATA_INIT_FUNC

TSKM_ERR_t TSKM_ErrorReboot (const TSKM_ERROR_REBOOT_t * *p_info*)

Summary:

Let system do abnormal reboot.

Parameters:

in	*p_info*	TSKM_ERROR_REBOOT_t* - REBOOT information
		1 typedef struct {
		2 TSKM_ERROR_REBOOT_TYPE_t type;
		3 TSKM_LOGGING_ INFO_t log;
		4 } TSKM_ERROR_REBOOT_t;
		5 type
		6 TSKM_ERROR_REBOOT_NORMAL : Abnormal detection
		7 log : Log saving information
		8 typedef struct {
		9 char messageStr[TSKM_LOGGING_MSG_STR_SIZE];
		10 } TSKM_LOGGING_INFO_ t;
		11 messageStr : Abnormal message character string

Return values:

TSKM_E_OK	Succeeded
TSKM_E_PAR	Parameter error

Preconditions:

TSKM_Init must be used and initialized

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

None

TSKM_ERR_t TSKM_GetExtBootInfo (T_SS_SM_START_ExtDataStructType * *p_info*)

Summary

Gets boot extended information.

Parameters:

in	*p_info*	T_SS_SM_START_ExtDataStructType * - Boot extended information 1 typedef struct { 2 BOOL isProgUpdated; 3 EMRelaunchStatus relaunchStatus; 4 BOOL isMapUpdated; 5 BOOL isMapDiffUpdated; 6 uint8_t reserved[]; 7 }T_SS_SM_START_ExtDataStructType;

Return values:

TSKM_E_OK	Normal end
TSKM_E_PAR	Parameter error

Preconditions

None

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

TSKM_DATA_INIT_FUNC

void tskm_initDownCtx (TSKM_GSTEP_CTX_t * *p_down*, BOOL *isVupMode*)

Summary

Get state of down step.

Parameters:

out	*p_down*	p_down - pointer of TSKM_GSTEP_CTX_t

p_wakeup TSKM_GSTEP_CTX_t*

```
1 typedef struct {
2   uint32_t gstepIdx;          // step Index
3   uint32_t gstepNum;          // number of step
4   TSKM_GSTEP_t* gstep;        // step start/stop info
5   uint64_t compState;
6 } TSKM_GSTEP_CTX_t;
```

Parameters:

in	*isVupMode*	isVupMode - version up mode flag

isVupMode BOOL TRUE version up mode FALSE not versoin up mode

Return values:

int tskm_initServiceList (TSKM_SVCS_CTX_t * *p_svcs*, int *iFd*)

Summary

init started services list

Parameters:

out	*p_svcs*	p_svcs - pointer of service list

p_svcs TSKM_SVCS_CTX_t*

```
1 typedef struct {
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;  // services list pointer
4 } TSKM_SVCS_CTX_t;
```

Parameters:

in	*iFd*	iFd - the file descriptor return by inotify_init1().

p_svcs int

Return values:

0	Success
-1	Failed

Preconditions

called in the ctxCreate()

Change of the internal state

The internal state is not changed.

Classification

public

Type

sync only

See also:

None

void tskm_initWakeupCtx (TSKM_GSTEP_CTX_t * *p_wakeup*, BOOL *isVupMode*)

Summary

Get state of started step.

Parameters:

out	*p_wakeup*	p_wakeup - pointer of TSKM_GSTEP_CTX_t

p_wakeup TSKM_GSTEP_CTX_t*

```
1 typedef struct {
2   uint32_t gstepIdx;          // step Index
3   uint32_t gstepNum;          // number of step
4   TSKM_GSTEP_t* gstep;        // step start/stop info
5   uint64_t compState;
6 } TSKM_GSTEP_CTX_t;
```

Parameters:

in	*isVupMode*	isVupMode - version up flag

isVupMode BOOL TRUE version up mode FALSE not versoin up mode

Return values:

void	

Preconditions

called in ctxCreate()

Change of the internal state

The internal state is not changed.

Classification

public

Type

sync only

See also:

None

TSKM_ERR_t TSKM_Logging (const TSKM_LOGGING_INFO_t * *p_info*)

Summary:

Stores LOG in internal non-volatile area.

Parameters:

in	*p_info*	TSKM_LOGGING_INFO_t * - Log storing information 1 typedef struct{ 2 TSKM_LOGGING_TYPE_NORMAL_t type; 3 char messageStr[TSKM_LOGGING_MSG_STR_SIZE]; 4 }TSKM_LOGGING_INFO_t; 5 type : Log storing type 6 TSKM_LOGGING_TYPE_MODULE_LOGS: General LOG storing request from service 7 TSKM_LOGGING_TYPE_GRP_RELAUNCH: Log collection by Group Relaunch 8 messageStr : Trouble factors

Return values:

TSKM_E_OK	Succeeded
TSKM_E_PAR	Parameter error

Preconditions:

TSKM_Init must be used and initialized

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

None

TSKM_ERR_t TSKM_Reboot (const TSKM_RSV_t * *p_rsv*)

Summary:

Let system do reboot.

Parameters:

in	*rsv*	TSKM_RSV_t* - Reservation (This must designate NULL)

Return values:

TSKM_E_OK	Succeeded
TSKM_E_PAR	Parameter error

Preconditions:

TSKM_Init must be used and initialized

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync Method

See also:

None

TSKM_ERR_t TSKM_SetWakeupOrder (const TSKM_WAKEUP_ORDER_t * *p_order*)

Summary:

Holds running order.

Parameters:

in	*p_order*	TSKM_WAKEUP_ORDER_t * - Running order information 1 typedef struct { 2 char orderName[TSKM_ORDER_NAME_MAX]; 3 } TSKM_WAKEUP_ORDER_t; 4 orderName: Order name (MAX:31 characters)

Return values:

TSKM_E_OK	Succeeded
TSKM_E_PAR	Parameter error
TSKM_E_NG	Error

Preconditions:

TSKM_Init must be used and initialized

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync Sync

See also:

None

TSKM_ERR_t TSKM_SvcCtl (TSKM_SVCID_t *svcId*, const TSKM_SVC_CTL_t * *ctl*)

Summary:

Run the control for service.

Parameters:

in	*svcid*	TSKM_SVCID_t - Service ID
in	*ctl*	TSKM_SVC_CTL_t* - Command 1 typedef struct{ 2 TSKM_SVC_CMD_t cmd; 3 } TSKM_SVC_CTL_t;

Return values:

TSKM_E_OK	Succeeded
TSKM_E_PAR	Parameter error
TSKM_E_STATE	State error
TSKM_E_NG	Error

Preconditions:

TSKM_Init must be used and initialized

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

None

TSKM_ERR_t tskm_svcDisableRequest (TSKM_SVC_CTX_t * *p_svc*)

Summary

Parameters:

in		

TSKM_ERR_t tskm_svcDownRequest (TSKM_SVC_CTX_t * *p_svc*, TSKM_GSTEP_REQ_INFO_t * *p_req*)

Summary

Send service down request to service.

Parameters:

in		

TSKM_ERR_t tskm_svcEnableRequest (TSKM_SVC_CTX_t * *p_svc*)

Summary

Set service state as start available.

Parameters:

in		

TSKM_ERR_t tskm_svcEventHandle (TSKM_SVC_CTX_t * *p_svc*, TSKM_EVENT_INFO_t * *p_ev*)

Summary

process event.

Parameters:

in	*p_svcs*	p_svcs - pointer of services

p_svcs TSKM_SVCS_CTX_t*

```
1 typedef struct {
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;   // services list pointer
4 } TSKM_SVCS_CTX_t;
```

Parameters:

in		

TSKM_ERR_t tskm_svcExec (TSKM_SVC_CTX_t * *p_svc*)

Summary

start service.

Parameters:

in		

TSKM_ERR_t TSKM_SvcGetInfo (TSKM_SVCID_t *svcId*, TSKM_SVC_INFO_t * *svcInfo*)

Summary:

Gets the information of designated service.

Parameters:

in	*svcid*	TSKM_SVCID_t - Service ID
out	*svcInfo*	TSKM_SVC_INFO_t* - Service information 1 typedef struct { 2 TSKM_SVCID_t svcId; 3 TSKM_BOOL_t isExecDisable; 4 } TSKM_SVC_INFO_t; 5 svcId : Service ID 6 isExecDisable : Service running prohibition state 7 TSKM_TRUE : Running prohibition 8 TSKM_FALSE : Running permission

Return values:

TSKM_E_OK	Succeeded
TSKM_E_PAR	Parameter error

Preconditions:

TSKM_Init must be used and initialized

Changes of the internal state:

The internal state is not changed

Classification:

Public

Type

Sync only

See also:

None

TSKM_BOOL_t tskm_svcIsCommnicatable (TSKM_SVC_CTX_t * *p_svc*)

Summary

Whether service is Commnicatable.

Parameters:

in	*p_svc*	p_svc - pointer of service

p_svc TSKM_SVC_CTX_t*

```
1 typedef struct {
2  TSKM_SVC_ATTR_t *attr;     // attribute of service
3  TSKM_SVC_STATE_t state;    // state of service
4  int iFd;                   // inotifyFd used by touch
5  pid_t pid;                 // PID of service
6  int connFd;                // service communicate fd
7  T_SS_SM_START_DataStructType bootInfo;       // BOOT info
8  T_SS_SM_START_ExtDataStructType extBootInfo;  // extend BOOT info
9  TSKM_BOOL_t isShmDone;     // shared memory init done
10  TSKM_BOOL_t isStepDone;    // step done
11  TSKM_BOOL_t isAvailable;   // Availability flag
12  uint32_t watchCnt;         // service monitor count
13  uint32_t waitResCnt;       // wait response num
14  uint32_t waitReqCnt;       // wait request num
15  TSKM_GSTEP_REQ_INFO_t request[TSKM_SVC_WAIT_REQ_MAX];
16  uint32_t errTermCnt;       // exception terminal count
17 } TSKM_SVC_CTX_t;
```

Return values:

TSKM_BOOL_t	

TSKM_FALSE TSKM_TRUE

Preconditions

Change of the internal state

The internal state is not changed.

Classification

public

Type

method

See also:

None

TSKM_ERR_t tskm_svcsAvtiveSvcTerm (TSKM_SVCS_CTX_t * *p_svcs*)

Summary

Terminal the dynamic services.

Parameters:

in	*p_svcs*	p_svcs - pointer of services

p_svcs TSKM_SVCS_CTX_t*

```
1 typedef struct {
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;   // services list pointer
4 } TSKM_SVCS_CTX_t;
```

Return values:

TSKM_ERR_t	

TSKM_E_OK TSKM_E_NG

Preconditions

none

Change of the internal state

The internal state is not changed.

Classification

pulbic

Type

method only

See also:

tskm_svcDownRequest

TSKM_ERR_t tskm_svcsCallDebugDump (TSKM_SVCS_CTX_t * *p_svcs*)

Summary

Send DebugDump message to services.

Parameters:

in	*p_svcs*	p_svcs - pointer of services

p_svcs TSKM_SVCS_CTX_t*

```
1 typedef struct {
```

```
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;   // services list pointer
4 } TSKM_SVCS_CTX_t;
```

Return values:

TSKM_ERR_t	

TSKM_E_OK TSKM_E_NG

Preconditions

none

Change of the internal state

The internal state is not changed.

Classification

public

Type

method only

See also:

tskm_sockSend

TSKM_ERR_t tskm_svcsCallLowMem (TSKM_SVCS_CTX_t * *p_svcs*)

Summary

Send checking low memory message to services.

Parameters:

in	*p_svcs*	p_svcs - pointer of services

p_svcs TSKM_SVCS_CTX_t*

```
1 typedef struct {
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;   // services list pointer
4 } TSKM_SVCS_CTX_t;
```

Return values:

TSKM_ERR_t	

TSKM_E_OK TSKM_E_NG

Preconditions

none

Change of the internal state

The internal state is not changed.

Classification

public

Type

method only

See also:

tskm_sockSend

TSKM_ERR_t tskm_svcsEventHandle (TSKM_SVCS_CTX_t * *p_svcs*, const TSKM_EVENT_INFO_t * *p_inEv*, TSKM_EVENT_INFO_t * *p_outEv*)

Summary

Parameters:

TSKM_SVC_CTX_t* tskm_svcsGetSvcByPid (TSKM_SVCS_CTX_t * *p_svcs*, pid_t *pid*)

Summary

find service in inputed service list by process id.

Parameters:

in	*p_svcs*	p_svcs - pointer of services

p_svcs TSKM_SVCS_CTX_t*

```
1 typedef struct {
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;   // services list pointer
4 } TSKM_SVCS_CTX_t;
```

Parameters:

in	*pid*	pid - process id

pid pid_t

Return values:

*TSKM_SVC_CTX_t**	service pointer
*TSKM_SVC_CTX_t**	NULL

Preconditions

Change of the internal state

The internal state is not changed.

Classification

public

Type

method only

See also:

getSvcCtxByPid

TSKM_SVC_CTX_t* tskm_svcsGetSvcBySvcId (TSKM_SVCS_CTX_t * *p_svcs*, TSKM_SVCID_t *svcId*)

Summary

find service in inputed service list by service id.

Parameters:

in	*p_svcs*	p_svcs - pointer of services object

p_svcs TSKM_SVCS_CTX_t*

```
1 typedef struct {
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;   // services list pointer
4 } TSKM_SVCS_CTX_t;
```

Parameters:

in	*svcId*	svcId - service Id

svcId uint32_t

Return values:

*TSKM_SVC_CTX_t**	service pointer
*TSKM_SVC_CTX_t**	NULL

Preconditions

none

Change of the internal state

The internal state is not changed.

Classification

public

Type

method only

See also:

getSvcCtxBySvcId

TSKM_SVC_WAIT_STATE_t tskm_svcsGetSvcTermWaitState (TSKM_SVCS_CTX_t * *p_svcs*)

Summary

Get the service which state equanl not equal terminal.

```
1 typedef struct {
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;   // services list pointer
4 } TSKM_SVCS_CTX_t;
```

Return values:

TSKM_SVC_WAIT_STATE_t	

TSKM_SVC_WAIT_STATE_t enum

```
1 typedef enum {
2   TSKM_SVC_WAIT_NONE,       // not waiting state
3   TSKM_SVC_WAIT_TRANSIENT,  // dynamic service terminal waiting state
4   TSKM_SVC_WAIT_BOTH,       // no dynamic/ dyanmic service terminal  waiting state
5 } TSKM_SVC_WAIT_STATE_t;
```

Preconditions

none

Change of the internal state

The internal state is not changed.

Classification

public

Type

method only

See also:

None

TSKM_BOOL_t tskm_svcsIsWaiting (TSKM_SVCS_CTX_t * *p_svcs*)

Summary

Is the service state waiting.

Parameters:

in	*p_svcs*	p_svcs - pointer of services

p_svcs TSKM_SVCS_CTX_t*

```
1 typedef struct {
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;   // services list pointer
4 } TSKM_SVCS_CTX_t;
```

Return values:

TSKM_BOOL_t	

Preconditions

Change of the internal state

The internal state is not changed.

Classification

public

Type

method only

See also:

None

TSKM_ERR_t tskm_svcsSetBootInfo (TSKM_SVCS_CTX_t * *p_svcs*, T_SS_SM_START_DataStructType * *p_info*, T_SS_SM_START_ExtDataStructType * *p_exInfo*)

Summary

initialize all services boot info.

Parameters:

in	*p_svcs*	p_svcs - pointer of services

p_svcs TSKM_SVCS_CTX_t*

```
1 typedef struct {
2   uint32_t svcNum;          // service num
3   TSKM_SVC_CTX_t *svcList;   // services list pointer
```

```
4 } TSKM_SVCS_CTX_t;
```

Parameters:

in	*p_info*	p_info - SS_SM_START '_CWORD33_OnStart'/'evStart' message pointer

p_info T_SS_SM_START_DataStructType*

```
1 Data passed as part of SS_SM_START '_CWORD78_OnStart'/'evStart' message
2 typedef struct T_SS_SM_START_DataStruct {
3    EPWR_WAKEUP_FACTOR_TYPE   startupReason;
4    BOOL                      isUserModeOn;
5    ESMDataResetModeInfo      dataResetMode;
6    EPWR_SC_SECURITY_STATUS   securityStatus;
7    EPWR_SC_WAKEUP_TYPE       wakeupType;
8    ESMDramBackupStatus       dramBackupStatus;
9    ESMResetStatus            resetStatus;
10   UI_32                     errResetCount;
11   T_SS_SM_START_DataStruct
12 } T_SS_SM_START_DataStructType;
```

Parameters:

in	*p_exInfo*	p_exInfo - _CWORD33_OnStart extend info pointer

p_exInfo T_SS_SM_START_ExtDataStructType*

```
1 /// Extended Parameter for _CWORD33_OnStart
2 typedef struct {
3       BOOL            isProgUpdated;
4       EMRelaunchStatus relaunchStatus;
5       BOOL            isMapUpdated;
6       BOOL            isMapDiffUpdated;
7       uint8_t         reserved[SS_SM_START_EXT_RSV_SIZE];
8 } T_SS_SM_START_ExtDataStructType;
```

Return values:

TSKM_E_OK	

TSKM_SVC_WAIT_STATE_t enum

Preconditions

call tskm_dataInitAll() to get boot info

Change of the internal state

The internal state is not changed.

Classification

pulbic

Type

method only

See also:

None

TSKM_ERR_t tskm_svcWakeupRequest (TSKM_SVC_CTX_t * *p_svc*, TSKM_GSTEP_REQ_INFO_t * *p_req*)

Summary

Send started request to service.

Parameters:

in		

Rom_access_library

struct LBM_boot_wpinfo_t
struct LBM_boot_t
struct _tmb_ram_t
class BOOT_AccessIf
BOOT_AccessIf. struct RAM_SM_INFO_t
class RAM_AccessIf
RAM_AccessIf. class ROM_AccessIf

***ROM_AccessIf*. Macros**

#define RET_DEV_NORMAL 0
#define RET_DEV_ERR_PARAM 3
#define RET_DEV_ERROR 4
#define RET_DEV_RD_ERR_OTHER 1
#define RET_DEV_RD_ERR 2
#define RET_DEV_RD_ERR_1 -1
#define RET_DEV_RD_ERR_2 -2
#define RET_DEV_RD_ERR_3 -3
#define RET_DEV_RD_ERR_4 -4
#define RET_DEV_WT_ERR_OTHER 1
#define RET_DEV_WT_ERR 2
#define **LBM_UINT32** unsigned int
#define **LBM_UINT16** uint16_t
#define **LBM_UINT8** unsigned char
#define **LBM_INT64** int64_t
#define **LBM_INT32** unsigned int
#define **LBM_INT16** uint16_t
#define **LBM_INT8** unsigned char
#define **LBM_NOR_MIRROR_SIZE** 0x10000

```
#define LBM_DRAM_SIZE  0x10000
#define LBM_NOR_BOOT_SIZE  128
#define LBM_NOR_TSKM_SIZE  128
#define LBM_NOR_VUP_SIZE  1024
#define LBM_NOR_ANA_SIZE  32
#define LBM_NOR_VEHI_SIZE  32
#define LBM_NOR_DEVMGR_SIZE  32
#define SYSUP_DRAM_NORMAL  (LBM_UINT32)***  /* normal value                    NRNR*/
#define SYSUP_DRAM_SFTERRSR  (LBM_UINT32) ***/* abnormal Soft
    Reset(softwareabnormality) sfsf*/
#define SYSUP_DRAM_DISCARD  (LBM_UINT32) ***/* need to discard backup
    DCDC*/
#define SYSUP_CND_FACTRESET  (LBM_UINT16) ***/* COLD START */
#define SYSUP_CND_NORMAL  (LBM_UINT16) ***/* HOT START  */
#define SYSUP_MODE_NORMAL  (LBM_UINT32) ***/* normal mode          */
#define SYSUP_MODE_VERUP  (LBM_UINT32) ***/* versionup mode       */
#define SUBCPU_STS_NOMAL  (LBM_UINT32) ***/* Communication normal    */
#define SUBCPU_STS_COMNG  (LBM_UINT32) ***/* Communication abnormal   */
#define SDRAM_STS_NOMAL  (LBM_UINT32) ***/* backup power supply is normal   */
#define SDRAM_STS_BUPNG  (LBM_UINT32) ***/* backup power supply is abnormal */
#define HWDT_STS_NOMAL  (LBM_UINT32) ***/* WDT normal              */
#define HWDT_STS_WDTTO  (LBM_UINT32) ***/* WDT time-out outbreak       */
#define SWDT_STS_NOMAL  HWDT_STS_NOMAL       /* SWDT normal             */
#define SWDT_STS_WDTTO  HWDT_STS_WDTTO       /* SWDT time-out outbreak      */
#define IFS_PROG1_CODE  (LBM_UINT8)0xAA  /* Side PROG1          */
#define IFS_PROG2_CODE  (LBM_UINT8)0x55  /* Side PROG2          */
#define UPTBLE_ID_MAX  (LBM_UINT8)0x20  /* LBM_UPTBLINFO Maximum value */
#define UPTBLE_ID_NORBOOT  (LBM_UINT8)0x00  /* NOR-BOOT ID             */
#define UPTBLE_ID_SCRLDR  (LBM_UINT8)0x01  /* SecureLoader ID         */
#define UPTBLE_ID_NANDBOOT  (LBM_UINT8)0x02  /* NAND-BOOT ID            */
#define UPTBLE_ID_ROOTFS  (LBM_UINT8)0x03  /* Rootfs1 ID              */
#define UPTBLE_ID_ROOTF_CWORD98_  (LBM_UINT8)0x04  /* Rootf_CWORD98_ ID              */
#define STUP_STS_NORMAL  (LBM_UINT32)0x00000000  /* normal status   */
#define STUP_STS_ERROR_DETEC  (LBM_UINT32)0x22222222
#define STUP_STS_UPDATING  (LBM_UINT32)0x33333333  /* updating status */
#define STUP_STS_SIGNVRFY_ERROR  (LBM_UINT32)0x11111111
#define UPDATE_NOTI_NONE  (LBM_UINT32)0x00000000 /* Notify is NOT required. */
#define UPDATE_NOTI_EXIST  (LBM_UINT32)0xEEEEEEEE /* Notify is required.    */
#define RAMJUDGE_STS_NOMAL  (LBM_UINT32)0x44444444 /* RAM Judge Port is High */
#define RAMJUDGE_STS_NG  (LBM_UINT32)0xDDDDDDDD /* RAM Judge Port is Low  */
#define SELF_REFRESH_OK  (LBM_UINT32)0x55555555 /* DRAM Self Refresh is OK */
#define SELF_REFRESH_NG  (LBM_UINT32)0xCCCCCCCC /* DRAM Self Refresh is NG */
#define PRODUCTROM  (LBM_UINT32)0x11111111
#define RELEASEROM  (LBM_UINT32)0x22222222
#define RELEASEROM_REMAIN_USBBOOT  (LBM_UINT32)0x44444444
#define DEBUGROM  (LBM_UINT32)0x88888888
#define WRITE_NONE  (LBM_UINT32)0x00000000
#define WRITE_DONE  (LBM_UINT32)0x11111111
```

#define **DEFAULT_OPIMAGE_DRAM_BACKUPED** (LBM_UINT32)0x00000000
#define **DEFAULT_OPIMAGE_FROM_EMMC** (LBM_UINT32)0xAAAAAAAA
#define **SPECIFIC_OPIMAGE_DRAM_BACKUPED** (LBM_UINT32)0x11111111
#define **SPECIFIC_OPIMAGE_FROM_EMMC** (LBM_UINT32)0xBBBBBBBB
#define **SPECIFIC_OPIMAGE_FLAG_NONE** (LBM_UINT32)0x00000000
#define **SPECIFIC_OPIMAGE_FILE_NONE** (LBM_UINT32)0x11111111
#define **SPECIFIC_OPIMAGE_FILESIZE_0** (LBM_UINT32)0x22222222
#define **SPECIFIC_OPIMAGE_FILE_EXIST** (LBM_UINT32)0xEEEEEEEE
#define **SS_SYS_AREA_BOOT_MAX_SIZE** (0x00010000UL)
#define **SS_SYS_AREA_ROM_OFFSET** (0x00010000UL)
#define **SS_SYS_AREA_ROM_MAX_SIZE** (0x00001000UL)
#define **SS_SYS_AREA_RAM_OFFSET** (0x00011000UL)
#define **SS_SYS_AREA_RAM_MAX_SIZE** (0x00001000UL)
#define RAM_PRODUCT_PRIVATE_MAX (96)
#define SS_SM_ORDER_NAME_MAX (32)
#define ROM_PRODUCT_PRIVATE_MAX 128

Typedefs

typedef struct LBM_boot_wpinfo_t **LBM_UPTBLINFO**
typedef struct LBM_boot_t **LBM_NOR_t**
typedef struct _tmb_ram_t **LBM_RAM_t**
typedef uint32_t EPROGUPDATE_STATE

Enumerations

enum BAI_OPEN_t { **BAI_OPEN_RO**, **BAI_OPEN_RW** }
enum RAM_WAKEUP_STATE { **RAM_WAKEUP_STATE_DONT_CARE** = ***,
RAM_WAKEUP_STATE_BACKUP_NG, **RAM_WAKEUP_STATE_BATTERY_DOWN** }
enum EBOOT_MODE { **APPLICATION_MODE** = ***, **PROGRAMMING_MODE** }
enum EACTIVE_FLASHLOADER { **NEW_FLASHLOADER** = ***, **OLD_FLASHLOADER** }
enum EUSER_MODE { **USER_OFF** = ***, **USER_ON** }
enum ECONTROL_MODE { **DISABLE_MODE** = ***, **ENABLE_MODE** }
enum EDATARESET_MODE { **DATARESET_NONE** = ***, **DATARESET_USER**,
DATARESET_FACTORY }
enum ELASTILGRESET_MODE { **LAST_ILGRESET_NORMAL** = ***, **LAST_ILGRESET_NG** }
enum ENEXT_WAKEUP_TYPE { **NEXT_WAKEUP_TYPE_NONE** = ***, **NEXT_WAKEUP_TYPE_COLD**,
NEXT_WAKEUP_TYPE_HOT }
enum DRAM_BACKUP_STATE { **DRAM_BACKUP_STATE_OK** = ***, **DRAM_BACKUP_STATE_NG** }

Functions

int mtdn_backup_Read (const char *path_name, int i_id, int i_offset, int i_size, void *p_buff)
int mtdn_backup_Write (const char *path_name, int i_id, int i_offset, int i_size, void *p_buff)

Detailed Description

Class Documentation

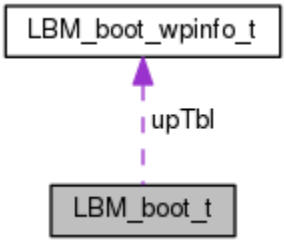
struct LBM_boot_wpinfo_t

Class Members:

LBM_UINT32	currentside	
LBM_UINT32	stupStsA	
LBM_UINT32	stupStsB	

struct LBM_boot_t

Collaboration diagram for LBM_boot_t:



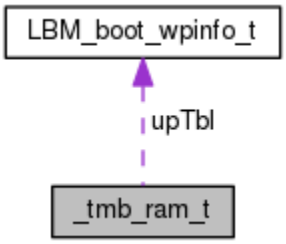
IMAGE

Class Members:

LBM_UINT32	btmode	
LBM_UINT32	sts1	
LBM_UINT32	sts2	
LBM_UINT32	sts3	
LBM_UPTBLINFO	upTbl[UPTBLE_ID_MAX]	

struct _tmb_ram_t

Collaboration diagram for _tmb_ram_t:



IMAGE

Class Members:

LBM_UINT32	bootNorWrite	
LBM_UINT32	dram1	
LBM_UINT32	dram2	
LBM_UINT32	dram3	
LBM_UINT32	dram_self_refresh	
LBM_UINT32	emmcTap	

LBM_UINT32	hwdtSts	
LBM_UINT32	mdUpdate	
LBM_UINT32	naviDetFactor	
LBM_UINT32	opdtLoadFactor	
LBM_UINT32	opdtLoadSatus	
LBM_UINT32	opdtUpdateReq	
LBM_UINT32	pwSts	
LBM_UINT32	ramjudgeSts	
LBM_UINT32	releaseNorType	
LBM_UINT32	sdUpdate	
LBM_UINT32	sectorNo	
LBM_UINT32	socCpuType	
LBM_UINT32	socEsNumber	
LBM_UINT32	spiBootDevice	
LBM_UINT32	sts	
LBM_UINT32	swdtSts	
LBM_UINT32	syscomSts	
LBM_UINT32	upmode	
LBM_UPTBLINFO	upTbl[UPTBLE_ID_MAX]	

struct RAM_SM_INFO_t

System management information

Class Members:

BOOL	isErrorReset	
BOOL	isImmediateReset	
uint32_t	lastDramBackupStatus	
uint32_t	lastResetStatus	
uint32_t	lastWakeupType	
BOOL	needRenotifyStartPrm	
char	order_name[SS_SM_ORDER_NAME_MAX]	
char	productPrivate[RAM_PRODUCT_PRIVATE_MAX]	
char	signature_in[4]	
char	signature_out[4]	

Macro Definition Documentation

#define RAM_PRODUCT_PRIVATE_MAX (96)

max length of ram product private

#define RET_DEV_ERR_PARAM 3

Common return value: Parameter error return value of API

#define RET_DEV_ERROR 4

Common return value: Error of getting device info

#define RET_DEV_NORMAL 0

Common return value: Normal return value of API

#define RET_DEV_RD_ERR 2

Read API return value: Acquire buffer error(malloc)

#define RET_DEV_RD_ERR_1 -1

Read API return value: error type 1

#define RET_DEV_RD_ERR_2 -2

Read API return value: error type 2

#define RET_DEV_RD_ERR_3 -3

Read API return value: error type 3

#define RET_DEV_RD_ERR_4 -4

Read API return value: error type 4

#define RET_DEV_RD_ERR_OTHER 1

Read API return value: Internal error

#define RET_DEV_WT_ERR 2

Write API return value: Internal error

#define RET_DEV_WT_ERR_OTHER 1

Write API return value: Other error

#define ROM_PRODUCT_PRIVATE_MAX 128

max length of rom product private

#define SS_SM_ORDER_NAME_MAX (32)

max length of order file name

Typedef Documentation

typedef uint32_t EPROGUPDATE_STATE

Program update state information

Enumeration Type Documentation

enum BAI_OPEN_t

Access type to the boot information

enum DRAM_BACKUP_STATE

DRAM backup state

enum EACTIVE_FLASHLOADER

active flash load information

enum EBOOT_MODE

boot mode information

enum ECONTROL_MODE

control mode information

enum EDATARESET_MODE

data reset mode information

enum ELASTILGRESET_MODE

last illegal reset mode information

enum ENEXT_WAKEUP_TYPE

next wakeup type

enum EUSER_MODE

user mode information

enum RAM_WAKEUP_STATE

System wakeup state type

Function Documentation

int mtdn_backup_Read (const char * *path_name*, int *i_id*, int *i_offset*, int *i_size*, void * *p_buff*)

Brief

Read data from norflash device.

Parameters:

in	*path_name*	const char* - Device path to read(such as "/dev/mtd1ro")
in	*i_id*	int - Main block ID(0, 2, 4, ...)
in	*i_offset*	int - Data offset
in	*i_size*	int - Data size to read
in	*p_buff*	Void* - Buffer for storing read data

Return values:

RET_DEV_NORMAL	OK
RET_DEV_ERROR	Get device error
RET_DEV_ERR_PARAM	Parameter error
RET_DEV_RD_ERR_OTHER	Internal error
RET_DEV_RD_ERR	Acquire buffer error(malloc)
RET_DEV_RD_ERR_1	Error type 1
RET_DEV_RD_ERR_2	Error type 2
RET_DEV_RD_ERR_3	Error type 3
RET_DEV_RD_ERR_4	Error type 4

Preconditions

There is no preconditions for this API.

Change of internal state

Change of internal state according to the API does not occur.

Classification

Public

Type

Sync Only

See also:

mtdn_backup_Write

int mtdn_backup_Write (const char * *path_name*, int *i_id*, int *i_offset*, int *i_size*, void * *p_buff*)

Brief

Write data to norflash device.

Parameters:

in	*path_name*	const char* - Device path to write(such as "/dev/mtd")
in	*i_id*	int - Main block ID(0, 2, 4, ...)
in	*i_offset*	int - Data offset
in	*i_size*	int - Data size to write
in	*p_buff*	Void* - Buffer for storing written data

Return values:

RET_DEV_NORMAL	OK
RET_DEV_ERROR	Get device error
RET_DEV_ERR_PARAM	Parameter error
RET_DEV_WT_ERR_OTHER	Other error
RET_DEV_WT_ERR	Internal error

Preconditions

There is no preconditions for this API.

Change of internal state

Change of internal state according to the API does not occur.

Classification

Public

Type

Sync Only

See also:

mtdn_backup_Read

Resource_manager

struct RESM_REQ_EVENT_t
struct RESM_REQ_EVENT_t.prm
struct RESM_INET_STATUS_t
struct RESM_INET_STATUS_t.ifInfo
struct RESM_STATUS_t

Macros

#define **NTFY_ResourceMgr_Availability** MN_SS_RESOURCEMGR"/Availability"
#define **RESM_ERR_t** int32_t
#define **RESM_E_OK** (0)
#define **RESM_E_PAR** (-1)
#define **RESM_E_NG** (-2)
#define **RESM_EV_t** uint32_t
#define **RESM_EV_NOP** (0x00)
#define **RESM_EV_MEM** (0x01)
#define **RESM_EV_NAND_STATUS** (0x02)

#define **RESM_RSV_t** int32_t
#define **RESM_INET_IF_MAX** 5 /* tentative */
#define **HWADDR_LEN** (6)

Enumerations

enum **RESM_NAND_WRITE_STATUS_t** { **RESM_NAND_WRITE_ENABLE** = ***,
RESM_NAND_WRITE_DISABLE = -1 }

Functions

RESM_ERR_t RESM_Open (const RESM_RSV_t *p_prim, uint32_t *p_ssnld)
RESM_ERR_t RESM_Close (uint32_t ssnld)
RESM_ERR_t RESM_ReqEvent (uint32_t ssnld, const RESM_REQ_EVENT_t *p_reqEvent)
RESM_ERR_t RESM_GetEventFd (uint32_t ssnld, int *p_fd)
RESM_ERR_t RESM_GetEvent (uint32_t ssnld, RESM_EV_t *p_evFlag)
RESM_ERR_t RESM_GetStatus (uint32_t ssnld, RESM_STATUS_t *p_status)

Detailed Description

Class Documentation

struct RESM_REQ_EVENT_t

Class Members:

struct RESM_REQ_EVENT_t	prm	
RESM_EV_t	reqEvent	

struct RESM_REQ_EVENT_t.prm

Class Members:

uint32_t	restMemThresh	

struct RESM_INET_STATUS_t

Class Members:

struct RESM_INET_STATUS_t	ifInfo[RESM_INET_IF_MAX]	
uint32_t	ifNum	

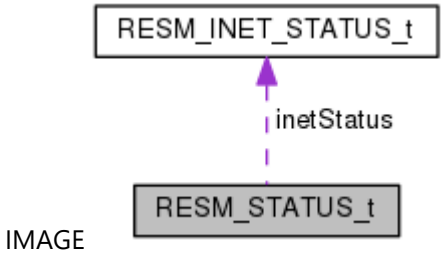
struct RESM_INET_STATUS_t.ifInfo

Class Members:

uint8_t	hwaddr[HWADDR_LEN]	
char	name[32]	
uint64_t	rxSize	
uint64_t	txSize	

struct RESM_STATUS_t

Collaboration diagram for RESM_STATUS_t:



IMAGE

Class Members:

RESM_INET_STATUS_t	inetStatus	
RESM_NAND_WRITE_STATUS_t	nandWriteStatus	
uint32_t	restMemSize	

Function Documentation

RESM_ERR_t RESM_Close (uint32_t *ssnld*)

Summary

Closes the session with resource manager.

Parameters:

in	*ssnld*	uint32_t - Session ID

Return values:

RESM_E_OK	Succeeded
RESM_E_PAR	Parameter error

Preconditions

An application must finish getting a session ID by RESM_Open().

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

RESM_Open

RESM_ERR_t RESM_GetEvent (uint32_t *ssnld*, RESM_EV_t * *p_evFlag*)

Summary

Gets the event from resource manager.

Parameters:

in	*ssnld*	uint32_t - Session ID
out	*p_evFlag*	RESM_EV_t* - Pointer to event

RESM_EV_t Variables

```
1 #define RESM_EV_t          uint32_t
2 #define RESM_EV_NOP        (0x00)
3 #define RESM_EV_MEM        (0x01)
4 #define RESM_EV_NAND_STATUS (0x02)
```

RESM_EV_NOP : No event notification
RESM_EV_MEM : Event notification for memory
RESM_EV_NAND_STATUS : Event notification for NAND state

Return values:

RESM_E_OK	Succeeded
RESM_E_PAR	Parameter error
RESM_E_NG	Unexpected error

Preconditions

An application must finish getting a session ID by RESM_Open().

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

RESM_ReqEvent

RESM_ERR_t RESM_GetEventFd (uint32_t *ssnld*, int * *p_fd*)

Summary

Gets FD of the event to receive from resource manager.

Parameters:

in	*ssnld*	uint32_t - Session ID
out	*p_fd*	int* - Pointer to file descriptor

Return values:

RESM_E_OK	Succeeded
RESM_E_PAR	Parameter error
RESM_E_NG	Unexpected error

Preconditions

An application must finish getting a session ID by RESM_Open().

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

None

RESM_ERR_t RESM_GetStatus (uint32_t *ssnld*, RESM_STATUS_t * *p_status*)

Summary

Gets system state.

Parameters:

in	*ssnld*	uint32_t - Session ID
out	*p_status*	RESM_STATUS_t* - Pointer to system state

RESM_STATUS_t Structure

```
1 typedef struct {
2    uint32_t restMemSize;                  /* Remaining memory length (KB)                  */
3    RESM_NAND_WRITE_STATUS_t  nandWriteStatus;  /* Access permission / Prohibition state to NAND flash
*/
4    RESM_INET_STATUS_t       inetStatus;      /* Network IF statistical information after start    */
5 } RESM_STATUS_t;
```

enum RESM_NAND_WRITE_STATUS_t Variables

RESM_NAND_WRITE_ENABLE : NAND access permission
RESM_NAND_WRITE_DISABLE : NAND access prohibition

Notes
When system records in NAND flash and it influences the lifetime of a device, STATUS(nandWriteStatus) becomes RESM_NAND_WRITE_DISABLE. It is not in the state where WRITE fails.

RESM_INET_STATUS_t Structure

```
1 typedef struct {
2    uint32_t ifNum;                /* Valid array at the number of interface and ifInfo */
3    struct {
4       char    name[32];           /* Interface name                          */
5       uint64_t  rxSize;           /* Receiving data length (KiB)             */
6       uint64_t  txSize;           /* Transmission data length (KiB)          */
7       uint8_t   hwaddr[HWADDR_LEN];  /* Hardware address (MAC address)       */
8    } ifInfo[RESM_INET_IF_MAX];
9 }RESM_INET_STATUS_t;
```

Return values:

RESM_E_OK	Succeeded
RESM_E_PAR	Parameter error
RESM_E_NG	Unexpected error

Preconditions

An application must finish getting a session ID by RESM_Open().

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

None

RESM_ERR_t RESM_Open (const RESM_RSV_t * *p_prim*, uint32_t * *p_ssnId*)

Summary

Opens the session with resource manager.

Parameters:

in	*p_prim*	RESM_RSV_t* - Unused, NULL
out	*p_ssnId*	uint32_t* - Pointer to session ID

RESM_RSV_t Variables

```
1 #define RESM_RSV_t       int32_t
2 #define RESM_RSV_NULL    (0x00)  // NULL
```

Return values:

RESM_E_OK	Succeeded
RESM_E_PAR	Parameter error
RESM_E_NG	Unexpected error

Preconditions

Availability of ResourceManager must be TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

RESM_Close

RESM_ERR_t RESM_ReqEvent (uint32_t *ssnId*, const RESM_REQ_EVENT_t * *p_reqEvent*)

Summary

Requests to issue an event.

Parameters:

in	*ssnId*	uint32_t - Session ID
in	*p_reqEvent*	RESM_REQ_EVENT_t* - Pointer to event

RESM_REQ_EVENT_t Structure

```
1 typedef struct {
2    RESM_EV_t reqEvent;         /* Event flag necessary to report */
3    struct {
4        uint32_t restMemThresh;  /* Threshold of the remaining capacity of system memory */
5    } prm;
6 }RESM_REQ_EVENT_t;
```

RESM_EV_t Variables

```
1 #define RESM_EV_t          uint32_t
2 #define RESM_EV_NOP        (0x00)
3 #define RESM_EV_MEM        (0x01)
4 #define RESM_EV_NAND_STATUS (0x02)
```

RESM_EV_NOP : No event notification
RESM_EV_MEM : Event notification for memory
RESM_EV_NAND_STATUS : Eventnotification for NAND state

Notes
restMemThresh is valid only when RESM_EV_MEM is set to reqEvent.
ResourceManager issues RESM_EV_MEM event at the timing when the remaining capacity of system memory fell below the value (unit is BYTE) which an application specifies.

Return values:

RESM_E_OK	Succeeded
RESM_E_PAR	Parameter error

Preconditions

An application must finish getting a session ID by RESM_Open().

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

RESM_GetEvent

Config

struct RAM_AccessPrivate

RAM Access Private data define. *More...*

union RAM_AccessPrivate.__unnamed__
struct RAM_AccessPrivate.__unnamed__.__unnamed__

CWORD80 error define *More...*

struct ROM_AccessPrivate

ROM Access Private data define. *More...*

union ROM_AccessPrivate.__unnamed__
struct ROM_AccessPrivate.__unnamed__.__unnamed__

CWORD80 error count, connect state *More...*

struct T_SS_SM_INIT_HOOK

Version up mode, Callback function. struct T_SS_SM_INIT_HOOK_OUT_PARAM

Not in OOM Killer's service object, Group Relaunch service. *More...*

Macros

#define **SS_CONN_NONE** 0
#define **SS_CONN_HAS__CWORD80_** 1
#define **SS_CONN_HASNOT__CWORD80_** 2

Typedefs

typedef struct T_SS_SM_INIT_HOOK **T_SS_SM_INIT_HOOK_IN_PARAM**

Functions

E_CWORD33_Status ss_sm_initHook (HANDLE hApp, const T_SS_SM_INIT_HOOK_IN_PARAM *inPrm, T_SS_SM_INIT_HOOK_OUT_PARAM *outPrm)

Detailed Description

Class Documentation

struct RAM_AccessPrivate

RAM Access Private data define.

Class Members:

union RAM_AccessPrivate	__unnamed__	

union RAM_AccessPrivate.__unnamed__

Class Members:

__unnamed__	__unnamed__	*CWORD80* error define
uint8_t	max[RAM_PRODUCT_PRIVATE_MAX]	RAM product private max.

struct RAM_AccessPrivate.__unnamed__.__unnamed__

CWORD80 error define

Class Members:

uint32_t	dummy	
BOOL	is_CWORD80_Error	
uint32_t	retryOverAuthNGCount	

struct ROM_AccessPrivate

ROM Access Private data define.

Class Members:

union ROM_AccessPrivate	__unnamed__	

union ROM_AccessPrivate.__unnamed__

Class Members:

__unnamed__	__unnamed__	*CWORD80* error count, connect state
uint8_t	max[ROM_PRODUCT_PRIVATE_MAX]	ROM product private max.

struct ROM_AccessPrivate.__unnamed__.__unnamed__

CWORD80 error count, connect state

Class Members:

uint32_t	_CWORD80_ConnState	
uint32_t	_CWORD80_ErrorCount	
uint32_t	_CWORD80_ForceDiscon	

struct T_SS_SM_INIT_HOOK_OUT_PARAM

Not in OOM Killer's service object, Group Relaunch service.

Class Members:

vector< string >	groupRelaunchSvcs	Not in OOM Killer's service object.
vector< string >	protectedSvcs	

Function Documentation

E_CWORD33_Status ss_sm_initHook (HANDLE *hApp*, const T_SS_SM_INIT_HOOK_IN_PARAM * *inPrm*, T_SS_SM_INIT_HOOK_OUT_PARAM * *outPrm*)

Summary

Parameters:

in	*hApp*	HANDLE - HANDLE Application
in	*inPrm*	T_SS_SM_INIT_HOOK_IN_PARAM -
in	*outPrm*	T_SS_SM_INIT_HOOK_OUT_PARAM -

T_SS_SM_INIT_HOOK_IN_PARAM struct

```
1 typedef struct T_SS_SM_INIT_HOOK {
2   BOOL bIsVupMode;                          /* version up model */
3   E_CWORD33_Status (*cbRebootNoticeFunc)(HANDLE hApp); /* call back function */
4   T_SS_SM_INIT_HOOK()
5     : bIsVupMode(FALSE),
6     cbRebootNoticeFunc(NULL) {
7   }                                         /* constructor  */
8 } T_SS_SM_INIT_HOOK_IN_PARAM;
```

T_SS_SM_INIT_HOOK_IN_PARAM struct

```
1 typedef struct {
2   std::vector<std::string> protectedSvcs;     /* OOM Killer's protected service */
3   std::vector<std::string> groupRelaunchSvcs; /* group relunch service */
4 } T_SS_SM_INIT_HOOK_OUT_PARAM;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

-None

Change of the internal state

save inPrm->cbRebootNoticeFunc to s_confCtx.cbRebootNoticeFunc

Classification

Public

Type

Sync

See also:

None

System_manager

file ss_system_manager_if.h

This file supports the System Manager client interface. file ss_system_manager_if_local.h

This file supports the System Manager client interface. file ss_system_manager_notifications.h

This file supports the System Manager client interface. file ss_system_manager_protocol.h

This file supports the System Manager client interface. file ss_system_manager_protocol_local.h

This file supports the System Manager client interface. file ss_test_clients.h

This file supports test client queue names. **Classes**

struct T_SystemManagerCpuResetInfo

CPU Reset Info. struct T_SystemManagerBootModeRequestResponse

struct T_SS_BootModeListStruct

Data Reset Reason. *More...*

struct T_SS_SM_UserModeOnOffNotification_Struct
struct T_SS_SM_START_DataStruct
struct T_SS_SM_STOP_DataStruct
struct T_SS_SM_START_ExtDataStructType
struct T_SS_SM_STOP_ExtDataStructType
struct T_SystemManagerLoggingRequestInfo

*Error Event Logging Info *.* *More...*

struct T_SystemManagerClearLogsInfo

*Error Event Clear Logs Info *.* *More...*

struct T_SMRequestMsg
struct _SS_SMCurrentState
struct _SMCompleteAck
struct _SMGroupLaunchTrigger
class Timer

Timer. class EnumStringMap< enumT, fp >

EnumStringMap. **Macros**

#define NTFY_HeartBeatAvailability SERVICE_HEARTBEAT"/Availability"

Indicates if HeartBeat is available or unavailable.

#define **SS_PKG_SPI_LOADER** "SPI_LOADER"
#define **SS_PKG_UBOOT** "UBOOT"
#define **SS_PKG_SECURE_LOADER** "SECURE_LOADER"
#define **SS_PKG_SYS_UCOM** "SYS_UCOM"
#define **SS_PKG_CAN_UCOM** "CAN_UCOM"
#define **SS_PKG__CWORD52__UCOM** "_CWORD52__UCOM"
#define **SS_PKG__CWORD52__DATA** "_CWORD52__DATA"
#define **SS_PKG__CWORD52__FIXEQ** "_CWORD52__FIXEQ"
#define **SS_PKG_MAIN_EMMC** "MAIN_EMMC"
#define **SS_PKG_TARGETBOARD** "TARGETBOARD"
#define **SS_PKG_MAIN_PRODUCT_SI** "MAIN_PRODUCT_SI"
#define **SS_PKG_MAIN_COMP_SI** "MAIN_COMP_SI"
#define **SS_PKG_MAIN_LPF_SI** "MAIN_LPF_SI"
#define **SS_PKG_FACTORY_FROM** "FACTORY_FROM"
#define **SS_PKG_FACTORY_FROM2** "FACTORY_FROM2"
#define **SS_PKG_FACTORY_EEPROM** "FACTORY_EEPROM"
#define **SS_PKG_DECK** "DECK"
#define **SS_PKG_REAGION_CODE** "REAGION_CODE"

#define **SS_PKG__CWORD8__DB** "_CWORD8__DB"
#define **SS_PKG_CAPACITIVE_TP** "CAPACITIVE_TP"
#define **SS_PKG__CWORD52___CWORD32_** "_CWORD52___CWORD32_"
#define **SS_PKG_RADIO_PARAMETER** "RADIO_PARAMETER"
#define **SS_PKG_VR_DATA** "VR_DATA"
#define **SS_PKG_NAVI_GPS** "NAVI_GPS"
#define **SS_PKG_DTV_MAIN** "DTV_MAIN"
#define **SS_PKG_DTV_EEPROM** "DTV_EEPROM"
#define **SS_PKG_DTV_BOOT** "DTV_BOOT"
#define **SS_PKG_DTV_STATION_DB** "DTV_STATION_DB"
#define **SS_PKG_XM_TYPE_ID** "XM_TYPE_ID"
#define **SS_PKG_XM_HW** "XM_HW"
#define **SS_PKG_XM_SW** "XM_SW"
#define **SS_PKG_XM_SXI** "XM_SXI"
#define **SS_PKG_XM_BB** "XM_BB"
#define **SS_PKG_XM_HDEC** "XM_HDEC"
#define **SS_PKG_XM_RF** "XM_RF"
#define **SS_PKG_XM_SPL** "XM_SPL"
#define **SS_PKG_XM_AREA** "XM_AREA"
#define **SS_PKG_NAVI_PROGRAM** "NAVI_PROGRAM"
#define **SS_PKG_NAVI_MAP** "NAVI_MAP"
#define **SS_PKG_PHASE_INFO** "PHASE_INFO"
#define **SS_PKG_SERIES_INFO** "SERIES_INFO"
#define **SS_PKG_FPGA** "FPGA"
#define **SS_PKG_VP__CWORD63_** "VP__CWORD63_"
#define **SS_PKG_VP__CWORD31_** "VP__CWORD31_"
#define **SS_PKG_PREINSTALL_DATA** "PREINSTALL_DATA"
#define **MSG_PSM_REBOOT_NOTIFY** 0x11
#define szNTFY_PowerLVI1 SERVICE_POWER"/LVI1"

Depreciated !!

#define szNTFY_PowerLVI2 SERVICE_POWER"/LVI2"

Depreciated !!

#define szNTFY_PowerLevel SERVICE_POWER"/Level"

Indicates the current power level state.

#define szNTFY_ShutdownPopup SERVICE_POWER"/ShutdownPopup"
#define szNTFY_PowerPopup SERVICE_POWER"/PowerPopup"
#define NTFY_SSSystemMgrAvailability SERVICE_SYSMANAGER"/Availability"

System Manager availability -.

#define NTFY_SSSystemMgrState SERVICE_SYSMANAGER"/State"

System Manager state - Init,.

#define NTFY_SSSystemMgrPowerOnOff SERVICE_SYSMANAGER"/PowerOnOffState"

Power ON notification for.

#define NTFY_SSSystemMgrStartUpType SERVICE_SYSMANAGER"/StartUpType"

Notification for start up type.

#define NTFY_SSHUBootModes SERVICE_SYSMANAGER"/HeadUnitBootModes"

Notification for boot modes.

#define NTFY_SSServiceWakeupStatus SERVICE_SYSMANAGER"/ServiceWakeupStatus"

Service Wakeup state.

#define NTFY_SSNeedAplRestart SERVICE_SYSMANAGER"/NeedAplRestart"

Notification for Need APL Restart.

#define NTFY_SSSystemMgrShutdownStarted SERVICE_SYSMANAGER"/ShutdownStarted"
#define **SS_SM_MAX_MODULE_NAME_LENGTH** (256)
#define **SS_SM_MAX_TEST_CASE_NAME_SIZE** (128)
#define **SS_SM_MAX_RESP_MSG_SIZE** (2048)
#define **SS_SM_MAX_MODULE_LOG_MASK_LENGTH** (50)
#define **SS_SM_MAX_MODULE_PATH_LENGTH** (256)
#define **SS_SM_MAX_MODULE_ARGS_LENGTH** (512)
#define **ZONE_ERR** ZONEMASK(31)
#define LOG_ERROR(pStr)
#define **LOG_SUCCESS**(pStr) _CWORD33_LOG(ZONE_INFO, __FUNCTION__, " %s successful", pStr);
#define LOG_STATUS(l_eStatus, pStr)
#define CALL_AND_LOG_STATUS(fnc)
#define LOG_STATUS_IF_ERRORED(l_eStatus, pStr)
#define CALL_AND_LOG_STATUS_IF_ERRORED(fnc)
#define MAP_ENTRY(f_map, f_enum) f_map[f_enum] = #f_enum
#define _CWORD33_LOG_RECEIVED_FROM(hApp) _CWORD33_LOG(ZONE_INFO, __FUNCTION__, " Received from %s", _CWORD33_GetMsgSrc(hApp))
#define _CWORD33_LOG_WHEN_COMPILED
#define **SS_ASERT**(x)
#define **SS_ASERT_ERRNO**(x)
#define **SS_ASERT_LOG**(x, fmt, ...)
#define **SS_STATIC_ASERT**(expr)

Typedefs

typedef enum T_SMHBProtocolMessages EHBProtocolMessages

HeatBeat Protocol Command heartbeat service protocol > define all protocol messages in and out of heart beat thread that are pertinent to the functionality offered by heart beat thread.

typedef UI_32 SMProgUpdateState

Program update status.

typedef struct T_SystemManagerCpuResetInfo TSystemManagerCpuResetInfo

CPU Reset Info.

typedef struct T_SystemManagerBootModeRequestResponse

TSystemManagerBootModeRequestResponse

typedef struct T_SS_BootModeListStruct TSS_BootModeListStruct

Data Reset Reason.

typedef struct T_SS_SM_UserModeOnOffNotification_Struct T_SS_SM_UserModeOnOffNotification_StructType

typedef struct T_SS_SM_START_DataStruct T_SS_SM_START_DataStructType

typedef struct T_SS_SM_STOP_DataStruct T_SS_SM_STOP_DataStructType
typedef struct T_SystemManagerLoggingRequestInfo TSystemManagerLoggingRequestInfo
*Error Event Logging Info *.*
typedef struct T_SystemManagerClearLogsInfo TSystemManagerClearLogsInfo
*Error Event Clear Logs Info *.*
typedef struct T_SMRequestMsg TSMRequestMessage
typedef enum _SS_SystemManagerProtocol SS_SystemManagerProtocol
typedef struct _SS_SMCurrentState **SS_SMCurrentState**
typedef struct _SMCompleteAck **SMStopCompleteAck**
typedef struct _SMGroupLaunchTrigger **SMGroupLaunchTrigger**
typedef UI_64 **SS_VersionNumberType**

Enumerations

enum T_SMHBProtocolMessages { SS_HEARTBEAT_PRINT_CONNECTIONS = ***, SS_HEARTBEAT_PRINT_STACK = ***, SS_HEARTBEAT_PERIODIC_STATUS_REQ = ***, SS_HEARTBEAT_PERIODIC_RESP = ***, SS_HEARTBEAT_START = ***, SS_HEARTBEAT_STOP = ***, SS_HEARTBEAT_DELETE_MODULE_ENTRY = ***, SS_HEARTBEAT_MSG_CATEGORY_REPORT = ***, SS_HEARTBEAT_ERROR_DETECTED = ***, SS_HEARTBEAT_APPEND_MODULE_ENTRY = ***, SS_HEARTBEAT_AVAIL_CHECK_REQ = ***, SS_HEARTBEAT_AVAIL_CHECK_RESP = ***, SS_HEARTBEAT_RESPONSE = ***, SS_HEARTBEAT_REQUEST = *** }*HeatBeat Protocol Command heartbeat service protocol > define all protocol messages in and out of heart beat thread that are pertinent to the functionality offered by heart beat thread.*

enum ESMBootModeInfo { **e_SS_SM_BOOT_MODE_INVALID** = -1, **e_SS_SM_BOOT_MODE_APPLICATION**, **e_SS_SM_BOOT_MODE_PROGRAMMING** }*Boot mode information.*

enum ESMDataResetModeInfo { **e_SS_SM_DATA_RESET_MODE_NONE**, **e_SS_SM_DATA_RESET_MODE_USER**, **e_SS_SM_DATA_RESET_MODE_FACTORY**, **e_SS_SM_DATA_RESET_MODE_PROGUPDATE** }*Data reset mode information.*

enum ESMDramBackupStatus { **e_SS_SM_DRAM_BACKUP_UNSET** = -1, **e_SS_SM_DRAM_BACKUP_OK**, **e_SS_SM_DRAM_BACKUP_NG** }*DRAM backup status.*

enum ESMResetStatus { **e_SS_SM_RESET_STATUS_UNSET** = -1, **e_SS_SM_RESET_STATUS_NONE**, **e_SS_SM_RESET_STATUS_NG**, **e_SS_SM_RESET_STATUS_IMMEDIATE** }*Reset status.*

enum EMRelaunchStatus { **e_SS_SM_RELAUNCH_STATUS_NONE**, **e_SS_SM_RELAUNCH_STATUS_SAFE**, **e_SS_SM_RELAUNCH_STATUS_ERR** }*Relaunch status.*

enum ESMCpuResetReason { **e_SS_SM_CPU_RESET_REASON_INVALID** = -1, **e_SS_SM_CPU_RESET_REASON_NORMAL** = ***, **e_SS_SM_CPU_RESET_REASON_DATA_RESET**, **e_SS_SM_CPU_RESET_REASON_USER_FORCE_RESET**, **e_SS_SM_CPU_RESET_REASON_GENERIC**, **e_SS_SM_CPU_RESET_REASON_GENERIC_ERR** =

e_SS_SM_CPU_RESET_REASON_GENERIC, **e_SS_SM_CPU_RESET_REASON_DSP_ERR**, **e_SS_SM_CPU_RESET_REASON_IMMRESET_NORMAL**, **e_SS_SM_CPU_RESET_REASON_CRITICAL_ERR** }*CPU Reset Reason.*

enum ESMDataResetType { **e_SS_SM_DATA_RESET_TYPE_USER** = ***, **e_SS_SM_DATA_RESET_TYPE_FACTORY**, **e_SS_SM_DATA_RESET_TYPE_CONFIGURATION**, **e_SS_SM_DATA_RESET_TYPE_PROGUPDATE** }*Data Reset Reason.*

enum eSMUserLogType { e_SS_SM_CAPTURE_ALL_LOGS = ***, e_SS_SM_SCREEN_CAPTURE, e_SS_SM_CAPTURE_CWORD33_LOGS, e_SS_SM_USER_FORCE_RESET, e_SS_SM_CAPTURE_DEV_LOGS, e_SS_SM_CAPTURE_MODULE_LOGS, e_SS_SM_CAPTURE_DTC_LOGS, e_SS_SM_CAPTURE_NAVI_LOGS, e_SS_SM_CAPTURE_GROUP_RELAUNCH }

enum _SS_SystemManagerProtocol { **SS_SYSTEM_MANAGER_PROTOCOL_BEGINNING_INDEX** = ***, SS_SM_NOTIFY_SYSTEM_LAUNCH_COMPLETE = SS_SYSTEM_MANAGER_PROTOCOL_BEGINNING_INDEX, **SS_SM_PROTOCOL_INTERFACE_END**, SS_SM_START, SS_SM_START_COMPL_RSPN, SS_SM_STOP, SS_SM_STOP_COMPL_RSPN, SS_SM_WAKEUP_MODULES_CMPL_RSPN, SS_SM_SHUTDOWN_MODULES_CMPL_RSPN, SS_SM_WAKEUP_MODULES, SS_SM_POWER_OFF_MODULES, SS_SM_SHUTDOWN_MODULES, SS_SM_CRNT_STATE_QUERY, SS_SM_CRNT_STATE_QUERY_RSPN, SS_SM_SYSTEM_MODE_INFO_REQ, SS_SM_SYSTEM_MODE_INFO_RSPN, SS_SM_INITCOMP_REP, **SS_SM_PROTOCOL_OPEN_SESSION_REQ**, **SS_SM_PROTOCOL_OPEN_SESSION_ACK**, SS_SM_GET_START_EXT_INFO, SS_SM_GET_STOP_EXT_INFO, SS_SM_BOOT_MODE_SET_REQ, SS_SM_BOOT_MODE_SET_RESP, SS_SM_DATA_RESET_MODE_SET_REQ, SS_SM_PROG_UPDATE_STATE_SET_REQ, **SS_SM_CPU_RESET_REQ**, **SS_SM_LOCAL_DATA_RESET_REQ**, **SS_SM_REMOTE_DATA_RESET_REQ**, SS_SM_FWD_STARTUP_CONFIRMATION_MSG_REQ, SS_SM_FWD_START_CONFIRMATION_MSG_RESP, SS_SM_POWER_REQUEST_MSG, SS_SM_POWER_REQUEST_MSG_RESP, SS_SM_USER_MODE_SET_RESP, **SS_SM_DEBUG_VAR_CODE_DATA_SET_REQ**, **SS_SM_SET_BOOTLOADER_INFO**, **SS_SM_GET_BOOTLOADER_INFO**, **SS_SM_WAKEUP_ORDER_SET_REQ**, **SS_SM_NEXT_WAKEUP_TYPE_SET_REQ**, SS_SM_EVENT_ERROR, SS_SM_EVENT_ERROR_TO_SSL, SS_SM_ERROR_EVENT_LOGGING_START_REQ, SS_SM_ERROR_EVENT_LOGGING_START_RSPN, SS_SM_ERROR_EVENT_ARTIFACT_REQ, SS_SM_ERROR_EVENT_ARTIFACT_RSPN, SS_SM_ERROR_EVENT_LOGGING_COMPLETE, SS_SM_ERROR_EVENT_LOGGING_COMPLETE_RSPN, **SS_SM_ERROR_EVENT_TIMER_ID_LOGGER_START_REQ**, **SS_SM_ERROR_EVENT_TIMER_ID_DEBUG_DUMP_RSPN**, **SS_SM_ERROR_EVENT_TIMER_ID_BOOT_MICRO_LOG_RSPN**, **SS_SM_ERROR_EVENT_TIMER_ID_CORE_FILE_POLL**, SS_SM_USER_INVOKED_LOG_REQ, SS_SM_ERROR_EVENT_EEL_EXPORT_REQ, SS_SM_ERROR_EVENT_CWORD33_EMMC_LOGS_REQ, SS_SM_ERROR_EVENT_CWORD33_EMMC_LOGS_RSPN, SS_SM_ERROR_EVENT_CLR_LOGS_REQ, SS_SM_ERROR_EVENT_PROCESS_EXIT, SS_SM_ERROR_EVENT_DIAG_LOG_REQ, SS_SM_ERROR_EVENT_CAN_LOG_REQ, SS_SM_ERROR_EVENT_DTC_LOG_REQ, **SS_SM_GROUP_LAUNCH_TRIGGER**, SS_SM_DEBUG_DUMP, SS_SM_DEBUG_DUMP_RSPN, SS_SM_MODULE_RELAUNCH_REQ,

SS_SM_CPU_HIGH_LOAD_DETECTED, SS_SM_PROPAGATE_SYSTEM_ERROR,
SS_SM__CWORD56__HEARTBEAT_REQ, SS_SM__CWORD56__HEARTBEAT_RSPN,
SS_SM_EVENT_USER_DATA_RESET, SS_SM_BOOT_MICRO_RESET_NTF,
SS_SM_BOOT_MICRO_LOG_REQ, SS_SM_BOOT_MICRO_LOG_RSP, SS_SM_PRE_START,
SS_SM_PRE_START_COMPL_RSPN, SS_SM_PRE_STOP, SS_SM_PRE_STOP_COMPL_RSPN,
SS_SM_BACKGROUND_START, SS_SM_BACKGROUND_START_COMPL_RSPN,
SS_SM_BACKGROUND_STOP, SS_SM_BACKGROUND_STOP_COMPL_RSPN,
SS_SYSTEM_MANAGER_PROTOCOL_ENDING_INDEX =
SS_SM_BACKGROUND_STOP_COMPL_RSPN }

Functions

VOID **SendDebugDumpResponseToSystemManager** (std::string &f_messageStr)
E_CWORD33_Status **_CWORD33_SSFrameworkInterface** (HANDLE hApp)
template<typename T > E_CWORD33_Status ReadMsg (HANDLE hApp, T &Data,
ESMRetrieveTypes f_eRetrieveMethod=eSMRRelease)
EnumStringMap< enumT, fp >::EnumStringMap ()
EnumStringMap< enumT, fp >::~EnumStringMap ()
SS_String **EnumStringMap< enumT, fp >::GetStr** (enumT f_enum)

Variables

const PCSTR **TIMER_SERVICE_NAME** = "TIMER"

Detailed Description

Class Documentation

struct T_SystemManagerBootModeRequestResponse

Class Members:

E_CWORD33_Status	e_CWORD33_Status	
ESMBootModeInfo	hostBootMode	

struct T_SS_BootModeListStruct

Data Reset Reason.

Class Members:

EPWR_SC__CWORD56__BOOT_MODE_TYPE	_CWORD56_BootMode	
ESMBootModeInfo	hostBootMode	

struct T_SS_SM_STOP_DataStruct

Data passed as part of SS_SM_STOP '_CWORD78_OnStop'/'evStop' message *

Class Members:

EPWR_USER_MODE_TYPE	lastUserMode	
EPWR_SHUTDOWN_TRIGGER_TYPE	shutdownTrigger	

struct T_SS_SM_START_ExtDataStructType

Extended Parameter for _CWORD33_OnStart *

Class Members:

BOOL	isMapDiffUpdated	
BOOL	isMapUpdated	
BOOL	isProgUpdated	
EMRelaunchStatus	relaunchStatus	
uint8_t	reserved[SS_SM_START_EXT_RSV_SIZE]	

struct T_SS_SM_STOP_ExtDataStructType

Extended Parameter for _CWORD33_OnStop *

Class Members:

BOOL	isProgUpdated	
uint8_t	reserved[SS_SM_STOP_EXT_RSV_SIZE]	

struct T_SystemManagerLoggingRequestInfo

Error Event Logging Info *.

Class Members:

eSMUserLogType	logType	
CHAR	messageStr[SS_SM_LOG_MSG_STR_SIZE]	
CHAR	suffixStr[SS_SM_SUFFIX_STR_SIZE]	

struct T_SystemManagerClearLogsInfo

Error Event Clear Logs Info *.

Class Members:

void *	rsv	

struct T_SMRequestMsg

Message data payload structure from system manager to heartbeat

Class Members:

CHAR	pstModuleName[MAX_QUEUE_NAME_SIZE]	

struct _SS_SMCurrentState

Class Members:

CHAR	respMsgString[SS_SM_MAX_RESP_MSG_SIZE]	
CHAR	testCaseIdString[SS_SM_MAX_TEST_CASE_NAME_SIZE]	

struct _SMCompleteAck

Class Members:

CHAR	szServiceName[MAX_QUEUE_NAME_SIZE]	
UI_16	unSessionId	

struct _SMGroupLaunchTrigger

Class Members:

UI_32	NextGroupId	

Macro Definition Documentation

#define _CWORD33_LOG_RECEIVED_FROM(hApp) _CWORD33_LOG(ZONE_INFO, __FUNCTION__, " Received from %s", _CWORD33_GetMsgSrc(hApp))

Log whom we received message or notification from.

Use this macro to ensure that the string "Received from" is uniformly logged; this string can be grepped on to find when/where callback functions were invoked.

#define _CWORD33_LOG_WHEN_COMPILED

Value:_CWORD33_LOG(ZONE_INFO, __FUNCTION__, " %s was compiled at %s @ %s", \
__FILE__, __DATE__, __TIME__);

Log when this file was compiled.

Useful when overlaying a delivered object file (from an official build) with a debug-built obj file - verifies that the debug file did indeed get loaded.

#define CALL_AND_LOG_STATUS(fnc)

Value:l_eStatus = (fnc); \
__LOG_STATUS(l_eStatus, #fnc)

Call the function and log the returned E_CWORD33_Status.

#define CALL_AND_LOG_STATUS_IF_ERRORED(fnc)

Value:l_eStatus = (fnc); \
  LOG_STATUS_IF_ERRORED(l_eStatus, #fnc)

Call the function and log the returned E_CWORD33_Status on failure.

#define LOG_ERROR(pStr)

Value:_CWORD33_LOG(ZONE_ERR, __FUNCTION__ \
  , " Error: %s errored: %d/'%s'" \
  , pStr \
  , l_eStatus \
  , GetStr(static_cast<E_CWORD33_Status>(l_eStatus)).c_str());

Log pStr and l_eStatus

#define LOG_STATUS(l_eStatus, pStr)

Value:if (e_CWORD33_StatusOK != l_eStatus) \
 { \
  LOG_ERROR(pStr); \
 } \
 else \
 { \
  LOG_SUCCESS(pStr); \
 }

Log pStr and success or failure. Include l_eStatus when not successful.

#define LOG_STATUS_IF_ERRORED(l_eStatus, pStr)

Value:if (e_CWORD33_StatusOK != l_eStatus) \
 { \
  LOG_ERROR(pStr); \
 }

Log pStr on failure, including E_CWORD33_Status.

#define MAP_ENTRY(f_map, f_enum) f_map[f_enum] = #f_enum

Simplify initializing string map entry.

Use to set a map entry's key and value to the specified enum and the enum's literal text (i.e., stringified) equivalent.

#define NTFY_SSSystemMgrShutdownStarted SERVICE_SYSMANAGER"/ShutdownStarted"

Notification used by HMI to know when a Shutdown Request has been received from the *CWORD56*

#define SS_ASERT(x)

Value:if (!(x)) { \
  _CWORD33_LOG(ZONE_ERR, __FUNCTION__, "SS_ASSERT"); \
 }

#define SS_ASERT_ERRNO(x)

Value:if (!(x)) { \
_CWORD33_LOG(ZONE_ERR, __FUNCTION__, "SS_ASSERT %d:%s", errno, strerror(errno)); \
}

#define SS_ASERT_LOG(x, fmt, ...)

Value:if (!(x)) { \
_CWORD33_LOG(ZONE_ERR, __FUNCTION__, "SS_ASSERT " fmt, ## __VA_ARGS__); \
}

#define SS_STATIC_ASERT(expr)

Value:{ \
char STATIC_ASSERTION_FAILED[(expr) ? 1 : -1]; \
(void)STATIC_ASSERTION_FAILED; \
}

#define szNTFY_PowerPopup SERVICE_POWER"/PowerPopup"

Notification that Power Service publishes for subscribers (ie, HMI) to display a popup indicating the type of CPMShowPowerPopup.

1. Normal
2. Critical
3. Application Critical

#define szNTFY_ShutdownPopup SERVICE_POWER"/ShutdownPopup"

Notification that Power Service publishes for subscribers (ie, HMI) to display a popup indicating the type of shutdown condition.

1. Power Save
2. BatteryCut
3. Low Voltage

Typedef Documentation

typedef enum _SS_SystemManagerProtocol SS_SystemManagerProtocol

System control protocol type.

typedef struct T_SS_SM_START_DataStruct T_SS_SM_START_DataStructType

Parameters of the following message.
SS_SM_START '_CWORD33_OnStart'/'evStart'
SS_SM_PRE_START '_CWORD33_OnPreStart'/'evPreStart'
SS_SM_BACKGROUND_START '_CWORD33_OnBackgroundStart'/'evBackgroundStart'

typedef struct T_SS_SM_STOP_DataStruct T_SS_SM_STOP_DataStructType

Data passed as part of SS_SM_STOP '_CWORD78_OnStop'/'evStop' message *

typedef struct T_SS_SM_UserModeOnOffNotification_Struct T_SS_SM_UserModeOnOffNotification_StructType

Data passed in the NTFY_SSSystemMgrPowerOnOff and * NTFY_SSSystemMgrUserMode notification messages *

typedef struct T_SMRequestMsg TSMRequestMessage

Message data payload structure from system manager to heartbeat

typedef struct T_SystemManagerClearLogsInfo TSystemManagerClearLogsInfo

Error Event Clear Logs Info *.

typedef struct T_SystemManagerLoggingRequestInfo TSystemManagerLoggingRequestInfo

Error Event Logging Info *.

Enumeration Type Documentation

enum _SS_SystemManagerProtocol

System control protocol type.

Enumerator

SS_SM_NOTIFY_SYSTEM_LAUNCH_COMPLETE System Manager to Power manager.

SS_SM_START Normal-boot start request.

SS_SM_START_COMPL_RSPN Normal-boot start completion notification.

SS_SM_STOP Shutdown request.

SS_SM_STOP_COMPL_RSPN Shutdown completion notification.

SS_SM_WAKEUP_MODULES_CMPL_RSPN Response to request of power state transition to power_service.(message format to use: wakeInfo)

SS_SM_SHUTDOWN_MODULES_CMPL_RSPN SM to power.

SS_SM_WAKEUP_MODULES client to service

SS_SM_POWER_OFF_MODULES client to service
SS_SM_SHUTDOWN_MODULES client to service
SS_SM_CRNT_STATE_QUERY client to service
SS_SM_CRNT_STATE_QUERY_RSPN service to client
SS_SM_SYSTEM_MODE_INFO_REQ client to service
SS_SM_SYSTEM_MODE_INFO_RSPN service to client
SS_SM_INITCOMP_REP client to service
SS_SM_GET_START_EXT_INFO _CWORD33_OnStart Extended Parameter
SS_SM_GET_STOP_EXT_INFO _CWORD33_OnStop Extended Parameter
SS_SM_BOOT_MODE_SET_REQ Client to Service (Sys Mgr)
SS_SM_BOOT_MODE_SET_RESP Service (Sys Mgr) to Client.
SS_SM_DATA_RESET_MODE_SET_REQ Client to Service (Sys Mgr)
SS_SM_PROG_UPDATE_STATE_SET_REQ Client to Service (Sys Mgr)
SS_SM_FWD_STARTUP_CONFIRMATION_MSG_REQ Pwr Svc to Sys Mgr.
SS_SM_FWD_START_CONFIRMATION_MSG_RESP Sys Mgr to Pwr Svc.
SS_SM_POWER_REQUEST_MSG Startup process start request from power_service.
SS_SM_POWER_REQUEST_MSG_RESP Sys Mgr to Pwr Svc.
SS_SM_USER_MODE_SET_RESP Service (Sys Mgr) to Client (Pwr Svc)
SS_SM_EVENT_ERROR SM to HMI(*CWORD77*)
SS_SM_EVENT_ERROR_TO_SSL SM to SSL.
SS_SM_ERROR_EVENT_LOGGING_START_REQ SSL to SM.
SS_SM_ERROR_EVENT_LOGGING_START_RSPN SM to SSL.
SS_SM_ERROR_EVENT_ARTIFACT_REQ SSL to SM.
SS_SM_ERROR_EVENT_ARTIFACT_RSPN SM to SSL.
SS_SM_ERROR_EVENT_LOGGING_COMPLETE SSL to SM.
SS_SM_ERROR_EVENT_LOGGING_COMPLETE_RSPN SM to SSL.
SS_SM_USER_INVOKED_LOG_REQ Key Shadow to SM.
SS_SM_ERROR_EVENT_EEL_EXPORT_REQ SSL to SM.
SS_SM_ERROR_EVENT__CWORD33__EMMC_LOGS_REQ SSL to SM.
SS_SM_ERROR_EVENT__CWORD33__EMMC_LOGS_RSPN SM to SVC.
SS_SM_ERROR_EVENT_CLR_LOGS_REQ SVC to SM.
SS_SM_ERROR_EVENT_PROCESS_EXIT SM to SM.
SS_SM_ERROR_EVENT_DIAG_LOG_REQ SSL to SM.
SS_SM_ERROR_EVENT_CAN_LOG_REQ SSL to SM.
SS_SM_ERROR_EVENT_DTC_LOG_REQ SSL to SM.

SS_SM_DEBUG_DUMP SM to client.

SS_SM_DEBUG_DUMP_RSPN client to SM

SS_SM_MODULE_RELAUNCH_REQ client to SM

SS_SM_CPU_HIGH_LOAD_DETECTED client to SM

SS_SM_PROPAGATE_SYSTEM_ERROR client to SM

SS_SM__CWORD56__HEARTBEAT_REQ PS to SM.

SS_SM__CWORD56__HEARTBEAT_RSPN SM to PS.

SS_SM_EVENT_USER_DATA_RESET SM to MS.

SS_SM_BOOT_MICRO_RESET_NTF Logging Shadow to SM.

SS_SM_BOOT_MICRO_LOG_REQ SM to Logging Shadow.

SS_SM_BOOT_MICRO_LOG_RSP Logging Shadow to SM aka ENDING_INDEX.

SS_SM_PRE_START Pre-boot(+BA) start request.

SS_SM_PRE_START_COMPL_RSPN Pre-boot(+BA) start completion notification.

SS_SM_PRE_STOP Pre-boot(+BA) stop request.

SS_SM_PRE_STOP_COMPL_RSPN Pre-boot(+BA) stop completion notification.

SS_SM_BACKGROUND_START Background-boot(+BA) start request.

SS_SM_BACKGROUND_START_COMPL_RSPN Background-boot(+BA) start completion notification.

SS_SM_BACKGROUND_STOP Background-boot(+BA) stop request.

SS_SM_BACKGROUND_STOP_COMPL_RSPN Background-boot(+BA) stop completion notification.

enum eSMUserLogType

1 Error Event Logging enumerations *

Enumerator

e_SS_SM_CAPTURE_ALL_LOGS Capture all logs.

e_SS_SM_SCREEN_CAPTURE Capture screenshot.

e_SS_SM_CAPTURE__CWORD33__LOGS Capture application log.

e_SS_SM_USER_FORCE_RESET Capture logs are forcibily reset.

e_SS_SM_CAPTURE_DEV_LOGS Capture device log.

e_SS_SM_CAPTURE_MODULE_LOGS Capture debug log.

e_SS_SM_CAPTURE_DTC_LOGS Capture DTC event log.

e_SS_SM_CAPTURE_NAVI_LOGS Capture Navi event log.

e_SS_SM_CAPTURE_GROUP_RELAUNCH Capture log by Group Relaunch.

enum T_SMHBProtocolMessages

HeatBeat Protocol Command heartbeat service protocol > define all protocol messages in and out of heart beat thread that are pertinent to the functionality offered by heart beat thread.

Enumerator

SS_HEARTBEAT_PRINT_CONNECTIONS client to hb thread

SS_HEARTBEAT_PRINT_STACK client to hb thread

SS_HEARTBEAT_PERIODIC_STATUS_REQ sysmgr to hb thread

SS_HEARTBEAT_PERIODIC_RESP hb thread to sysmgr

SS_HEARTBEAT_START sysmgr to hb thread

SS_HEARTBEAT_STOP sysmgr to hb thread

SS_HEARTBEAT_DELETE_MODULE_ENTRY sysmgr to hb thread

SS_HEARTBEAT_MSG_CATEGORY_REPORT report message

SS_HEARTBEAT_ERROR_DETECTED hb thread to sysmgr

SS_HEARTBEAT_APPEND_MODULE_ENTRY sysmgr to hb thread

SS_HEARTBEAT_AVAIL_CHECK_REQ sysmgr to hb thread

SS_HEARTBEAT_AVAIL_CHECK_RESP sysmgr to hb thread

SS_HEARTBEAT_RESPONSE client to hb thread

SS_HEARTBEAT_REQUEST hb thread to client

Function Documentation

template<typename enumT , void(*)(std::map< enumT, SS_String > &m_strMap) fp> EnumStringMap< enumT, fp >::EnumStringMap ()[inline]

Summary

Default constructor of EnumStringMap class.

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

~EnumStringMap, Initialize

template<typename T > E_CWORD33_Status ReadMsg (HANDLE *hApp*, T & *Data*, ESMRetrieveTypes *f_eRetrieveMethod* = eSMRRelease)

Summary

check the data information of msg.

Parameters:

in	*hApp*	HANDLE - HANDLE Application
in	*Data*	T - The reference to the Data memory location
in	*f_eRetrieveMethod*	ESMRetrieveTypes - The msg retrieval method (release or retain)

T template type

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldParam	Invalid parameter

Preconditions

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method only

See also:

None

template<typename enumT , void(*)(std::map< enumT, SS_String > &m_strMap) fp>
EnumStringMap< enumT, fp >::~EnumStringMap ()[inline]

~

Summary

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

EnumStringMap

Logger_service

struct _TVINnumber

VIN numbers. struct _T_CWORD33_CANMileageInfo

CAN Diagnostic status data. More...

struct _T_CWORD33_ScreenCaptureEvt

Screen Capture Event data. struct _SStatisticalCounter

Statistical counter. More...

struct _SEventLoggerCommonInfo

Event logger common information. More...

struct _SEventLoggerResetInfo

Event logger resets counter information. More...

struct _SEventLoggerEventInfo

Event logger CWORD56 events information. More...

struct _SEventCANLoggerEventInfo

Event logger CAN events information. More...

class LoggerServiceIf

logger_service struct _T_CWORD33_SetLoggerParams

Screen Capture Event data. More...

struct _T_CWORD33_SetLoggerAllParams

struct _TErrorLogger_FolderInfo

union _uEvtLoggerCommonInfo

Event logger resets counter information. More...

struct _stLogEventss_

Logger Event information. More...

struct _T_CWORD33_EventLogPersistBuffer

struct __CWORD62_DateAndTime

CAN current date and time. More...

Macros

#define SHMEM_DRLOCATIONLOG "/DRLocationLog"

Shared Memory names.

#define SHMEM_DRINITIALLOG "/DRInitialLog"

Shared Memory names.

#define MAX_EVT_RECORDS 20000

Macros definition for the Event Logs.

#define MAX_EVENTLOG_SIZE 10*MAX_EVT_RECORDS

Max size for event logging.

#define NTFY_SS_LoggerService_Availability SERVICE_LOGGER"/Availability"

Logger Service Availability Notification.

Typedefs

typedef struct _TVINnumber STVIN_NUMBER

VIN numbers.

typedef struct _T_CWORD33_CANMileageInfo STLOGGER_CANDIAGSTAT

CAN Diagnostic status data.

typedef enum _ELoggerDeviceTypes EDEV_TYPE

Logger Device Type.

typedef enum _ELoggerState ELOGGER_STAT

Logger State.

typedef struct _T_CWORD33_ScreenCaptureEvt STScreenCaptureEvt

Screen Capture Event data.

typedef enum _EEventLoggerUSBDeviceNumber EEvtLogUSBDevNumber

USB source number to store event logs.

typedef enum _EDevNumber EDevNumber

USB source number to store emergency error logs.

typedef enum _ECounterGroupID EStatCounterGroupID

Counter group ID.

typedef struct _SStatisticalCounter SStatisticalCounter

Statistical counter.

typedef enum _EEventLoggerErrorCode EEvtLoggerErrorCode

Event Logger error code.

typedef enum _EEmergencyLogErrorCode EELL_ErrorCode

Emergency Log error code.

typedef struct _SEventLoggerCommonInfo STEventLoggerCommonInfo

Event logger common information.

typedef struct _SEventLoggerResetInfo STEventLoggerResetInfo

Event logger resets counter information.

typedef struct _SEventLoggerEventInfo STEventLoggerEventInfo

Event logger CWORD56 events information.

typedef struct _SEventCANLoggerEventInfo STEventCANLoggerEventInfo

Event logger CAN events information.

typedef struct _T_CWORD33_SetLoggerParams STLoggerSetParams

Screen Capture Event data.

typedef struct _T_CWORD33_SetLoggerAllParams **STLoggerSetAllParams**

typedef struct _TErrorLogger_FolderInfo **STLoggerFolderInfo**

typedef union _uEvtLoggerCommonInfo UEvtLoggerCommonInfo

Event logger resets counter information.

typedef struct _stLogEventss_ st_LogEvent_ss

Logger Event information.

typedef struct _T_CWORD33_EventLogPersistBuffer **STEventLogPersistBuffer**

typedef enum _LoggerServiceProtocol SS_LoggerServiceProtocol

Logger service event IDs.

typedef enum _SS_LOGGERSERVICEPROTOCOL SS_loggerserviceprotocol

Logger service event IDs.

typedef enum _LoggerServerEvents SS_LoggerServerEvents

Private events for the following categories of devices.

Enumerations

enum _ELoggerDeviceTypes { **eDevUSB1**, **eDevUSB2**, **eDevSD**, **eEthernet**, **eTotalDevicesTypes**, **eInvalid_DevType** }*Logger Device Type.*

enum _ELoggerState { **eDeactivate**, **eActivate**, **eInvalid_LoggerState** }*Logger State.*

enum _EEventLoggerUSBDeviceNumber { **USB0** = ***, **USB1** = ***, **SD** = *** }*USB source number to store event logs.*

enum _EDevNumber { **eEEL_USB1** = ***, **eEEL_USB2** = ***, **eEEL_SD** = *** }*USB source number to store emergency error logs.*

enum _ECounterGroupID { **STARTUP_SHUTDOWN_COUNTER_GROUP** = ***, **_CWORD105__EVENTS_COUNTER_GROUP**, **APP_USAGE_COUNTER_GROUP**, **F_BLK_STABILITY_COUNTER_GROUP**, **MAX_COUNTER_GROUP** }*Counter group ID.*

enum _EEventLoggerErrorCode { **USB_DEVICE_NOT_AVAILABLE** = ***, **USB_DEVICE_WRITE_ERROR** = ***, **CLEAR_EVENT_LOG_FAILED** = ***, **STATISTICAL_COUNTER_READ_FAILED** = ***, **NO_ERROR_INFO** = *** }*Event Logger error code.*

enum _EEmergencyLogErrorCode { **eDEVICE_NOT_AVAILABLE** = ***, **eDEVICE_WRITE_ERROR** = ***, **eNO_ERROR_INFO** = *** }*Emergency Log error code.*

enum _LoggerServiceProtocol { **SS_LOGGER_MILEAGE_DATA** = ***, **SS_LOGGER_SET_PARAMS**, **SS_LOGGER_STORE_SCREENCAPTURE_AND_LOG**, **SS_LOGGER_STORE_SCREENCAPTURE**, **SS_LOGGER_STORE_LOG**, **SS_LOGGER_SCREENCAPTURE_EVT_ACK**, **SS_LOGGER_EVENT_COMMONINFO**, **SS_LOGGER__CWORD56__EVENT_INFO**, **SS_LOGGER__CWORD56__RESET_INFO**, **SS_LOGGER_SET_DATETIME**, **SS_LOGGER_SET_VIN**, **SS_LOGGER_UDP_LOGGING**, **SS_LOGGER_ERROR_EVENT_ARTIFACT_REQ**, **SS_LOGGER_ERROR_EVENT_ARTIFACT_RESP**, **SS_LOGGER__CWORD105_SHUTDOWN_COMPLETE**, **SS_LOGGER_DIAGDTC_ACTIVE**, **SS_LOGGER_SET_DIAGID**, **SS_LOGGER_ERROR_EVENT_TIMER_ID_LOGGING_START_RSPN**, **SS_LOGGER_ERROR_EVENT_TIMER_ID_ARTIFACT_RESPONSE**, **SS_LOGGER_ERROR_EVENT_TIMER_ID_SCREEN_CAPTURE_RSPN** }*Logger service event IDs.*

enum _SS_LOGGERSERVICEPROTOCOL { **SS_LOGGERCOPYEVENTUSB** = ***, **SS_LOGGERCOPYEVENTUSB_SUCCESS_RESP**, **SS_LOGGERCOPYEVENTUSB_ERROR_RESP**, **SS_LOGGERCOPYEMERGENCYLOGS**, **SS_LOGGERCOPYEMERGENCYLOGS_SUCCESS_RESP**, **SS_LOGGERCOPYEMERGENCYLOGS_ERROR_RESP**, **SS_LOGGERCLEAREVENT**, **SS_LOGGERCLEAREVENT_SUCCESS_RESP**, **SS_LOGGERCLEAREVENT_ERROR_RESP**, **SS_LOGGER_READ_STATL_COUNTER**, **SS_LOGGER_READ_STATL_COUNTER_SUCCESS_RESP**, **SS_LOGGER_READ_STATL_COUNTER_ERROR_RESP**, **SS_LOGGER_RESET_STATL_COUNTER**, **SS_LOGGER_RESET_STATL_COUNTER_SUCCESS_RESP**, **SS_LOGGER_RESET_STATL_COUNTER_ERROR_RESP**, **SS_LOGGER_ENG_READ_NUMOFEVENTS**, **SS_LOGGER_ENG_READ_NUMOFEVENTS_RESP**, **SS_LOGGER_ENG_READ_STATISTICAL_COUNTERS**, **SS_LOGGER_ENG_READ_STATISTICAL_COUNTERS_RESP**, **SS_LOGGER_UPLOAD_EVENTLOG**, **SS_LOGGER_UPLOAD_EVENTLOG_RESP** }*Logger service event IDs.*

enum _LoggerServerEvents { **SS_LOGGER_SCREENCAPTURE_EVT** = ***, **SS_LOGGER_ERRORINFO_EVT**, **SS_LOGGER_LOGINFO_EVT**, **SS_LOGGER_LOGSTARTED_EVT** }*Private events for the following categories of devices.*

Functions

E_CWORD33_Status SS_LoggerStoreLogs (SS_STORELOGS_OPE_TYPE type)

SSLoggerSrvIfWriteDebugLogs.

Detailed Description

Class Documentation

struct _T_CWORD33_CANMileageInfo

CAN Diagnostic status data.

Class Members:

UI_8	DidA_ExtTest_Pres	
UI_8	EngRun_Stat	
UI_8	Odo__CWORD34__H	
UI_8	Odo__CWORD34__L	
UI_8	Odo_LSB_H	
UI_8	Odo_LSB_L	
UI_8	PN14_SupBat_Volt	

struct _SStatisticalCounter

Statistical counter.

Class Members:

UI_8	StatisticalCountBuffer[MAX_STATISTICAL_BUFFER]	Counter values for startup,normal and shutsown phases.
UI_16	u16NormalCounterLength	No of Counter from Normal phase.
UI_16	u16ShutDownCounterLength	No of Counter from Shut down phase.
UI_16	u16StartupCounterLength	No of Counter from startup phase.

struct _SEventLoggerCommonInfo

Event logger common information.

Class Members:

UI_8	BodyCAN_Stat:4	
UI_8	FOT_Temp:2	
UI_8	HeadUnitCAN_Stat:4	
UI_8	HMIInteraction:4	
UI_8	IGN_Status:4	
UI_8	Syst_CWORD61_oltage:6	

struct _SEventLoggerResetInfo

Event logger resets counter information.

Class Members:

UI_8	_CWORD102__ResetInfo	
UI_8	_CWORD56__ResetInfo	

struct _SEventLoggerEventInfo

Event logger *CWORD56* events information.

Class Members:

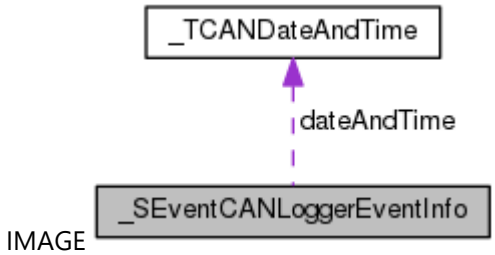
UI_8	EventData[4]	
UI_8	EventGroup	
UI_8	EventIdentifier	
UI_8	NumberOfEvents	

struct _SEventCANLoggerEventInfo

Event logger CAN events information.

Collaboration diagram for _SEventCANLoggerEventInfo:

IMAGE



Class Members:

STCanCurrentDateTime	dateAndTime	
BOOL	success	
UI_32	triggerNumber	
BOOL	valid	

struct _T_CWORD33_SetLoggerParams

Screen Capture Event data.

Class Members:

EDEV_TYPE	Device_Type	
ELOGGER_STAT	Logger_State	

struct _T_CWORD33_SetLoggerAllParams

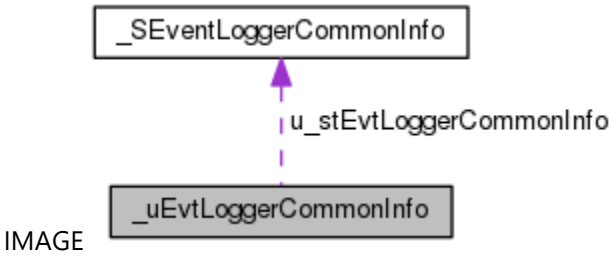
Class Members:

EDEV_TYPE	Device_Type	
UI_32	Log_FolderCnt	
ELOGGER_STAT	Logger_State	
ELOGGER_STAT	Logger_UDPState	

union _uEvtLoggerCommonInfo

Event logger resets counter information.

Collaboration diagram for _uEvtLoggerCommonInfo:



IMAGE

Class Members:

UI_8	CommonData[4]	
STEventLoggerCommonInfo	u_stEvtLoggerCommonInfo	

struct _stLogEventss_

Logger Event information.

Class Members:

UI_8	data[4]	
UI_8	event_id	
UI_8	grp_ID	
UI_32	ts	

struct __CWORD62_DateAndTime

CAN current date and time.

CAN current date and time

Class Members:

UI_8	DateTime_Stat	Date time status
UI_8	DateTimeDay	Date
UI_8	DateTimeHour	Hour
UI_8	DateTimeMinute	Minute
UI_8	DateTimeMonth	Month
UI_8	DateTimeSecond	Second
UI_8	DateTimeYear	Year
UI_8	TimeFormat	Format

Function Documentation

E_CWORD33_Status SS_LoggerStoreLogs (SS_STORELOGS_OPE_TYPE *type*)

SSLoggerSrvIfWriteDebugLogs.

Returns:

Status E_CWORD33_Status - success or error

Power_service

struct _upinfo
struct _wakeinfo
struct _SystemModeInfo
struct StartupConfirmationMsgStrut
struct PowerRequestMsgStrut
struct PowerRequestMsgStrutWithUMCR
struct ShutdownRequestMsgStrut
struct EPWR_HB_REQ_MSG_STRUCT
struct SS_Pwr_CpuResetMsgStruct
struct Pwr_ServiceSetInterface

Macros

#define szNTFY_PowerAvailability SERVICE_POWER"/Availability"

Indicated if Power Service is available or not portion of this message is BOOL.

Typedefs

typedef enum _PwLevelTypes **PowerSrvLevelType**
typedef enum _PwLevelTypes **EPWR_LEVEL_TYPE**
typedef enum PowerSrvSessionType **EPWR_SESSION_TYPE**
typedef enum ePowerSrvCANStates **EPWR_CAN_STATE_TYPE**
typedef enum ePowerSrvWakeupLevels **EPWR_WAKEUP_LEVEL_TYPE**
typedef EPWR_WAKEUP_LEVEL_TYPE * **PEPWR_WAKEUP_LEVEL_TYPE**
typedef enum ePowerSrvPowerStates **EPWR_POWER_STATE_TYPE**
typedef enum ePowerSrvVoltageStates **EPWR_VOLTAGE_STATE_TYPE**
typedef EPWR_VOLTAGE_STATE_TYPE * **PEPWR_VOLTAGE_STATE_TYPE**
typedef enum ePowerSrvCrankStates **EPWR_CRANK_STATE_TYPE**
typedef enum ePowerSrvWakeupFactors **EPWR_WAKEUP_FACTOR_TYPE**
typedef EPWR_WAKEUP_FACTOR_TYPE * **PEPWR_WAKEUP_FACTOR_TYPE**
typedef enum ePwrServiceShutdownPopupType **EPWR_SHUTDOWN_POPUP_TYPE**
typedef enum ePwrServicePowerPopupType **EPWR_POWER_POPUP_TYPE**
typedef enum ePwrServiceLHCType **EPWR_LHC_TYPE**
typedef enum ePwrServiceProdModeType **EPWR_PROD_MODE_TYPE**
typedef enum ePwrServiceTransportModeType **EPWR_TRANSPORT_MODE_TYPE**
typedef enum ePwrServiceUserModeType **EPWR_USER_MODE_TYPE**
typedef enum ePwrServiceUserModeChangeReasonType
EPWR_USER_MODE_CHANGE_REASON_TYPE
typedef struct _upinfo **upInfo**
typedef struct _wakeinfo **wakeInfo**
typedef enum ePowerSrvSystemModeInfo **EPWR_SYSTEM_MODE_INFO**
typedef enum ePowerSrvStartupStageType **EPWR_STARTUP_STAGE_TYPE**
typedef struct _SystemModeInfo **SystemModeInfo**
typedef struct StartupConfirmationMsgStrut **EPWR_SC_MSG_STRUCT**
typedef enum ePowerSrv_CWORD102_StartupInfoType
EPWR__CWORD102__STARTUP_INFO_TYPE
typedef enum ePowerSrvPwrReqMsgCIDIOpeningType **EPWR_OPENING_TYPE**
typedef enum ePowerSrvPwrManResetInfoType **EPWR_MAN_RESET_INFO_TYPE**
typedef enum ePowerSrvTypeLastSysRetReq **EPWR_LAST_SYS_RET_REQ_TYPE**
typedef struct PowerRequestMsgStrut **EPWR_POWER_REQUEST_MSG_STRUCT**
typedef struct PowerRequestMsgStrutWithUMCR
EPWR_POWER_REQUEST_MSG_STRUCT_WITH_UMCR
typedef enum ePowerSrvONSType **EPWR_ONS_TYPE**
typedef enum ePowerSrvPwrShutdownTriggerType **EPWR_SHUTDOWN_TRIGGER_TYPE**
typedef struct ShutdownRequestMsgStrut **EPWR_SHUTDOWN_REQUEST_MSG_STRUCT**

Enumerations

enum **_PwLevelTypes** { **epspltUNKNOWN**, **epspltWAKEUP** = ***, **epspltNORMAL**,
epspltSHUTDOWN, **epspltEMERGENCY** }

enum **PowerSrvSessionType** { **epsstUNKNOWN**, **epsstBASIC**, **epsstSUPERVISOR**, **epsstSYSTEM** }
enum **ePowerSrvCANStates** { **epscnINVALID** = ***, **epscnCANWAKEUP** = ***, **epscnCANSLEEP** = *** }
enum **ePowerSrvWakeupLevels** { **epswlINVALID** = ***, **epswlFULLRUN** = ***, **epswlLOCALWAKEUP** = *** }
enum **ePowerSrvPowerStates** { **epswsINVALID** = ***, **epswsPWRON** = ***, **epswsPWROFF** = *** }
enum **ePowerSrvVoltageStates** { **epsvsINVALID** = ***, **epsvsNORMAL** = ***, **epsvsLVI1** = ***, **epsvsLVI2** = *** }
enum **ePowerSrvCrankStates** { **epscsINVALID** = ***, **epscsENTRY** = ***, **epscsEXIT** = *** }
enum **ePowerSrvWakeupFactors** { **epswfINVALID** = ***, **epswfTESTACC** = ***, **epswfON_KEY** = ***, **epswfIGN_ACC** = ***, **epswfDR_OPEN_CLOSE** = ***, **epswfDX_ACTIVATION** = ***, **epswfETHERNET** = ***, **epswfPASS_ACTIVATION** = ***, **epswfSPVACTIVATION** = ***, **epswfUSER_DATA_RESET** = *** }
enum **ePwrServiceShutdownPopupType** { **epsspPowerSave1** = ***, **epsspPowerSave2** = ***, **epsspPowerSave3** = ***, **epsspPowerSaveClr** = ***, **epsspLowVoltage1** = ***, **epsspLowVoltage2** = ***, **epsspLowVoltage3** = ***, **epsspLowVoltageClr** = ***, **epsspBattCouplingSW1** = ***, **epsspBattCouplingSW2** = ***, **epsspBattCouplingSW3** = ***, **epsspBattCouplingSWClr** = ***, **epsspAbnormalTemp_1st** = ***, **epsspAbnormalTemp_Clr** = ***, **epsspLimpHome_1st** = ***, **epsspLimpHome_2nd** = ***, **epsspLimpHome_3rd** = ***, **epsspLimpHome_Clr** = ***, **epsspProdMd_1st** = ***, **epsspProdMd_Clr** = ***, **epsspTransMd_1st** = ***, **epsspTransMd_Clr** = ***, **epsspAllClr** = *** }
enum **ePwrServicePowerPopupType** { **epssppNormal** = ***, **epssppCritical** = ***, **epssppAppCritical** = ***, **epssppAllClr** = *** }
enum **ePwrServiceLHCType** { **epslhcINVALID** = ***, **epslhcDISABLED** = ***, **epslhcENABLED** = *** }
enum **ePwrServiceProdModeType** { **epspmINVALID** = ***, **epspmDISABLED** = ***, **epspmENABLED** = *** }
enum **ePwrServiceTransportModeType** { **epstmINVALID** = ***, **epstmDISABLED** = ***, **epstmENABLED** = *** }
enum **ePwrServiceUserModeType** { **epsumINVALID** = ***, **epsumOFF** = ***, **epsumON** = *** }
enum ePwrServiceUserModeChangeReasonType { epsumcrNOT_AVAILABLE = ***, **epsumcrON_KEY** = ***, epsumcrPARKING_B, epsumcrPRE_BA, epsumcrNORMAL, epsumcrBACKGROUND_BA }
enum **ePowerSrvSystemModeInfo** { **epssinfINVALID** = ***, **epssinfNORMAL** = ***, **epssinfPROGRAMMING** = *** }
enum **ePowerSrvStartupStageType** { **epssusfINVALID** = ***, **epssusSYSTEM_SERVICES_STARTED** = ***, **epssusALL_SERVICES_LAUNCHED** = *** }
enum **EPWR_SC__CWORD56__BOOT_MODE_TYPE** { **eps_CWORD56_bmINVALID** = ***, **eps_CWORD56_bmAPPLICATION_MODE** = ***, **eps_CWORD56_bmPROGRAMMING_MODE** = *** }
enum **EPWR_SC_WAKEUP_TYPE** { **epsstINVALID** = ***, **epsstWARMSTART** = ***, **epsstCOLDSTART** = *** }
enum **EPWR_SC_COLD_START_REQ_TYPE** { **epsscrtINVALID** = ***, **epsscrtNOT_REQUIRED** = ***, **epsscrtREQUIRED** = *** }

enum **EPWR_SC_SECURITY_STATUS** { **epsssINVALID** = ***, **epsssUNLOCK** = ***, **epsssLOCK** = *** }
enum ePowerSrv_CWORD102_StartupInfoType { **epsprm_CWORD102_siINVALID** = ***, epsprm_CWORD102_si_CWORD102__STARTUP_NORMAL = ***, epsprm_CWORD102_si_CWORD102__STARTUP_AFTER_RESET = *** }
enum ePowerSrvPwrReqMsgCIDIOpeningType { **epsprmotINVALID** = ***, epsprmotTIMING_TYPE_1 = ***, epsprmotTIMING_TYPE_2 = *** }
enum ePowerSrvPwrManResetInfoType { **epsprmriINVALID** = ***, **epsprmriNORMAL** = ***, **epsprmriWAKEUP_AFTER_RESET** = *** }
enum ePowerSrvTypeLastSysRetReq { **epsprlsrrINVALID** = ***, **epsprlsrrLAST_SYSTEM_OFF** = ***, **epsprlsrrLAST_SYSTEM_ON** = *** }
enum **ePowerSrvONSType** { **epssdmonsINVALID** = ***, **epssdmonsNO_ONS** = ***, **epssdmonsONS** = *** }
enum **ePowerSrvPwrShutdownTriggerType** { **epssdmsdtINVALID** = ***, **epssdmsdtTESTACC_OFF** = ***, **epssdmsdtON_KEY** = ***, **epssdmsdtIGN_LOCK** = ***, **epssdmsdtPWR_SAVE** = ***, **epssdmsdtTMP_STARTUP** = ***, **epssdmsdtDIAG_DEACTIVATION** = ***, **epssdmsdtABNORMAL_VOLTAGE** = ***, **epssdmsdtABNORMAL_TEMP** = ***, **epssdmsdtBATTERYCUTOFF** = ***, **epssdmsdtLIMPHOME** = ***, **epssdmsdtHU_CAN_ERROR** = ***, **epssdmsdtBODY_CAN_ERROR** = ***, **epssdmsdtTRANSPORT_MODE** = ***, **epssdmsdtPRODUCTION_MODE** = ***, **epssdmsdtIGN_OFF** = ***, **epssdmsdtGENERIC_ERROR_RESET** = ***, **epssdmsdtFATAL_ERROR_RESET** = ***, **epssdmsdtUSER_DATA_RESET** = ***, **epssdmsdtFACTORY_DATA_RESET** = ***, **epssdmsdtFAST_SLEEP_MODE** = ***, **epssdmsdtNORMAL_RESET** = ***, **epssdmsdtPROGUPDATE_RESET** = *** }
enum **epsCpuResetReason** { **epsCpuResetReasonGeneric** = ***, **epsCpuResetReasonFatalError**, **epsCpuResetReasonUserDataReset**, **epsCpuResetReasonFactoryDataReset**, **epsCpuResetReasonUserForceReset**, **epsCpuResetReasonShipmentModeReset**, **epsCpuResetReasonHeadUnitReset**, **epsCpuResetReasonNormalReset**, **epsCpuResetReasonProgupdateReset** }

Functions

HANDLE OpenPowerService (HANDLE f_hApp)
E_CWORD33_Status ClosePowerService (HANDLE f_hApp, HANDLE f_hService)
E_CWORD33_Status PwrServiceOpenSessionRequest (HANDLE f_hService, EPWR_SESSION_TYPE f_eSessionType)
E_CWORD33_Status PwrServiceCloseSessionRequest (HANDLE f_hService, HANDLE f_hSession)
E_CWORD33_Status PwrServiceDecodeOpenSessionResponse (HANDLE f_hApp, HANDLE &f_hSession)
E_CWORD33_Status PwrServiceSetVoltageState (HANDLE f_hSession, EPWR_VOLTAGE_STATE_TYPE f_eVoltage_state)
E_CWORD33_Status PwrServiceSetCrankState (HANDLE f_hSession, EPWR_CRANK_STATE_TYPE f_eCrank_state)
E_CWORD33_Status PwrServiceSystemModeInfoRequest (HANDLE f_hSession)
E_CWORD33_Status PwrServiceSendInitCompReport (HANDLE f_hSession)
E_CWORD33_Status PwrServiceSendSetShutdownPopupRequest (HANDLE f_hSession, EPWR_SHUTDOWN_POPUP_TYPE f_eShutdownPopup)

E_CWORD33_Status PwrServiceSendStartupConfirmationMsgRequest (HANDLE f_hSession,
EPWR_SC_MSG_STRUCT f_eState)
E_CWORD33_Status PwrServiceSendPowerRequest (HANDLE f_hSession,
EPWR_POWER_REQUEST_MSG_STRUCT_WITH_UMCR &f_eState)
E_CWORD33_Status PwrServiceSendShutdownRequest (HANDLE f_hSession,
EPWR_SHUTDOWN_REQUEST_MSG_STRUCT &f_eState)
E_CWORD33_Status PwrServiceSendHeartBeatRequest (HANDLE f_hSession,
EPWR_HB_REQ_MSG_STRUCT *f_eHbReqMsg)

Detailed Description

Class Documentation

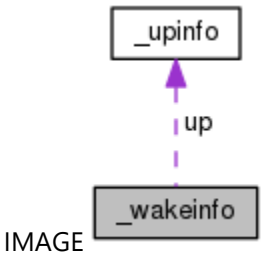
struct _upinfo

Class Members:

EPWR_WAKEUP_FACTOR_TYPE	factor	
EPWR_WAKEUP_LEVEL_TYPE	level	
EPWR_USER_MODE_CHANGE_REASON_TYPE	userModeChangeReason	State transition reason.

struct _wakeinfo

Collaboration diagram for _wakeinfo:



IMAGE

Class Members:

EPWR_POWER_STATE_TYPE	powerupType	
upInfo	up	

struct _SystemModeInfo

Class Members:

EPWR_USER_MODE_TYPE	lastUserMode	
EPWR_LHC_TYPE	limpHomeCutoff	

EPWR_PROD_MODE_TYPE	productionMode	
EPWR_STARTUP_STAGE_TYPE	startupStage	
EPWR_SYSTEM_MODE_INFO	systemMode	
EPWR_TRANSPORT_MODE_TYPE	transportMode	

struct StartupConfirmationMsgStrut

Class Members:

EPWR_SC__CWORD56__BOOT_MODE_TYPE	_CWORD56_BootMode	
EPWR_SC_COLD_START_REQ_TYPE	coldStartRequest	
UI_32	HWVersion	
UI_32	imageType	
UI_32	majorVersion	
UI_32	matchedHardwareType	
UI_32	minorVersion	
EPWR_SC_SECURITY_STATUS	securityStatus	
UI_32	softwareType	
UI_32	SWVersionPatchLevel	
UI_32	SWVersionWeek	
UI_32	SWVersionYear	
EPWR_SC_WAKEUP_TYPE	wakeupType	

struct PowerRequestMsgStrut

Class Members:

EPWR_LAST_SYS_RET_REQ_TYPE	lastSystemRetentionReq	
EPWR_USER_MODE_TYPE	lastUserMode	
EPWR_MAN_RESET_INFO_TYPE	manResetInfo	
EPWR_OPENING_TYPE	openingType	
EPWR_WAKEUP_FACTOR_TYPE	startupReason	
EPWR__CWORD102__STARTUP_INFO_TYPE	startupType	
EPWR_USER_MODE_TYPE	userMode	

struct PowerRequestMsgStrutWithUMCR

structure for power request message

Class Members:

EPWR_LAST_SYS_RET_REQ_TYPE	lastSystemRetentionReq	
EPWR_USER_MODE_TYPE	lastUserMode	
EPWR_MAN_RESET_INFO_TYPE	manResetInfo	
EPWR_OPENING_TYPE	openingType	
EPWR_WAKEUP_FACTOR_TYPE	startupReason	
EPWR__CWORD102__STARTUP_INFO_TYPE	startupType	
EPWR_USER_MODE_TYPE	userMode	

EPWR_USER_MODE_CHANGE_REASON_TYPE	userModeChangeReason	State transition reason.

struct ShutdownRequestMsgStrut

Class Members:

EPWR_USER_MODE_TYPE	lastUserMode	
EPWR_LHC_TYPE	limpHomeCutoffRequest	
EPWR_ONS_TYPE	ONS_Type	
EPWR_PROD_MODE_TYPE	productionMode	
EPWR_SHUTDOWN_TRIGGER_TYPE	shutdownTrigger	
EPWR_TRANSPORT_MODE_TYPE	transportMode	

struct EPWR_HB_REQ_MSG_STRUCT

Class Members:

UI_8	IntervalSec	

struct SS_Pwr_CpuResetMsgStruct

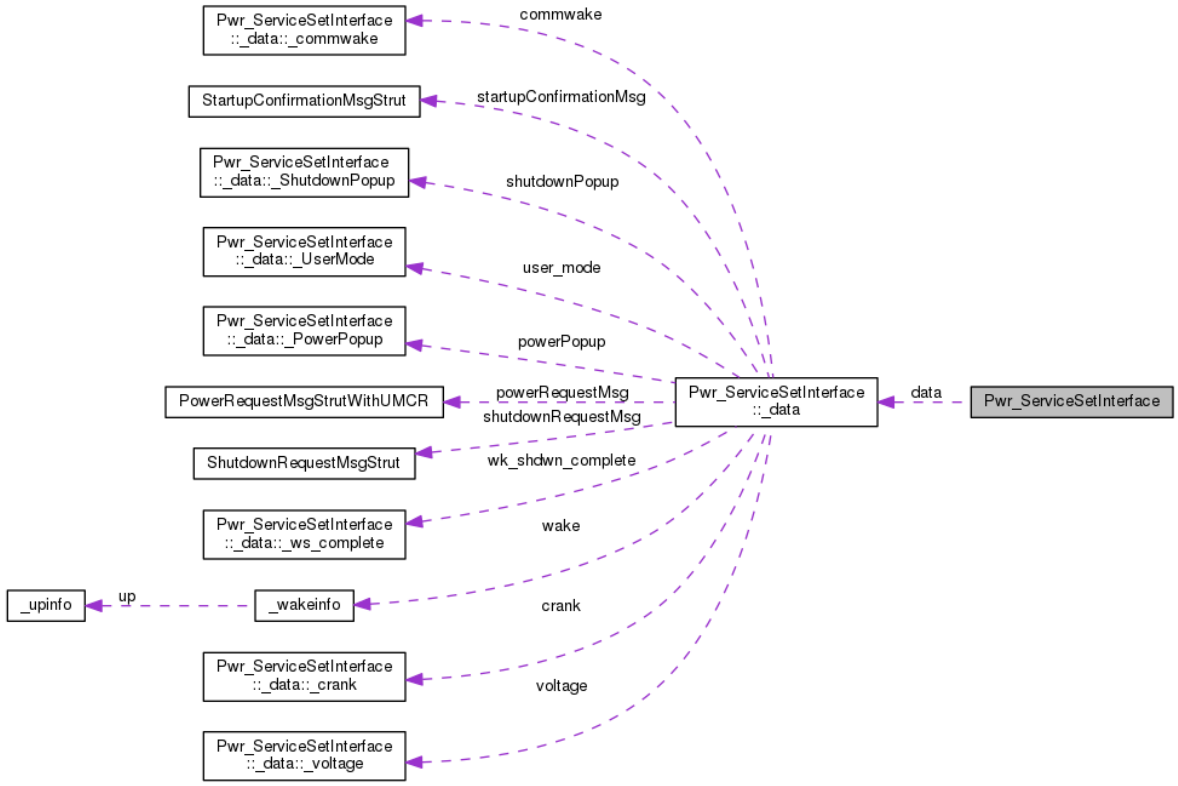
Class Members:

CHAR	messageStr[PWR_RESET_MSG_STR_SIZE]	
epsCpuResetReason	resetReason	

struct Pwr_ServiceSetInterface

Collaboration diagram for Pwr_ServiceSetInterface:

IMAGE



Class Members:

union _data	data	

Enumeration Type Documentation

enum ePowerSrv_CWORD102_StartupInfoType

CWORD102 Startup Information

Enumerator

epsprm_CWORD102_si_CWORD102__STARTUP_NORMAL Normal Startup.

epsprm_CWORD102_si_CWORD102__STARTUP_AFTER_RESET Startup after Irregular Reset.

enum ePowerSrvPwrManResetInfoType

Manual Reset Information

enum ePowerSrvPwrReqMsgCIDIOpeningType

Opening Type

Enumerator

epsprmotTIMING_TYPE_1 CI/DI Show timing type1.

epsprmotTIMING_TYPE_2 CI/DI Show timing type2.

enum ePowerSrvTypeLastSysRetReq

Last System Retention Request

enum ePwrServiceUserModeChangeReasonType

State transition reason.

Enumerator

epsumcrNOT_AVAILABLE Not available.

epsumcrPARKING_B Parking(+B) Power state transition.

epsumcrPRE_BA Pre-boot(+BA) Power state transition.

epsumcrNORMAL Normal-boot Power state transition.

epsumcrBACKGROUND_BA Background-boot(+BA) Power state transition.

Function Documentation

E_CWORD33_Status ClosePowerService (HANDLE *f_hApp*, HANDLE *f_hService*)

Summary

Free a handle for using PowerService.

Parameters:

in	*f_hApp*	HANDLE - Application handle
in	*f_hService*	HANDLE - PowerService handle

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Sync only

See also:

None

HANDLE OpenPowerService (HANDLE *f_hApp*)

Summary

Obtain a handle for using the PowerService.

Parameters:

in	*f_hApp*	HANDLE - Application handle

Return values:

Handle	Success
NULL	Failed

Classification

Public

Type

Sync only(None communication)

See also:

None

E_CWORD33_Status PwrServiceCloseSessionRequest (HANDLE *f_hService*, HANDLE *f_hSession*)

Summary

End using the PowerService.

Parameters:

in	*f_hService*	HANDLE - PowerService handle
in	*f_hSession*	HANDLE - Session type

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Fire and Forget Method

See also:

None

E_CWORD33_Status PwrServiceDecodeOpenSessionResponse (HANDLE *f_hApp*, HANDLE & *f_hSession*)

Summary

Obtain a handle for OpenSessionRequest response message.

Parameters:

in	*f_hApp*	HANDLE - Application handle
in	*f_hSession*	HANDLE& - Address of storing session handle

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	NULL Pointer

Classification

Public

Type

Sync only(None communication)

See also:

None

E_CWORD33_Status PwrServiceOpenSessionRequest (HANDLE *f_hService*, EPWR_SESSION_TYPE *f_eSessionType*)

Summary

Make the PowerService available.

Parameters:

in	*f_hService*	HANDLE - PowerService handle
in	*f_eSessionType*	EPWR_SESSION_TYPE - Session type

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Fire and Forget Method

See also:

None

E_CWORD33_Status PwrServiceSendHeartBeatRequest (HANDLE *f_hSession*, EPWR_HB_REQ_MSG_STRUCT * *f_eHbReqMsg*)

Summary

Send a request for HeartBeat monitoring start.

Parameters:

in	*f_hSession*	HANDLE - Session handle
in	*f_eHbReqMsg*	EPWR_HB_REQ_MSG_STRUCT* 2 Address of the structure for HeartBeat monitoring start request message

Structure for HeartBeat monitoring start request message

```
1 typedef struct {
2     UI_8  IntervalSec;
3 } EPWR_HB_REQ_MSG_STRUCT;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Method x Method

See also:

None

E_CWORD33_Status PwrServiceSendInitCompReport (HANDLE *f_hSession*)

Summary

Send an initialize complete response.

Parameters:

in	*f_hSession*	HANDLE - Session handle

Return values:

e_CWORD33_StatusOK	Success

e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Fire and Forgeat Fire and Forget

See also:

None

E_CWORD33_Status PwrServiceSendPowerRequest (HANDLE *f_hSession*, EPWR_POWER_REQUEST_MSG_STRUCT_WITH_UMCR & *f_eState*)

Brief

Send power request.

Parameters:

in	*f_hSession*	HANDLE - Session handle
in	*f_eState*	EPWR_POWER_REQUEST_MSG_STRUCT_WITH_UMCR& - Address of the structure for power request message

Of the eight elements of the EPWR_POWER_REQUEST_MSG_STRUCT_WITH_UMCR structure, five elements set fixed values.
startupType = epsprm_CWORD102_si_CWORD102__STARTUP_NORMAL
openingType = epsprmotTIMING_TYPE_1
lastUserMode = epsumON
manResetInfo = epsprmriNORMAL
lastSystemRetentionReq = epsprlsrrLAST_SYSTEM_OFF

For the following, set different values at system startup and shutdown.
(Element name) = (Startup settings or Shutdown settings)
startupReason = (epswfIGN_ACC or epswfINVALID)
userMode = (epsumON or epsumOFF)

For the following, set different values at power state transition.

Element name	Power state	Value
userModeChangeReason		
Not Available		epsumcrNOT_AVAILABLE
		epsumcrON_KEY
Parking(+B)		epsumcrPARKING_B
Pre-boot(+BA)		epsumcrPRE_BA
Normal-boot		epsumcrNORMAL
Background-boot(+BA)		epsumcrBACKGROUND_BA

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusFail	Failed

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Method only

See also:

None

E_CWORD33_Status PwrServiceSendSetShutdownPopupRequest (HANDLE *f_hSession*, EPWR_SHUTDOWN_POPUP_TYPE *f_eShutdownPopup*)

Summary

Obtain shutdown state.

Parameters:

in	*f_hSession*	HANDLE - Session handle
in	*f_eShutdownPopup*	EPWR_SHUTDOWN_POPUP_TYPE - Shutdown state

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Method only

See also:

None

E_CWORD33_Status PwrServiceSendShutdownRequest (HANDLE *f_hSession*, EPWR_SHUTDOWN_REQUEST_MSG_STRUCT & *f_eState*)

Summary

Send shutdown request.

Parameters:

in	*f_hSession*	HANDLE - Session handle
in	*f_eState*	EPWR_SHUTDOWN_REQUEST_MSG_STRUCT& 3 Address of the structure for shutdown request message

Structure for shutdown request message

```
1 typedef struct {
2    EPWR_ONS_TYPE              ONS_Type;
3    EPWR_SHUTDOWN_TRIGGER_TYPE    shutdownTrigger;
4    EPWR_USER_MODE_TYPE          lastUserMode;
5    EPWR_TRANSPORT_MODE_TYPE      transportMode;
6    EPWR_PROD_MODE_TYPE          productionMode;
7    EPWR_LHC_TYPE              limpHomeCutoffRequest;
8 } ShutdownRequestMsgStrut, EPWR_SHUTDOWN_REQUEST_MSG_STRUCT
```

EPWR_ONS_TYPE

Name	Value
epssdmonsINVALID	0xFF
epssdmonsNO_ONS	0x00
epssdmonsONS	0x01

EPWR_SHUTDOWN_TRIGGER_TYPE

Name	Value
epssdmsdtINVALID	0xFF
epssdmsdtTESTACC_OFF	0x00
epssdmsdtON_KEY	0x01
epssdmsdtIGN_LOCK	0x02
epssdmsdtPWR_SAVE	0x03
epssdmsdtTMP_STARTUP	0x04
epssdmsdtDIAG_DEACTIVATION	0x05
epssdmsdtABNORMAL_VOLTAGE	0x06
epssdmsdtABNORMAL_TEMP	0x07
epssdmsdtBATTERYCUTOFF	0x08
epssdmsdtLIMPHOME	0x09

epssdmsdtHU_CAN_ERROR	0x0A
epssdmsdtBODY_CAN_ERROR	0x0B
epssdmsdtTRANSPORT_MODE	0x0C
epssdmsdtPRODUCTION_MODE	0x0D
epssdmsdtIGN_OFF	0x0E
epssdmsdtGENERIC_ERROR_RESET	0x0F
epssdmsdtFATAL_ERROR_RESET	0x10
epssdmsdtUSER_DATA_RESET	0x11
epssdmsdtFACTORY_DATA_RESET	0x12
epssdmsdtFAST_SLEEP_MODE	0x13
epssdmsdtNORMAL_RESET	0x14
epssdmsdtPROGUPDATE_RESET	0x15

EPWR_USER_MODE_TYPE

Name	Value
epsumINVALID	0xFF
epsumOFF	0x00
epsumON	0x01

EPWR_TRANSPORT_MODE_TYPE

Name	Value
epstmINVALID	0xFF
epstmDISABLED	0x00
epstmENABLED	0x01

EPWR_PROD_MODE_TYPE

Name	Value
epspmINVALID	0xFF
epspmDISABLED	0x00
epspmENABLED	0x01

EPWR_LHC_TYPE

Name	Value
epslhcINVALID	0xFF
epslhcDISABLED	0x00
epslhcENABLED	0x01

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Method x Method

See also:

None

E_CWORD33_Status PwrServiceSendStartupConfirmationMsgRequest (HANDLE *f_hSession*, EPWR_SC_MSG_STRUCT *f_eState*)

Summary

Send startup confirmation request.

Parameters:

in	*f_hSession*	HANDLE - Session handle
in	*f_eState*	EPWR_SC_MSG_STRUCT 4 Address of the structure for startup confirmation request message

Structure for startup confirmation request message

```
1 typedef struct {
2    EPWR_SC__CWORD56__BOOT_MODE_TYPE _CWORD56_BootMode;
3    EPWR_SC_WAKEUP_TYPE        wakeupType;
4    EPWR_SC_COLD_START_REQ_TYPE coldStartRequest;
5    EPWR_SC_SECURITY_STATUS     securityStatus;
6    UI_32                      HWVersion;
7    UI_32                      matchedHardwareType;
8    UI_32                      softwareType;
9    UI_32                      imageType;
10    UI_32                      majorVersion;
11    UI_32                      minorVersion;
12    UI_32                      SWVersionYear;
13    UI_32                      SWVersionWeek;
14    UI_32                      SWVersionPatchLevel;
15 } StartupConfirmationMsgStrut, EPWR_SC_MSG_STRUCT;
```

EPWR_SC__CWORD56__BOOT_MODE_TYPE

Name	Value
eps_CWORD56_bmINVALID	0xFF
eps_CWORD56_bmAPPLICATION_MODE	0x00

eps_CWORD56_bmPROGRAMMING_MODE	0x08

EPWR_SC_WAKEUP_TYPE

Name	Value
epsstINVALID	0xFF
epsstWARMSTART	0x00
epsstCOLDSTART	0x01

EPWR_SC_COLD_START_REQ_TYPE

Name	Value
epsscrtINVALID	0xFF
epsscrtNOT_REQUIRED	0x00
epsscrtREQUIRED	0x01

EPWR_SC_SECURITY_STATUS

Name	Value
epsssINVALID	0xFF
epsssUNLOCK	0x00
epsssLOCK	0x01

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Fire and Forget Fire and Forget

See also:

None

E_CWORD33_Status PwrServiceSetCrankState (HANDLE *f_hSession*, EPWR_CRANK_STATE_TYPE *f_eCrank_state*)

Summary

Set crank state.

Parameters:

in	*f_hSession*	HANDLE - Session handle
in	*f_eCrank_state*	EPWR_CRANK_STATE_TYPE - Crank state value

Crank state value

Name	Value
epscsINVALID	0xFF
epscsENTRY	0xA0
epscsEXIT	0xA2

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Fire and Forget only

See also:

None

E_CWORD33_Status PwrServiceSetVoltageState (HANDLE *f_hSession*, EPWR_VOLTAGE_STATE_TYPE *f_eVoltage_state*)

Summary

Set power supply state.

Parameters:

in	*f_hSession*	HANDLE - Session handle
in	*f_eVoltage_state*	EPWR_VOLTAGE_STATE_TYPE - Power supply status value

Power supply status value

Name	Value

epsvsINVALID	0xFF
epsvsNORMAL	0x20
epsvsLVI1	0x22
epsvsLVI2	0x24

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Method only

See also:

None

E_CWORD33_Status PwrServiceSystemModeInfoRequest (HANDLE *f_hSession*)

Summary

Request to obtain the system mode information.

Parameters:

in	*f_hSession*	HANDLE - Session handle

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusFail	Failed

Classification

Public

Type

Method Method

See also:

None

Version_library

struct SSVER_PkgInfo

The structure of version package information. *More...*

class CSSVer

***Version information set-up/acquisition.* Macros**

#define **SSVER_VERCHAR_MAX** (64)
#define **SSVER_DATE_MAX** (16)

Typedefs

typedef std::map< std::string, SSVER_PkgInfo > **SSVerPkgList**
typedef SSVerPkgList::const_iterator **SSVerPkgListIter**

Functions

CSSVer::CSSVer ()
CSSVer::~CSSVer ()
SSVerPkgListIter CSSVer::begin ()
SSVerPkgListIter CSSVer::end ()
bool CSSVer::empty ()
E_CWORD78_Status CSSVer::getPkgInfo (const std::string &name, SSVER_PkgInfo *p_info)
E_CWORD78_Status CSSVer::setPkgInfo (const std::string &name, SSVER_PkgInfo &info)

Variables

char SSVER_PkgInfo::version [SSVER_VERCHAR_MAX]
char SSVER_PkgInfo::date [SSVER_DATE_MAX]

Detailed Description

Class Documentation

struct SSVER_PkgInfo

The structure of version package information.

Class Members:

char	date[SSVER_DATE_MAX]	Date (String):The termination of string must be "\0" Date (String):Store "\0" when date information does not exist.
char	version[SSVER_VERCHAR_MAX]	Version information (String): The termination of string must be "\0"

Function Documentation

SSVerPkgListIter CSSVer::begin ()[inline]

Summary

Returns the first iterator of package list.

Parameters:

None	

Return values:

SSVerPkgListIter	The first iterator of package list

The structure of version package information

```
#define SSVER_VERCHAR_MAX   (64)
#define SSVER_DATE_MAX      (16)
typedef struct{
   char version[SSVER_VERCHAR_MAX];    /* Version information (String): The termination of
*                                      string must be "\0" */
   char date[SSVER_DATE_MAX];          /* Date (String): The termination of string must be "\0"*/
                                       /* Store "\0" when date information does not exist. */
}SSVER_PkgInfo;

typedef std::map<std::string, SSVER_PkgInfo> SSVerPkgList;  /* Package list definition           */
typedef SSVerPkgList::const_iterator SSVerPkgListIter;      /* Iterator definition of package list */
```

Preconditions

None

Changes of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

end

CSSVer::CSSVer ()

Summary

Constructor for CSSVer class

Parameters:

None	

Return values:

None	

Preconditions

None

Changes of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

~CSSVer

bool CSSVer::empty ()[inline]

Summary

Checks whether data exists in package list or not.

Parameters:

None	

Return values:

bool	The possibility of data existence of package list

true : Data does not exist
false : Data exists

Preconditions

None

Changes of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

None

SSVerPkgListIter CSSVer::end ()[inline]

Summary

Returns the end iterator of package list.

Parameters:

None	

Return values:

SSVerPkgListIter	The end iterator of package list

The structure of version package information

```
#define SSVER_VERCHAR_MAX   (64)
#define SSVER_DATE_MAX      (16)
typedef struct{
   char version[SSVER_VERCHAR_MAX];    /* Version information (String): The termination of
*                                          string must be "\0" */
   char date[SSVER_DATE_MAX];          /* Date (String): The termination of string must be "\0" */
                                       /* Store "\0" when date information does not exist. */
}SSVER_PkgInfo;

typedef std::map<std::string, SSVER_PkgInfo> SSVerPkgList;  /* Package list definition           */
typedef SSVerPkgList::const_iterator SSVerPkgListIter;      /* Iterator definition of package list */
```

Preconditions

None

Changes of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

begin

E_CWORD78_Status CSSVer::getPkgInfo (const std::string & *name*, SSVER_PkgInfo * *p_info*)

Summary

Gets the package information of specified package.

Parameters:

in	*name*	std::string & - Package name
out	*p_info*	SSVER_PkgInfo* - Pointer to the storage area of

		package information

The structure of version package information

```
#define SSVER_VERCHAR_MAX   (64)
#define SSVER_DATE_MAX      (16)
typedef struct{
   char version[SSVER_VERCHAR_MAX];    /* Version information (String): The termination of
*                                        string must be "\0" */
   char date[SSVER_DATE_MAX];          /* Date (String): The termination of string must be "\0" */
                                       /* Store "\0" when date information does not exist. */
}SSVER_PkgInfo;

typedef std::map<std::string, SSVER_PkgInfo> SSVerPkgList;  /* Package list definition           */
typedef SSVerPkgList::const_iterator SSVerPkgListIter;      /* Iterator definition of package list */
```

Return values:

e_CWORD78_StatusOK	Succeeded in getting package information
e_CWORD78_StatusFileLoadError	Loading error for file
e_CWORD78_StatusFail	Failed at some process

Preconditions

None

Changes of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

setPkgInfo

E_CWORD78_Status CSSVer::setPkgInfo (const std::string & *name*, SSVER_PkgInfo & *info*)

Summary

Sets the package information of specified package.

Parameters:

in	*name*	std::string & - Package name
in	*info*	SSVER_PkgInfo* - Package information

The structure of version package information

```
#define SSVER_VERCHAR_MAX   (64)
#define SSVER_DATE_MAX      (16)
typedef struct{
   char version[SSVER_VERCHAR_MAX];    /* Version information (String): The termination of
```

```
*                                       string must be "\0" */
    char date[SSVER_DATE_MAX];          /* Date (String): The termination of string must be \ */
                                        /* Store "\0" when date information does not exist. */
}SSVER_PkgInfo;

typedef std::map<std::string, SSVER_PkgInfo> SSVerPkgList;  /* Package list definition          */
typedef SSVerPkgList::const_iterator SSVerPkgListIter;      /* Iterator definition of package list */
```

Return values:

e_CWORD78_StatusOK	Succeeded in setting package information
e_CWORD78_StatusFail	Faild at some process

Preconditions

None

Changes of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

getPkgInfo

CSSVer::~CSSVer ()

Summary

Destructor for CSSVer class

Parameters:

None	

Return values:

None	

Preconditions

None

Changes of the internal state

The internal state is not changed.

Classfication

Public

Type

None

See also:

CSSVer

Variable Documentation

char SSVER_PkgInfo::date[SSVER_DATE_MAX]

Date (String):The termination of string must be "\0" Date (String):Store "\0" when date information does not exist.

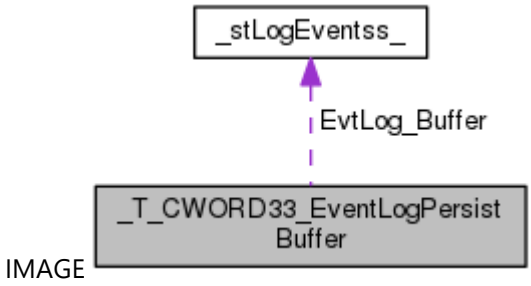
char SSVER_PkgInfo::version[SSVER_VERCHAR_MAX]

Version information (String): The termination of string must be "\0"

Class Documentation

_T_CWORD33_EventLogPersistBuffer Struct Reference

Collaboration diagram for _T_CWORD33_EventLogPersistBuffer:



IMAGE

Public Attributes

st_LogEvent_ss **EvtLog_Buffer** [EVT_BUFMAXSIZE]
UI_16 **Current_Log_Size**

Static Public Attributes

static const UI_16 **EVT_BUFMAXSIZE** = MAX_EVT_RECORDS

The documentation for this struct was generated from the following file:

5 ss_logger_service_local.h

_T_CWORD33_ScreenCaptureEvt Struct Reference

Screen Capture Event data.

#include <ss_logger_service.h>

Public Attributes

BOOL **fSucessful**
UI_32 **uiSceenShotId**
CHAR **strNameAndLocation** [STR_BUFF_LEN]
UI_32 **uiFaultReasonCode**

Static Public Attributes

static const UI_16 **STR_BUFF_LEN** = ***

Detailed Description

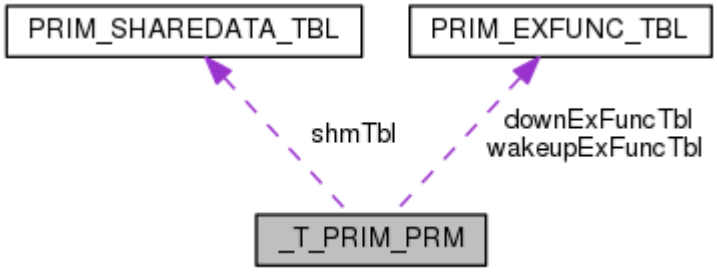
Screen Capture Event data.

The documentation for this struct was generated from the following file:

6 ss_logger_service.h

_T_PRIM_PRM Struct Reference

Collaboration diagram for _T_PRIM_PRM:



IMAGE

Public Attributes

PCSTR **name**
const PRIM_SHAREDATA_TBL * **shmTbl**
const PRIM_EXFUNC_TBL * **wakeupExFuncTbl**
const PRIM_EXFUNC_TBL * **downExFuncTbl**
E_CWORD33_Status(* **onInit**)(HANDLE hApp)
E_CWORD33_Status(* **onDestory**)(HANDLE hApp)
E_CWORD33_Status(* **onDebugDump**)(HANDLE hApp)
E_CWORD33_Status(* **onTouch**)(HANDLE hApp)
E_CWORD33_Status(* **onLowMem**)(HANDLE hApp)
void * **priv**

The documentation for this struct was generated from the following file:

7 INI_API.hpp

_TErrorLogger_FolderInfo Struct Reference

Public Attributes

CHAR **FoldernameAndLogname** [STR_BUFF_LEN]
CHAR **StorageTarget** [STR_BUFF_LEN]

Static Public Attributes

static const UI_16 **STR_BUFF_LEN** = ***

The documentation for this struct was generated from the following file:

8 ss_logger_service_local.h

_TSKM_EVENT_INFO_t Struct Reference

Public Attributes

TSKM_EVENT_t **event**
TSKM_ERR_t **errCode**
pid_t **fromPid**
TSKM_BOOL_t **hasExtend**
uint32_t **extendSize**
void * **extendPrm**
union {
TSKM_EV_PRI_REQ_WAKEUP_PRM_t **reqWakeup**
TSKM_EV_PRI_REQ_DOWN_PRM_t **reqDown**
TSKM_EV_PRI_REP_WAKEUP_COMP_PRM_t **repWakeupComp**
TSKM_EV_PRI_REP_DOWN_COMP_PRM_t **repDownComp**
TSKM_EV_PRI_REP_CONNECT_PRM_t **repConnect**
TSKM_EV_PRI_RES_WAKEUP_PRM_t **resWakeup**
TSKM_EV_PRI_RES_DOWN_PRM_t **resDown**
TSKM_EV_LCL_CHG_SVC_STATE_PRM_t **chgSvc**
} **prm**

The documentation for this struct was generated from the following file:

9 tskm_local_type.h

_TVINnumber Struct Reference

VIN numbers.

#include <ss_logger_service.h>

Public Attributes

CHAR **VINstr** [VIN_LEN]

Static Public Attributes

static const UI_8 **VIN_LEN** = ***

Detailed Description

VIN numbers.

The documentation for this struct was generated from the following file:

10 ss_logger_service.h

BOOT_AccessIf Class Reference

BOOT_AccessIf.

#include <ss_sm_boot_access.h>

Public Member Functions

BOOT_AccessIf (BAI_OPEN_t type)
~BOOT_AccessIf ()
E_CWORD33_Status getBootInfo (LBM_NOR_t *p_boot_info)
E_CWORD33_Status setBootInfo (LBM_NOR_t *p_boot_info)

Detailed Description

BOOT_AccessIf.

Brief Introduction

Class to provide the function of BOOT AccessIf

Constructor & Destructor Documentation

BOOT_AccessIf::BOOT_AccessIf (BAI_OPEN_t *type*)[explicit]

Summary:

Obtain the access permission to the boot information (secondary storage area).

Parameters:

in	*type*	BAI_OPEN_t - Access type to the boot information

BAI_OPEN_t
BAI_MTD_DEV_RO Access type for the Read only
BAI_MTD_DEV_RW Access type for the Read and Write

Return values:

None	

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public

Type

None

See also:

~BOOT_AccessIf, CL_LockMap, CL_LockGet

BOOT_AccessIf::~BOOT_AccessIf ()

~

Summary:

Release the access permission to the boot information (secondary storage area).

Parameters:

None	

Return values:

None	

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public

Type

None

See also:

BOOT_AccessIf, CL_LockRelease

Member Function Documentation

E_CWORD33_Status BOOT_AccessIf::getBootInfo (LBM_NOR_t * *p_boot_info*)

Summary:

Read the boot information (secondary storage area).

Parameters:

out	*p_boot_info*	LBM_NOR_t * - Destination buffer to read the boot information

LBM_NOR_t Structure body

typedef struct LBM_boot_t{
  LBM_UINT32 sts1;   //SYSUP_CND_*
  LBM_UINT32 btmode;   //SYSUP_MODE_*
  LBM_UINT32 sts2;   //SYSUP_CND_*
  LBM_UPTBLINFO upTbl[UPTBLE_ID_MAX];
  LBM_UINT32 sts3;   //SYSUP_CND_*
}LBM_NOR_t;

Refer to "ss_boot_map.h" for the parameters to be set to the member of struct LBM_NOR_t.

Return values:

e_CWORD33_StatusOK	Reading success
e_CWORD33_StatusFail	Reading failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public

Type

Sync only(None communication)

See also:

setBootInfo

E_CWORD33_Status BOOT_AccessIf::setBootInfo (LBM_NOR_t * *p_boot_info*)

Summary:

Write the boot information (secondary storage area).

Parameters:

in	*p_boot_info*	LBM_NOR_t * - Original buffer to write the boot information

LBM_NOR_tStructure body

typedef struct LBM_boot_t{
  LBM_UINT32 sts1;   //SYSUP_CND_*
  LBM_UINT32 btmode;   //SYSUP_MODE_*
  LBM_UINT32 sts2;   //SYSUP_CND_*
  LBM_UPTBLINFO upTbl[UPTBLE_ID_MAX];
  LBM_UINT32 sts3;   //SYSUP_CND_*
}LBM_NOR_t;

Refer to "ss_boot_map.h" for the parameters to be set to the member of struct LBM_NOR_t.

Return values:

e_CWORD33_StatusOK	Write success

e_CWORD33_StatusFail	Write failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public

Type

Sync only(None communication)

See also:

getBootInfo

The documentation for this class was generated from the following file:

11 ss_sm_boot_access.h

CHeartBeatServiceIf Class Reference

Public Member Functions

CHeartBeatServiceIf (BOOL avail, HANDLE service, PCSTR name)
virtual ~CHeartBeatServiceIf ()
E_CWORD33_Status OnHeartBeatRequest (HANDLE hApp)

Friends

E_CWORD33_Status **DBG_CloseHeatBeat** (HANDLE hApp)

Constructor & Destructor Documentation

CHeartBeatServiceIf::CHeartBeatServiceIf (BOOL *avail*, HANDLE *service*, PCSTR *name*)

Summary

Constructor of CHeartBeatServiceIf class.

Parameters:

in	*avail*	BOOL avail - default value of HBSession available
in	*service*	HANDLE service - set default value of Service handle
in	*name*	PSTR name - set application name used by client

Return values:

None	

Preconditions

None.

Change of the internal state

Change of internal state according to the API does not occur.

Classification

Public

Type

None

See also:

~CHeartBeatServiceIf

virtual CHeartBeatServiceIf::~CHeartBeatServiceIf ()[virtual]

Summary

Destructor of CHeartBeatServiceIf class.

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

Change of internal state according to the API does not occur.

Classification

Public

Type

None

See also:

CHeartBeatServiceIf

Member Function Documentation

E_CWORD33_Status CHeartBeatServiceIf::OnHeartBeatRequest (HANDLE *hApp*)

Summary

Subscribe hApp to message queue of the HeartBeat client application.

Parameters:

in	*f_hApp*	HANDLE - Handle to message queue of the HeartBeat client application.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Preconditions

Change of the internal state

Change of internal state according to the API does not occur.

Classification

Public

Type

None

See also:

_CWORD33_GetAppName _CWORD33_SubscribeNotificationsWithCallback

---

The documentation for this class was generated from the following file:

12 ss_heartbeat_if.h

CSSVer Class Reference

Version information set-up/acquisition.

#include <ss_ver.h>

Public Member Functions

CSSVer ()
~CSSVer ()
SSVerPkgListIter begin ()
SSVerPkgListIter end ()
bool empty ()
E_CWORD78_Status getPkgInfo (const std::string &name, SSVER_PkgInfo *p_info)
E_CWORD78_Status setPkgInfo (const std::string &name, SSVER_PkgInfo &info)

Detailed Description

Version information set-up/acquisition.

Brief Introduction

Class to provide the function of the version information set-up/acquisition

---

The documentation for this class was generated from the following file:
13 ss_ver.h

EnumStringMap< enumT, fp > Class Template Reference

EnumStringMap.

#include <ss_templates.h>

Public Member Functions

EnumStringMap ()
~EnumStringMap ()
SS_String **GetStr** (enumT f_enum)

Detailed Description

template<typename enumT, void(*)(std::map< enumT, SS_String > &m_strMap) fp>

class EnumStringMap< enumT, fp >

EnumStringMap.

Brief Introduction

Class to provide EnumStringMap template function

The documentation for this class was generated from the following file:

14 ss_templates.h

LoggerServiceIf Class Reference

logger_service

#include <ss_logger_service_ifc.h>

Public Member Functions

LoggerServiceIf ()
virtual ~LoggerServiceIf ()
BOOL Initialize (HANDLE hApp)
E_CWORD33_Status NotifyOnLoggerServiceAvailability (CbFuncPtr f_pCallBackFn)
E_CWORD33_Status NotifyOnOpenSessionAck (CbFuncPtr f_pCallBackFn)
E_CWORD33_Status NotifyOnCloseSessionAck (CbFuncPtr f_pCallBackFn)
E_CWORD33_Status OpenSessionRequest ()
E_CWORD33_Status DecodeOpenSessionResponse ()
E_CWORD33_Status CloseSessionRequest ()
E_CWORD33_Status RegisterForLoggerEvent (SS_LoggerServerEvents f_eLoggerEvent, CbFuncPtr f_pCallBackFn)
E_CWORD33_Status UnRegisterForLoggerEvent (SS_LoggerServerEvents f_eLoggerEvent)
E_CWORD33_Status SendDiagStat (STLOGGER_CANDIAGSTAT *pCanDiagStat_t)
E_CWORD33_Status LoggerIf_CANGetCurrentDateAndTime (STCanCurrentDateTime stDateAndTime)
E_CWORD33_Status LoggerIf_CANSetVIN (STVIN_NUMBER &stVinNumber)
E_CWORD33_Status LoggerIf_SetLoggerParams (ELOGGER_STAT eLoggerStatus, EDEV_TYPE eDevType)
E_CWORD33_Status LoggerIf_UDPLogging (ELOGGER_STAT eLoggerStatus)
E_CWORD33_Status LoggerIf_ScreenCaptureEventACK (STScreenCaptureEvt stScreenCaptureInfo)
E_CWORD33_Status LoggerIf_EventLoggingCommonInfo (STEventLoggerCommonInfo stEventCommonInfo)
E_CWORD33_Status LoggerIf_EventLoggingEventInfo (STEventLoggerEventInfo stEventInfo)
E_CWORD33_Status LoggerIf_EventLoggingResetInfo (STEventLoggerResetInfo stResetInfo)
E_CWORD33_Status LoggerIf_PersistEventLogOnActiveDTC (void)
E_CWORD33_Status LoggerIf__CWORD105_Shutdown_Complete (void)
E_CWORD33_Status StoreEventLogsToUSB (EEvtLogUSBDevNumber eUSBDevNumber)
E_CWORD33_Status StoreEmergencyErrorLogsToUSB (EDevNumber eDevNumber)
E_CWORD33_Status StartCANLogging (void)
E_CWORD33_Status RegisterLoggingStartNotification (CbFuncPtr f_pCallBackFn)
E_CWORD33_Status RegisterLoggingFinishNotification (CbFuncPtr f_pCallBackFn)
E_CWORD33_Status StartDTCLoggingToEmmc (UI_32 f_dtc)
E_CWORD33_Status ClearEventLogs (void)
E_CWORD33_Status ReadStatisticalCounter (EStatCounterGroupID eCounterGroup)
E_CWORD33_Status ResetStatisticalCounter (EStatCounterGroupID eCounterGroup)
E_CWORD33_Status LoggerIf_SetDiagID (UI_16 f_u16DiagID)
E_CWORD33_Status LoggerIf_UploadEventLog ()

Detailed Description

logger_service

Brief Introduction

Class to provide LoggerServiceIf

Constructor & Destructor Documentation

LoggerServiceIf::LoggerServiceIf ()

Summary

Constructor of LoggerServiceIf class.

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

~LoggerServiceIf, Initialize

virtual LoggerServiceIf::~LoggerServiceIf ()[virtual]

~

Summary

Destructor of LoggerServiceIf class.

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

LoggerServiceIf

Member Function Documentation

E_CWORD33_Status LoggerServiceIf::ClearEventLogs (void)

Summary

API to request to clear the event log to SS_LoggerService.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Pub-Sub

See also:

None

E_CWORD33_Status LoggerServiceIf::CloseSessionRequest ()

Summary

API to discard service/session of SS_LoggerService.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop etc.) has been done, and Availability of SS_LoggerService is TRUE.
Use the NotifyOnCloseSessionAck(), that you register a callback function for receiving a completion response of CloseSessionRequest to Dispatcher

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Pub-Sub Pub-Sub

See also:

OpenSessionRequest

E_CWORD33_Status LoggerServiceIf::DecodeOpenSessionResponse ()

Summary

API to hold the session ID in SS_LoggerServiceIf class

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusFail	Some sort of error occurred

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop etc.) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

OpenSessionRequest, CloseSessionRequest

BOOL LoggerServiceIf::Initialize (HANDLE *hApp*)

Summary

API to initialize the LoggerServiceIf class.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application

Return values:

TRUE	Success
FALSE	Failure

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

LoggerServiceIf

E_CWORD33_Status LoggerServiceIf::LoggerIf__CWORD105_Shutdown_Complete (void)

Summary

API to send a shutdown completion message of the *CWORD105* to SS_LoggerService.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_CANGetCurrentDateAndTime (STCanCurrentDateTime *stDateAndTime*)

Summary

API to send the current date and time of the CAN to SS_LoggerService.

Parameters:

in	*stDateAndTime*	STCanCurrentDateTime - Structure of CAN current date and time

STCanCurrentDateTime Structure

```
typedef struct __CWORD62_DateAndTime
{
    UI_8  DateTime_Stat;    /* Date and Time State */
    UI_8  DateTimeDay;      /* Day                 */
    UI_8  DateTimeHour;     /* Hour                */
    UI_8  DateTimeMinute;   /* Minute              */
    UI_8  DateTimeMonth;    /* Month               */
    UI_8  DateTimeSecond;   /* Second              */
    UI_8  DateTimeYear;     /* Year                */
    UI_8  TimeFormat;       /* Time display format */
}STCanCurrentDateTime;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Sync

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_CANSetVIN (STVIN_NUMBER & *stVinNumber*)

Summary

API to send the Vehicle Identification Number(VIN) to SS_LoggerService.

Parameters:

in	*stVinNumber*	STVIN_NUMBER& - Structure of Vehicle Identification Number(VIN)

STVIN_NUMBER Structure

```
typedef struct _TVINnumber
{
    static const UI_8 VIN_LEN = ***;
    CHAR   VINstr[VIN_LEN];       /* Vehicle Identification Number(VIN)  */
}STVIN_NUMBER;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Sync

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_EventLoggingCommonInfo (STEventLoggerCommonInfo *stEventCommonInfo*)

Summary

API to send the event information of the common log in SS_LoggerService.

Parameters:

in	*stEventCommonInfo*	STEventLoggerCommonInfo - Event information structure of the common log

STEventLoggerCommonInfo Structure

```
typedef struct _SEventLoggerCommonInfo
{
    UI_8  BodyCAN_Stat:4;       /* CAN state of the body          */
    UI_8  HeadUnitCAN_Stat:4;   /* CAN state of the Head Unit     */
    UI_8  HMIInteraction:4;     /* HMI Interaction                */
    UI_8  IGN_Status:4;         /* Ignition state                 */
    UI_8  FOT_Temp:2;           /* Field Operation Tests Temporary */
    UI_8  Syst_CWORD61_oltage:6;      /* System Voltage               */
}STEventLoggerCommonInfo;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_EventLoggingEventInfo (STEventLoggerEventInfo *stEventInfo*)

Summary

API to send the event information of the *CWORD56* log to SS_LoggerService.

Parameters:

in	*stEventInfo*	STEventLoggerEventInfo - Event information structure of *CWORD56* log

STEventLoggerEventInfo Structure

```
typedef struct _SEventLoggerCommonInfo
{
    UI_8 NumberOfEvents;      /* Event count      */
    UI_8 EventGroup;          /* Event group      */
    UI_8 EventIdentifier;     /* Event identifier */
    UI_8 EventData[4];        /* Event data       */
}STEventLoggerEventInfo;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_EventLoggingResetInfo (STEventLoggerResetInfo *stResetInfo*)

Summary

API to send the event information of *CWORD56* reset count to SS_LoggerService.

Parameters:

in	*stResetInfo*	STEventLoggerResetInfo - Event information structure of *CWORD56* reset counter

STEventLoggerResetInfo Structure

```
typedef struct _SEventLoggerResetInfo
{
    UI_8 _CWORD56__ResetInfo;     /* Reset counter information of _CWORD56_   */
    UI_8 _CWORD102__ResetInfo;      /* Reset counter information of _CWORD102_    */
}STEventLoggerResetInfo;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_PersistEventLogOnActiveDTC (void)

Summary

API to persist the event log on the DTC to SS_LoggerService.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Sync

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_ScreenCaptureEventACK (STScreenCaptureEvt *stScreenCaptureInfo*)

Summary

API to send the event response of screen capture to SS_LoggerService.

Parameters:

in	*stScreenCaptureInfo*	STScreenCaptureEvt - Structure of event data of screen capture

STScreenCaptureEvt Structure

```
typedef struct _T_CWORD33_ScreenCaptureEvt
{
    static const UI_16  STR_BUFF_LEN = ***;
    BOOL            fSucessful;                 /* Success propriety flag       */
    UI_32           uiSceenShotId;              /* Screen capture ID            */
    CHAR            strNameAndLocation[STR_BUFF_LEN]; /* Screen capture save file name */
    UI_32           uiFaultReasonCode;          /* Failure reason code          */
}STScreenCaptureEvt;
```

uiFaultReasonCode : Failure reason code (plm::screen_shot::status List)
ok : Success
invalid_window_handle : Invalid window handle
invalid_context_handle : Invalid context handle
display_count_error : Display count error
rgb_display_not_found : RGB display not found
create_pixmap_error : Create error of PIXMAP
usage_pixmap_property_error : Usage error of PIXMAP property
format_pixmap_property_error : Format error of PIXMAP property
buffer_pixmap_property_error : Buffer error of PIXMAP property
create_pixmap_buffer_error : Create error of PIXMAP buffer
render_pixmap_property_error : Render error of PIXMAP property
pointer_pixmap_property_error : Pointer error of PIXMAP property
stride_pixmap_property_error : Stride error of PIXMAP property
window_read_error : Window read error
write_bitmap_file_error : Writing error of bitmap files
window_post_error : Window post error
window_property_error : Window property error

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size

e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Sync

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_SetDiagID (UI_16 *f_u16DiagID*)

Summary

API to send the diagnosis ID to SS_LoggerService.

Parameters:

in	*f_u16DiagID*	UI_16 - Diagnosis ID

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Sync

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_SetLoggerParams (ELOGGER_STAT *eLoggerStatus*, EDEV_TYPE *eDevType*)

Summary

API to send the logging propriety information of the selected device to SS_LoggerService.

Parameters:

in	*eLoggerStatus*	ELOGGER_STAT - Logging propriety information
in	*eDevType*	EDEV_TYPE - Device type

ELOGGER_STAT enum value

```
typedef enum _ELoggerState
{
    eDeactivate,         /* Logging impossible(Deativate) */
    eActivate,           /* Logging possible(Activate)    */
    eInvalid_LoggerState  /* Invalid logging state         */
}ELOGGER_STAT;
```

EDEV_TYPE enum value

```
typedef enum _ELoggerDeviceTypes
{
    eDevUSB1,           /* USB1                */
    eDevUSB2,           /* USB2                */
    eDevSD,             /* SD                  */
    eEthernet,          /* Ethernet            */
    eTotalDevicesTypes,   /* Total device types   */
    eInvalid_DevType      /* Invalid device type  */
}EDEV_TYPE;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred

e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Sync

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_UDPLogging (ELOGGER_STAT *eLoggerStatus*)

Summary

API to send the logging propriety information of UDP to SS_LoggerService.

Parameters:

in	*eLoggerStatus*	ELOGGER_STAT - Logging propriety information

ELOGGER_STAT enum value

```
typedef enum _ELoggerState
{
    eDeactivate,         /* Logging impossible(Deativate) */
    eActivate,           /* Logging possible(Activate)    */
    eInvalid_LoggerState  /* Invalid logging state         */
}ELOGGER_STAT;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Sync

See also:

None

E_CWORD33_Status LoggerServiceIf::LoggerIf_UploadEventLog ()

Summary

API to request the upload of the event log to SS_LoggerService.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method x Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::NotifyOnCloseSessionAck (CbFuncPtr *f_pCallBackFn*)

Summary

API to set the Callback function for CloseSessionRequest response from SS_LoggerService.

Parameters:

in	*f_pCallBackFn*	CbFuncPtr - Callback function pointer for receiving the response of OpenSessionRequest

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle

Preconditions

Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

NotifyOnOpenSessionAck

E_CWORD33_Status LoggerServiceIf::NotifyOnLoggerServiceAvailability (CbFuncPtr *f_pCallBackFn*)

Summary

API to set the Callback function for Availability notification of SS_LoggerService.

Parameters:

in	*f_pCallBackFn*	CbFuncPtr - Pointer to function callback for SS_LoggerService Availability Notification

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Preconditions

Generation/initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop etc.) has been done.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Pub-Sub Pub-Sub

See also:

None

E_CWORD33_Status LoggerServiceIf::NotifyOnOpenSessionAck (CbFuncPtr *f_pCallBackFn*)

Summary

API to set the Callback function for OpenSessionRequest response from SS_LoggerService.

Parameters:

in	*f_pCallBackFn*	CbFuncPtr - Callback function pointer for receiving the response of OpenSessionRequest

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle

Preconditions

Generation/initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop etc.) has been done.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

NotifyOnCloseSessionAck

E_CWORD33_Status LoggerServiceIf::OpenSessionRequest ()

Summary

API to generate service/session of SS_LoggerService.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Preconditions

SS_LoggerService process has been started.
Availability of SS_LoggerService is TRUE.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.
Use the NotifyOnOpenSessionAck(), that you register a callback function for receiving a completion response of OpenSessionRequest to Dispatcher

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method x Fire and Forget

See also:

CloseSessionRequest

E_CWORD33_Status LoggerServiceIf::ReadStatisticalCounter (EStatCounterGroupID *eCounterGroup*)

Summary

API to request a statistics counter reading of the event log to SS_LoggerService.

Parameters:

in	*eCounterGroup*	EStatCounterGroupID - Group ID of the statistics counter

EStatCounterGroupID enum value

```
typedef enum  ECounterGroupID
{
    STARTUP_SHUTDOWN_COUNTER_GROUP =0x01, /* Statistics counter group for StartUp/Shutdown    */
    _CWORD105__EVENTS_COUNTER_GROUP,          /* Statistics counter group for _CWORD105_ event        */
    APP_USAGE_COUNTER_GROUP,            /* Statistics counter group for application usage   */
    F_BLK_STABILITY_COUNTER_GROUP,      /* Do not use : Statistics counter for flash access */
    MAX_COUNTER_GROUP                   /* Max value of statistics counter group           */
}EStatCounterGroupID ;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method x Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::RegisterForLoggerEvent (SS_LoggerServerEvents *f_eLoggerEvent*, CbFuncPtr *f_pCallBackFn*)

Summary

API to register a Callback function for SS_LoggerService.

Parameters:

in	*f_eLoggerEvent*	SS_LoggerServerEvents - Event type
in	*f_pCallBackFn*	CbFuncPtr - Pointer to a callback function that corresponds to the event

SS_LoggerServerEvents enum value

```
typedef enum _LoggerServerEvents
{
    SS_LOGGER_SCREENCAPTURE_EVT = ***, /* Event notification of the Screen Capture    */
    SS_LOGGER_ERRORINFO_EVT,           /* Event notification of error log information */
    SS_LOGGER_LOGINFO_EVT,             /* Event notification of log information       */
    SS_LOGGER_LOGSTARTED_EVT           /* Event notification at the time of log start */
}SS_LoggerServerEvents;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop etc.) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

UnRegisterForLoggerEvent

E_CWORD33_Status LoggerServiceIf::RegisterLoggingFinishNotification (CbFuncPtr *f_pCallBackFn*)

Summary

API to set a callback function for CAN logging finish notification.

Parameters:

in	*f_pCallBackFn*	CbFuncPtr - Callback function pointer for CAN logging finish notification

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop etc.) has been done.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Pub-Sub Pub-Sub

See also:

StartCANLogging, RegisterLoggingStartNotification

E_CWORD33_Status LoggerServiceIf::RegisterLoggingStartNotification (CbFuncPtr *f_pCallBackFn*)

Summary

API to set a callback function for CAN logging start notification.

Parameters:

in	*f_pCallBackFn*	CbFuncPtr - Callback function pointer for CAN logging start notification

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop etc.) has been done.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Pub-Sub Pub-Sub

See also:

StartCANLogging, RegisterLoggingFinishNotification

E_CWORD33_Status LoggerServiceIf::ResetStatisticalCounter (EStatCounterGroupID *eCounterGroup*)

Summary

API to request the reset of statistics counters of the event log to SS_LoggerService.

Parameters:

in	*eCounterGroup*	EStatCounterGroupID - Group ID of the statistics counter

EStatCounterGroupID enum value

```
typedef enum  ECounterGroupID
{
    STARTUP_SHUTDOWN_COUNTER_GROUP =0x01, /* Statistics counter group for StartUp/Shutdown    */
    _CWORD105__EVENTS_COUNTER_GROUP,          /* Statistics counter group for _CWORD105_ event         */
    APP_USAGE_COUNTER_GROUP,            /* Statistics counter group for application usage   */
    F_BLK_STABILITY_COUNTER_GROUP,      /* Do not use : Statistics counter for flash access */
    MAX_COUNTER_GROUP                   /* Max value of statistics counter group           */
}EStatCounterGroupID ;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::SendDiagStat (STLOGGER_CANDIAGSTAT * *pCanDiagStat_t*)

Summary

API to send CAN Diagnostic status data(it has mileage information) to SS_LoggerService.

Parameters:

in	*pCanDiagStat_t*	STLOGGER_CANDIAGSTAT - Pointer to structure of CAN Diagnostic state data

STLOGGER_CANDIAGSTAT Structure

```
typedef struct _T_CWORD33_CANMileageInfo
{
    UI_8  DidA_ExtTest_Pres;    /* For extended test(unused)     */
    UI_8  EngRun_Stat;          /* Engine run state(unused)      */
    UI_8  Odo__CWORD34__H;          /* Mileage information(_CWORD34_:High) */
    UI_8  Odo__CWORD34__L;          /* Mileage information(_CWORD34_:Low)  */
    UI_8  Odo_LSB_H;            /* Mileage information(LSB:High) */
    UI_8  Odo_LSB_L;            /* Mileage information(LSB:Low)  */
    UI_8  PN14_SupBat_Volt;     /* Battery voltage(unused)       */
}STLOGGER_CANDIAGSTAT;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget x Sync

See also:

None

E_CWORD33_Status LoggerServiceIf::StartCANLogging (void)

Summary

API to request the start of CAN logging to SS_LoggerService.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget Fire and Forget

See also:

RegisterLoggingStartNotification, RegisterLoggingFinishNotification

E_CWORD33_Status LoggerServiceIf::StartDTCLoggingToEmmc (UI_32 *f_dtc*)

Summary

API to request the start of CAN logging to SS_LoggerService.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success

e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::StoreEmergencyErrorLogsToUSB (EDevNumber *eDevNumber*)

Summary

API to request the emergency error log output to USB/SD to SS_LoggerService.

Parameters:

in	*eDevNumber*	EDevNumber - Emergency error log output destination USB/SD device number

EDevNumber enum

```
typedef enum  _EDevNumber
{
    eEEL_USB1=1,        /* USB0 */
    eEEL_USB2=2,        /* USB1 */
    eEEL_SD=3           /* SD   */
}EDevNumber ;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle

e_CWORD33_StatusNullPointer	Invalid pointer
e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Fire and Forget Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::StoreEventLogsToUSB (EEvtLogUSBDevNumber *eUSBDevNumber*)

Summary

API to request an event log output to USB/SD to SS_LoggerService.

Parameters:

in	*eUSBDevNumber*	EEvtLogUSBDevNumber - Event Log output destination USB/SD device number

EEvtLogUSBDevNumber enum value

```
typedef enum  _EEventLoggerUSBDeviceNumber
{
    USB0=0,       /* USB0 */
    USB1=1,       /* USB1 */
    SD=2          /* SD   */
}EEvtLogUSBDevNumber;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	Invalid pointer

e_CWORD33_StatusMsgQFull	Message queue is full
e_CWORD33_StatusErrNoEBADF	Invalid File-Descriptor
e_CWORD33_StatusErrNoEINTR	An interrupt is generated by the system call (signal)
e_CWORD33_StatusInvldBufSize	Invalid buffer-size
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error(It failed to open/allocation of shared memory)

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method x Fire and Forget

See also:

None

E_CWORD33_Status LoggerServiceIf::UnRegisterForLoggerEvent (SS_LoggerServerEvents *f_eLoggerEvent*)

Summary

API to cancel a Callback function for SS_LoggerService.

Parameters:

in	*f_eLoggerEvent*	SS_LoggerServerEvents - Event type

SS_LoggerServerEvents enum value

```
typedef enum  LoggerServerEvents
{
    SS_LOGGER_SCREENCAPTURE_EVT = ***, /* Event notification of the Screen Capture    */
    SS_LOGGER_ERRORINFO_EVT,           /* Event notification of error log information */
    SS_LOGGER_LOGINFO_EVT,            /* Event notification of log information       */
    SS_LOGGER_LOGSTARTED_EVT           /* Event notification at the time of log start */
}SS_LoggerServerEvents;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle

Preconditions

SS_LoggerService process has been started.
Generation/initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop etc.) has been done, and Availability of SS_LoggerService is TRUE.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

RegisterForLoggerEvent

The documentation for this class was generated from the following file:

15 ss_logger_service_ifc.h

PRIM_BACKUPCHK_TBL Struct Reference

Public Attributes

int32_t(* **func**)(int32_t kind)

The documentation for this struct was generated from the following file:

16 Primary_common.h

PRIM_EXFUNC_TBL Struct Reference

Public Attributes

void(* **func**)(void *prm)
void * **prm**
uint8_t **localStep**
uint8_t **rsv** [3]

The documentation for this struct was generated from the following file:
17 Primary_common.h

PRIM_FORK_TBL Struct Reference

Public Attributes

PRI_DWORD(* **threadFunc**)(void *prm)
void * **prm**
uint32_t **priority**
uint16_t **pno**
uint8_t **rsv**
uint8_t **wakeLocalStep**
char **threadName** [16]
char **downLocalStep**
void * **primaryLocal**

The documentation for this struct was generated from the following file:

18 Primary_common.h

Process Class Reference

Public Types

enum **eProcessLoadMode** { **WAIT**, **NOWAIT** }
enum **eProcessSchedulingPolicy** { **FIFO**, **ROUND_ROBIN**, **OTHER** }

Public Member Functions

Process ()
Process (const Process &p_rhs_i)
virtual ~Process ()
Process & operator= (const Process &p_rhs_i)
int const GetProcessId (void)
void SetCallingArgumentList (const char *p_pcArgv_i[], const int p_iArgc_i)
void SetCallingArgumentList (const StringList &p_strlstParameters_i)
const char *const GetExecutableFileName ()
int GetProcessReturnCode (void)
void SetSchedulingPolicy (const eProcessSchedulingPolicy p_eSchedulingPolicy_i)
eProcessSchedulingPolicy const GetSchedulingPolicy (void)
void SetPriority (const int p_iPriority_i)
void SetPriority (const pid_t p_iPid_i, const int p_iPriority_i)
void IncreasePriorityByOne (void)
void DecreasePriorityByOne (void)
int const GetPriority (void)
void SetProcessName (const SS_String &p_strProcessName_i)
SS_String GetProcessName (void) const
void CreateProcess (const SS_String &p_strFile_i, const StringList &p_strlstArgv_i, const eProcessLoadMode p_eMode_i, const SS_String &p_strProcessName_i, const eProcessSchedulingPolicy p_eSchedulingPolicy_i, const int p_iPriority_i, const char *unix_user_name, const long p_lSpawnFlags_i)
void CreateProcess (const SS_String &p_strFile_i, const SS_String &p_strProcessName_i, const char *unix_user_name=NULL, const long p_lSpawnFlags_i=iProcess_DEFAULT_PROCESS_FLAGS)
void CreateProcess (const SS_String &p_strFile_i, const SS_String &p_strProcessName_i, const int p_iPriority_i, const char *unix_user_name=NULL, const long p_lSpawnFlags_i=iProcess_DEFAULT_PROCESS_FLAGS)
void CreateProcess (const SS_String &p_strFile_i, const char *unix_user_name=NULL, const long p_lSpawnFlags_i=iProcess_DEFAULT_PROCESS_FLAGS)
void CreateProcess (const SS_String &p_strFile_i, const SS_String &p_strProcessName_i, const StringList &p_strlstArgv_i, const char *unix_user_name=NULL, const long p_lSpawnFlags_i=iProcess_DEFAULT_PROCESS_FLAGS)
void CreateProcess (const SS_String &p_strFile_i, const SS_String &p_strProcessName_i, const int p_iPriority_i, const StringList &p_strlstArgv_i, const char *unix_user_name=NULL, const long p_lSpawnFlags_i=iProcess_DEFAULT_PROCESS_FLAGS)
void CreateProcess (const SS_String *p_str_file, char *const *c_argv, char *environment_string, const CL_ProcessAttr_t *cl_attr)
void CreateProcessWait (const SS_String &p_strFile_i)
void CreateProcessWait (const SS_String &p_strFile_i, const StringList &p_strlstArguments_i)

void KillProcess (int signal=SIGKILL)
BOOL DoesProcessExist (void)
void DisableAutoKill (void)
void EnableAutoKill (void)

Constructor & Destructor Documentation

Process::Process ()

Summary

Default Constructor, called when the class is instantiated.

Parameters:

None	

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

initialize all var

Classification

Public

Type

sync only

See also:

~Process

Process::Process (const Process & *p_rhs_i*)

Summary

Copy Constructor

Parameters:

in	-	p_rhs_i 19 Process - Reference to a Process that you want to copy.

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

~Process

virtual Process::~Process ()[virtual]

~

Summary

Destructor function

Parameters:

None	

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

set m_lValidationTag with 0

Classification

Public

Type

sync only

See also:

Process

Member Function Documentation

void Process::CreateProcess (const SS_String & *p_strFile_i*, const StringList & *p_strlstArgv_i*, const eProcessLoadMode *p_eMode_i*, const SS_String & *p_strProcessName_i*, const eProcessSchedulingPolicy *p_eSchedulingPolicy_i*, const int *p_iPriority_i*, const char * *unix_user_name*, const long *p_lSpawnFlags_i*)

Summary

This method will create a process with the executable provided and mode as a calling parameter.The caller can also provide a list of arguments that will be provided to the executable at startup.

Parameters:

in	*p_strFile_i*	20 SS_String - Path and Filename of executable to create process for
in	*p_strlstArgv_i*	21 StringList - List of ARGV values for new process
in	*p_eMode_i*	22 eProcessLoadMode - Mode to create and load new process
in	*p_strProcessName_i*	23 SS_String - This is the name that will be registered to the OS for this process
in	*p_eSchedulingPolicy_i*	24 eProcessSchedulingPolicy
in	*p_iPriority_i*	25 int - Priority for this process
in	*unix_user_name*	26 char* - unix user name
in	*p_lSpawnFlags_i*	27 long - Spawning flags. These are PosixBasedOS001 specific....

eProcessLoadMode enum
- enum eProcessLoadMode {
  WAIT, //The invoked program is loaded into available memory, is executed,

```
            //and then the original program resumes execution.
  NOWAIT,     // Causes the current program to execute concurrently with the new child process.
 };
```

eProcessSchedulingPolicy enum

```
- enum eProcessSchedulingPolicy {
  FIFO,        // A fixed priority scheduler in which the highest ready process runs until it
               // blocks or is preempted by a higher priority process.
  ROUND_ROBIN,  // The same as FIFO, except processes at the same priority level time-slice.
  OTHER        // A general time sharing scheduler in which a process decays in priority if it
               // consumes too much processor before blocking. It reverts to its default priority
               // when it blocks. Should it fail to run over a 2 second period and it has decayed
               // then it's boosted one priority level up to a maximum of its default priority.
};
```

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

set m_iErrorCode with 0 set m_iReturnCode with 0 set m_eProcessLoadMode with p_eMode_i; set m_strFile with p_strFile_i; set m_strProcessName with p_strProcessName_i; use the result of function CL_ProcessCreate() to set m_tProcessId

Classification

Public

Type

sync only

See also:

None

void Process::CreateProcess (const SS_String & *p_strFile_i*, const SS_String & *p_strProcessName_i*, const char * *unix_user_name* = NULL, const long *p_lSpawnFlags_i* = iProcess_DEFAULT_PROCESS_FLAGS)

Summary

This method will create a PosixBasedOS001 process with the executable provided and mode as a calling parameter.

Parameters:

in	*p_strFile_i*	28 SS_String - Path and Filename of executable

		to create process for
in	*p_strProcessName_i*	29 SS_String - This is the name that will be registered to the OS for this process
in	*unix_user_name*	30 char* - unix user name
in	*p_lSpawnFlags_i*	31 long - Posix Spawning flags. These are PosixBasedOS001 specific

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::CreateProcess (const SS_String & *p_strFile_i*, const SS_String & *p_strProcessName_i*, const int *p_iPriority_i*, const char * *unix_user_name* = NULL, const long *p_lSpawnFlags_i* = iProcess_DEFAULT_PROCESS_FLAGS)

Summary

This method will create a process with the executable provided and mode as a calling parameter.

Parameters:

in	*p_strFile_i*	32 SS_String - Path and Filename of executable to create process for
in	*p_strProcessName_i*	

		33 SS_String - This is the name that will be registered to the OS for this process
in	*p_iPriority_i*	34 int - Priority of process
in	*unix_user_name*	35 char* - unix user name
in	*p_lSpawnFlags_i*	36 long - Posix Spawning flags. These are PosixBasedOS001 specific

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::CreateProcess (const SS_String & *p_strFile_i*, const char * *unix_user_name* = NULL, const long *p_lSpawnFlags_i* = iProcess_DEFAULT_PROCESS_FLAGS)

Summary

This method will create a process with the executable provided and mode as a calling parameter.

Parameters:

in	*p_strFile_i*	37 SS_String - p_strFile_i Path and Filename of executable to create process for
in	*unix_user_name*	38 char* - unix user name

in	*p_lSpawnFlags_i*	39 long - Spawning flags. These are PosixBasedOS001 specific.

Return values:

None	

Preconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::CreateProcess (const SS_String & *p_strFile_i*, const SS_String & *p_strProcessName_i*, const StringList & *p_strlstArgv_i*, const char * *unix_user_name* = NULL, const long *p_lSpawnFlags_i* = iProcess_DEFAULT_PROCESS_FLAGS)

Summary

This method will create a process with the executable provided and mode as a calling parameter.

Parameters:

in	*p_strFile_i*	40 SS_String - p_strFile_i Path and Filename of executable to create process for
in	*p_strProcessName_i*	41 SS_String - This is the name that will be registered to the OS for this process
in	*p_strlstArgv_i*	42 StringList - List of ARGV values for new process

in	*unix_user_name*	43 char* - unix user name
in	*p_lSpawnFlags_i*	44 long - Spawning flags. These are PosixBasedOS001 specific.

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::CreateProcess (const SS_String & *p_strFile_i*, const SS_String & *p_strProcessName_i*, const int *p_iPriority_i*, const StringList & *p_strlstArgv_i*, const char * *unix_user_name* = NULL, const long *p_lSpawnFlags_i* = iProcess_DEFAULT_PROCESS_FLAGS)

Summary

This method will create a process with the executable provided and mode as a calling parameter.

Parameters:

in	*p_strFile_i*	45 SS_String - Path and Filename of executable to create process for
in	*p_strProcessName_i*	46 SS_String - This is the name that will be registered to the OS for this process
in	*p_iPriority_i*	47 int - Priority for this process

in	*p_strlstArgv_i*	48 StringList - List of ARGV values for new process
in	*unix_user_name*	49 char* - unix user name
in	*p_lSpawnFlags_i*	50 long - Spawning flags.These are PosixBasedOS001 specific.

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::CreateProcess (const SS_String * *p_str_file*, char *const * *c_argv*, char * *environment_string*, const CL_ProcessAttr_t * *cl_attr*)

Summary

This method will create a process with the executable provided and mode as a calling parameter.

Parameters:

in	*p_strFile_i*	51 SS_String - Path and Filename of executable to create process for
in	*c_argv*	52 char - This is the argument that will be registered to the OS for this process

in	*environment_string*	53 char - enviroment for this process
in	*cl_attr*	54 CL_ProcessAttr_t - List of ARGV values for new process

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::CreateProcessWait (const SS_String & *p_strFile_i*)

Summary

This method will create a process with the executable provided.

Parameters:

in	*p_strFile_i*	55 SS_String - Path and Filename of executable to create process for

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::CreateProcessWait (const SS_String & *p_strFile_i*, const StringList & *p_strlstArguments_i*)

Summary

This method will create a PosixBasedOS001 process with the executable provided.

Parameters:

in	*p_strFile_i*	56 SS_String - Path and Filename of executable to create process for
in	*p_strlstArguments_i*	57 StringList - List of process calling arguments

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::DecreasePriorityByOne (void)

Summary

This method will decrease the priority for the process this object represents by one.

Parameters:

None	

Return values:

None	

Preconditions

-noprecondition

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::DisableAutoKill (void)[inline]

Summary

-none

Parameters:

None	

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

set m_fAutoKill as FALSE

Classification

Public

Type

sync only

See also:

None

BOOL Process::DoesProcessExist (void)

Summary

-Checkiftheprocessexisted

Parameters:

None	

Return values:

TRUE	- Process Exists
FALSE	- Process does not exist

Preconditions

-nopreconditions

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

void Process::EnableAutoKill (void)[inline]

Summary

Enable the Auto Kill

Parameters:

None	

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

set m_fAutoKill as TRUE

Classification

Public

Type

sync only

See also:

None

const char* const Process::GetExecutableFileName ()[inline]

Summary

Get executable file name

Parameters:

None	

Return values:

Pointer	58 Points to SS_String object that holds the path and executable file name for this process

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

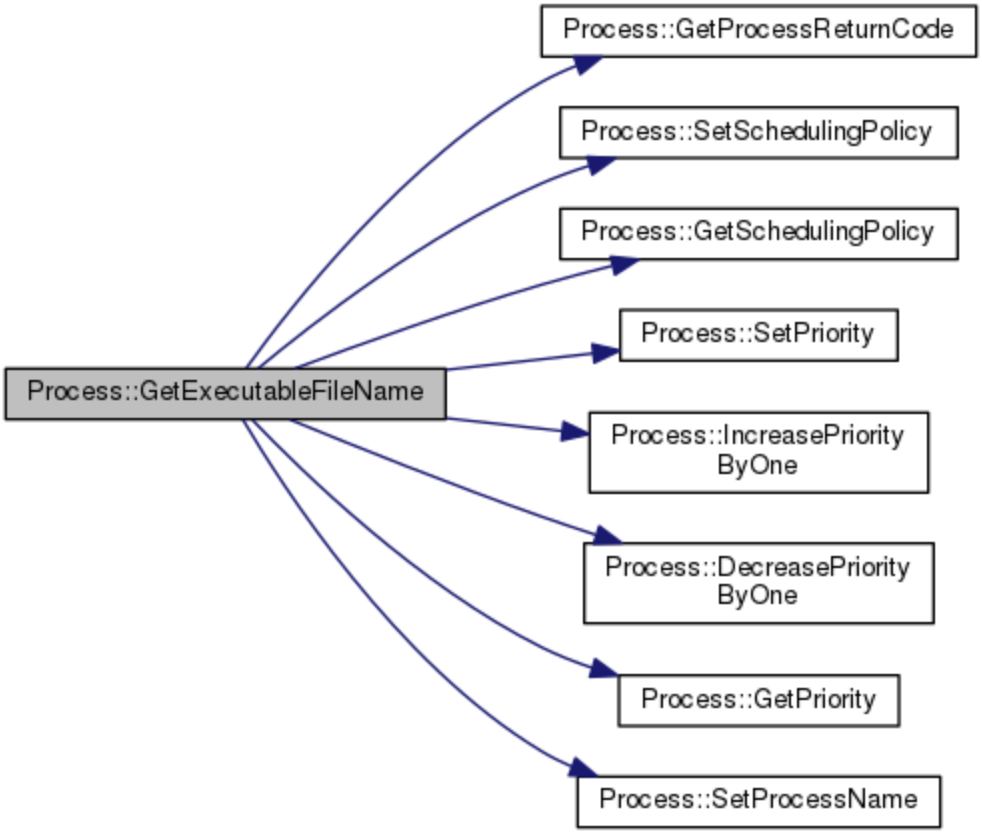
Type

sync only

See also:

None

Here is the call graph for this function:



IMAGE

int const Process::GetPriority (void)

Summary

This method will return to the caller the currently configured process priority.

Parameters:

None	

Return values:

int	priority - process priority

Preconditions

-nopreconditions

Change of the internal state

If the result of function sched_getparam(m_tProcessId, &cur_sch_params) is less than -1, then m_iErrorCode will be seted with errno.

Classification

Public

Type

sync only

See also:

None

int const Process::GetProcessId (void)[inline]

Summary

get process id

Parameters:

None	

Return values:

process	pid - the process pid of the process running.

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

Here is the call graph for this function:

IMAGE 

SS_String Process::GetProcessName (void) const[inline]

Summary

Get process name

Parameters:

None	

Return values:

SS_String	- Process Name

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

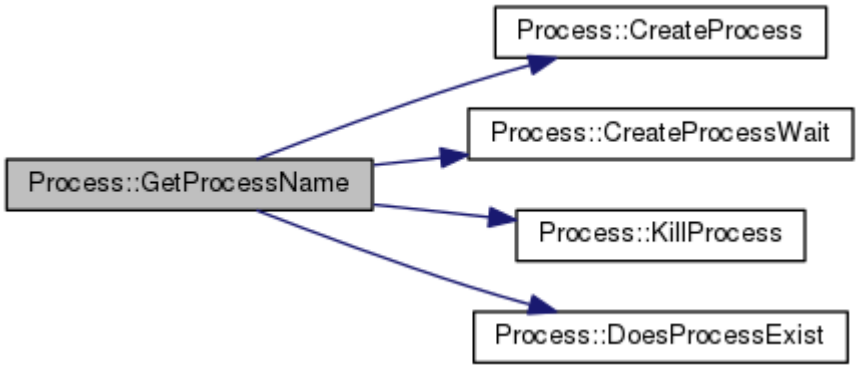
Type

sync only

See also:

None

Here is the call graph for this function:



IMAGE

int Process::GetProcessReturnCode (void)

Summary

This function will return the processes exit/return code.

Parameters:

None	

Return values:

process	exit code - Code for the last process to execute

Preconditions

-nopreconditions

Change of the internal state

set m_iReturnCode with WEXITSTATUS(iProcessReturn)

Classification

Public

Type

sync only

See also:

None

eProcessSchedulingPolicy const Process::GetSchedulingPolicy (void)

Summary

Get currently configured scheduling policy

Parameters:

None	

Return values:

FIFO	- A fixed priority scheduler in which the highest ready process runs until it blocks or is preempted by a higher priority process.
ROUND_ROBIN	- The same as FIFO, except processes at the same priority level time-slice.
OTHER	- A general time sharing scheduler in which a process decays in priority if it consumes too much processor before blocking. It reverts to its default priority when it blocks. Should it fail to run over a 2 second period and it has decayed then it's boosted one priority level up to a maximum of its default priority.

Preconditions

-nopreconditions

Change of the internal state

If the result of calling function sched_getscheduler(m_tProcessId) is -1 then set
m_iErrorCode with errno

Classification

Public

Type

sync only

See also:

None

void Process::IncreasePriorityByOne (void)

Summary

This method will increase the priority for the process this object represents by one.

Parameters:

None	

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

None

Classification

Public

Type

sync only

See also:

None

void Process::KillProcess (int *signal* = SIGKILL)

Summary

This method will send the specified signal to the process represented by this object

Parameters:

in	*signal*	int - signal

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

Intialize the objects m_iErrorCode member variable to 0

Classification

Public

Type

sync only

See also:

None

Process& Process::operator= (const Process & *p_rhs_i*)

=

Summary

Assignment Operator

Parameters:

in		

void Process::SetCallingArgumentList (const char * *p_pcArgv_i*[], const int *p_iArgc_i*)

Summary

This method will set the calling argument list that this object represents

Parameters:

in	*p_pcArgv_i*	const char *[] - Pointer to array of null terminated parameter list.
in	*p_iArgc_i*	const int - - Number of arguments in the array passed.

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

put the new arguments into m_strlstArgv

Classification

Public

Type

sync only

See also:

None

void Process::SetCallingArgumentList (const StringList & *p_strlstParameters_i*)

Summary

This method will set the calling argument list that this object represents.

Parameters:

in	*p_strlstParameters_i*	const StringList& - String list of arguments that will be used to start.

Return values:

None.	

Preconditions

nopreconditions.

Change of the internal state

set m_strlstArgv with p_strlstParameters_i.

Classification

Public.

Type

sync only

See also:

None

void Process::SetPriority (const int *p_iPriority_i*)

Summary

This method will change the priority for the process this object represents.

Parameters:

in	*p_iPriority_i*	int - Priority of the process

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

If the result of function sched_setparam(m_tProcessId, &cur_sch_params) is -1,then set m_iErrorCode with errno, else set m_iErrorCode with 0.

Classification

Public

Type

sync only

See also:

None

void Process::SetPriority (const pid_t *p_iPid_i*, const int *p_iPriority_i*)

:'

Summary

This method will change the priority for the specified process ID.

Parameters:

in	*p_iPid_i*	pid_t - PID of the process to change the priority
in	*p_iPriority_i*	int - Priority of the process

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

Classification

Public

Type

sync only

See also:

None

void Process::SetProcessName (const SS_String & *p_strProcessName_i*)

Summary

This method will set this objects process name member variable to the provided string value.

Parameters:

in	*p_strProcessName_i*	SS_String - Process Name to set the m_strProcessName member variable to

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

Using p_strProcessName_i to set m_strProcessName

Classification

Public

Type

sync only

See also:

None

void Process::SetSchedulingPolicy (const eProcessSchedulingPolicy *p_eSchedulingPolicy_i*)

Summary

This method will change the scheduling policy for the process this object represents.

Parameters:

in	*p_eSchedulingPolicy_i*	const eProcessSchedulingPolicy - Scheduling Policy for this process

eProcessSchedulingPolicy

- enum eProcessSchedulingPolicy {
  FIFO, A fixed priority scheduler in which the highest ready process runs until it\n blocks or is preempted by a higher priority process.
  ROUND_ROBIN, The same as FIFO, except processes at the same priority level time-slice.
  OTHER A general time sharing scheduler in which a process decays in priority if it\n consumes too much processor before blocking. It reverts to its default priority\n

when it blocks. Should it fail to run over a 2 second period and it has decayed\n
then it's boosted one priority level up to a maximum of its default priority.

};

Return values:

None	

Preconditions

-nopreconditions

Change of the internal state

-If the result of function is not 0,then set m_iErrorCode with errno
else set m_iErrorCode with 0.

Classification

Public

Type

sync only

See also:

None

The documentation for this class was generated from the following file:

ss_system_process.h

RAM_AccessIf Class Reference

RAM_AccessIf.

#include <ss_sm_ram_access.h>

Public Member Functions

RAM_AccessIf (RAM_WAKEUP_STATE wup_state=RAM_WAKEUP_STATE_DONT_CARE)
~RAM_AccessIf ()
E_CWORD33_Status getRamInfo (RAM_SM_INFO_t *p_boot_info)
E_CWORD33_Status setRamInfo (RAM_SM_INFO_t *p_boot_info)

Detailed Description

RAM_AccessIf.

Brief Introduction

Class to provide the function of RAM AccessIf

Constructor & Destructor Documentation

RAM_AccessIf::RAM_AccessIf (RAM_WAKEUP_STATE *wup_state* = RAM_WAKEUP_STATE_DONT_CARE)[explicit]

Summary:

Obtain the access permission to the system management information.

Parameters:

in	*wup_state*	RAM_WAKEUP_STATE - System maneger wakeup state type

RAM_WAKEUP_STATE
RAM_WAKEUP_STATE_DONT_CARE Don't care wakeup state
RAM_WAKEUP_STATE_BACKUP_NG Wakeup state is backup NG
RAM_WAKEUP_STATE_BATTERY_DOWN Wakeup state is battery down

Return values:

None	

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public

Type

None

See also:

~RAM_AccessIf, CL_LockMap, CL_LockGet

RAM_AccessIf::~RAM_AccessIf ()

~

Summary:

Release the access permission to the system management information.

Parameters:

None	

Return values:

None	

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public

Type

None

See also:

RAM_AccessIf, CL_LockRelease

Member Function Documentation

E_CWORD33_Status RAM_AccessIf::getRamInfo (RAM_SM_INFO_t * *p_boot_info*)

Summary:

Read the system management information.

Parameters:

out	*p_boot_info*	RAM_SM_INFO_t * - Destination buffer to read the system management information.

RAM_SM_INFO_t Structure body

```
typedef struct {
  char          signature_in[4];
  BOOL          isImmediateReset;
  BOOL          needRenotifyStartPrm;
  uint32_t      lastWakeupType;
  uint32_t      lastDramBackupStatus;
  uint32_t      lastResetStatus;
  char          productPrivate[RAM_PRODUCT_PRIVATE_MAX];
  char          order_name[SS_SM_ORDER_NAME_MAX];
  BOOL          isErrorReset;
  char          signature_out[4];
}RAM_SM_INFO_t;
```

Return values:

e_CWORD33_StatusOK	Reading success
e_CWORD33_StatusFail	Reading failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public

Type

Sync only(None communication)

See also:

setRamInfo

E_CWORD33_Status RAM_AccessIf::setRamInfo (RAM_SM_INFO_t * *p_boot_info*)

Summary:

Write the system management information.

Parameters:

in	*p_boot_info*	RAM_SM_INFO_t * - Original buffer to write the system management information

RAM_SM_INFO_t Structure body

```
typedef struct {
  char          signature_in[4];
  BOOL          isImmediateReset;
  BOOL          needRenotifyStartPrm;
  uint32_t      lastWakeupType;
  uint32_t      lastDramBackupStatus;
  uint32_t      lastResetStatus;
  char          productPrivate[RAM_PRODUCT_PRIVATE_MAX];
  char          order_name[SS_SM_ORDER_NAME_MAX];
```

```
    BOOL            isErrorReset;
    char            signature_out[4];
}RAM_SM_INFO_t;
```

Return values:

e_CWORD33_StatusOK	Write success
e_CWORD33_StatusFail	Write failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public

Type

Sync only(None communication)

See also:

getRamInfo

The documentation for this class was generated from the following file:

ss_sm_ram_access.h

ROM_AccessIf Class Reference

ROM_AccessIf.

#include <ss_sm_rom_access.h>

Public Member Functions

ROM_AccessIf ()
~ROM_AccessIf ()
E_CWORD33_Status Initialize ()
E_CWORD33_Status SetProductPrivate (UI_8 buf[ROM_PRODUCT_PRIVATE_MAX])
E_CWORD33_Status GetProductPrivate (UI_8 buf[ROM_PRODUCT_PRIVATE_MAX])
E_CWORD33_Status SystemInitialize (DRAM_BACKUP_STATE bkup_state)
E_CWORD33_Status GetBootMode (EBOOT_MODE *p_boot_mode)
E_CWORD33_Status SetBootMode (EBOOT_MODE boot_mode)
E_CWORD33_Status GetSignature (UI_32 *p_sig_value)
E_CWORD33_Status SetSignature (UI_32 sig_value)
E_CWORD33_Status SetActiveFlashloader (EACTIVE_FLASHLOADER active_flash_loader)
E_CWORD33_Status GetLastUserMode (EUSER_MODE *p_user_mode)
E_CWORD33_Status SetLastUserMode (EUSER_MODE user_mode)
E_CWORD33_Status GetTransportMode (ECONTROL_MODE *p_control_mode)
E_CWORD33_Status SetTransportMode (ECONTROL_MODE control_mode)
E_CWORD33_Status GetProductionMode (ECONTROL_MODE *p_control_mode)
E_CWORD33_Status SetProductionMode (ECONTROL_MODE control_mode)
E_CWORD33_Status GetLimpHomeCutOffReqMode (ECONTROL_MODE *p_control_mode)
E_CWORD33_Status SetLimpHomeCutOffReqMode (ECONTROL_MODE control_mode)
E_CWORD33_Status GetDataResetMode (EDATARESET_MODE *p_data_reset_mode)
E_CWORD33_Status GetDataResetModeFast (EDATARESET_MODE *p_data_reset_mode)
E_CWORD33_Status SetDataResetMode (EDATARESET_MODE data_reset_mode)
E_CWORD33_Status GetResetCount (UI_32 *p_reset_count)
E_CWORD33_Status SetResetCount (UI_32 reset_count)
E_CWORD33_Status GetLastIlgReset (ELASTILGRESET_MODE *p_last_ilg_reset)
E_CWORD33_Status SetLastIlgReset (ELASTILGRESET_MODE last_ilg_reset)
E_CWORD33_Status GetProgUpdateState (EPROGUPDATE_STATE *p_prog_update_state)
E_CWORD33_Status SetProgUpdateState (EPROGUPDATE_STATE prog_update_state)
E_CWORD33_Status GetErrLogCount (UI_32 *p_err_log_count)
E_CWORD33_Status SetErrLogCount (UI_32 err_log_count)
E_CWORD33_Status GetNextWakeupType (ENEXT_WAKEUP_TYPE *p_next_wakeup_type)
E_CWORD33_Status SetNextWakeupType (ENEXT_WAKEUP_TYPE next_wakeup_type)

Detailed Description

ROM_AccessIf.

Brief Introduction

Class to provide the function of ROM AccessIf

Constructor & Destructor Documentation

ROM_AccessIf::ROM_AccessIf ()

Summary:

Obtain the access permission to SYS area (secondary storage area).

Parameters:

None	

Return values:

None	

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

None

See also:

~ROM_AccessIf, CL_LockMap, CL_LockGet

ROM_AccessIf::~ROM_AccessIf ()

~

Summary:

Release the access permission to SYS area (secondary storage area).

Parameters:

None	

Return values:

None	

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

None

See also:

ROM_AccessIf, CL_LockRelease

Member Function Documentation

E_CWORD33_Status ROM_AccessIf::GetBootMode (EBOOT_MODE * *p_boot_mode*)

Summary:

Get boot mode information.

Parameters:

out	*p_boot_mode*	EBOOT_MODE* - Destination buffer to store information of boot mode.

EBOOT_MODE
APPLICATION_MODE
PROGRAMMING_MODE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetBootMode

E_CWORD33_Status ROM_AccessIf::GetDataResetMode (EDATARESET_MODE * *p_data_reset_mode*)

Summary:

Get reserved data reset mode information.

Parameters:

out	*p_data_reset_mode*	EDATARESET_MODE* - Destination buffer to store information of reserved data reset mode.

EDATARESET_MODE
DATARESET_NONE
DATARESET_USER
DATARESET_FACTORY

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetDataResetMode

E_CWORD33_Status ROM_AccessIf::GetDataResetModeFast (EDATARESET_MODE * *p_data_reset_mode*)

Summary:

Get reserved data reset mode information faster.This api can get correct value before calling SetDataResetMode.So it shoud be used only immediately after boot.

Parameters:

out	*p_data_reset_mode*	EDATARESET_MODE* - Destination buffer to store information of reserved data reset mode faster.

EDATARESET_MODE

DATARESET_NONE
DATARESET_USER
DATARESET_FACTORY

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetDataResetMode

E_CWORD33_Status ROM_AccessIf::GetErrLogCount (UI_32 * *p_err_log_count*)

Summary:

Get error logging count information.

Parameters:

out	*p_err_log_count*	UI_32* - Destination buffer to store information of error logging count.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetErrLogCount

E_CWORD33_Status ROM_AccessIf::GetLastIlgReset (ELASTILGRESET_MODE * *p_last_ilg_reset*)

Summary:

Get last illegal reset information.

Parameters:

out	*p_last_ilg_reset*	ELASTILGRESET_MODE* - Destination buffer to store information of last illegal reset.

ELASTILGRESET_MODE
LAST_ILGRESET_NORMAL
LAST_ILGRESET_NG

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetLastIlgReset

E_CWORD33_Status ROM_AccessIf::GetLastUserMode (EUSER_MODE * *p_user_mode*)

Summary:

Get last user mode information.

Parameters:

out	*p_user_mode*	EUSER_MODE* - Destination buffer to store information of user mode.

EUSER_MODE

USER_ON
USER_OFF

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetLastUserMode

E_CWORD33_Status ROM_AccessIf::GetLimpHomeCutOffReqMode (ECONTROL_MODE * *p_control_mode*)

Summary:

Get limp home cut off request mode information.

Parameters:

out	*p_control_mode*	ECONTROL_MODE* - Destination buffer to store information of limp home cut off request mode.

ECONTROL_MODE
DISABLE_MODE
ENABLE_MODE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetLimpHomeCutOffReqMode

E_CWORD33_Status ROM_AccessIf::GetNextWakeupType (ENEXT_WAKEUP_TYPE * *p_next_wakeup_type*)

Summary:

Get next wakeup type information.

Parameters:

out	*p_next_wakeup_type*	ENEXT_WAKEUP_TYPE* - Destination buffer to store information of next wakeup type.

ENEXT_WAKEUP_TYPE
NEXT_WAKEUP_TYPE_NONE
NEXT_WAKEUP_TYPE_COLD
NEXT_WAKEUP_TYPE_HOT

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetNextWakeupType

E_CWORD33_Status ROM_AccessIf::GetProductionMode (ECONTROL_MODE * *p_control_mode*)

Summary:

Get production mode information.

Parameters:

out	*p_control_mode*	ECONTROL_MODE* - Destination buffer to store information of production mode.

ECONTROL_MODE
DISABLE_MODE
ENABLE_MODE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetProductionMode

E_CWORD33_Status ROM_AccessIf::GetProductPrivate (UI_8 *buf*[ROM_PRODUCT_PRIVATE_MAX])

Summary:

Read the data information depending on the product specification from the secondary storage area.

Parameters:

out	*buf[ROM_PRODUCT_PRIVATE_MAX]*	UI_8[] - Storing destination of data information depending on the product specification

Definition of ROM_PRODUCT_PRIVATE_MAX
#define ROM_PRODUCT_PRIVATE_MAX 128

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetProductPrivate

E_CWORD33_Status ROM_AccessIf::GetProgUpdateState (EPROGUPDATE_STATE * *p_prog_update_state*)

Summary:

Get program update status information.

Parameters:

out	*p_prog_update_state*	EPROGUPDATE_STATE* - Destination buffer to store information of program update status.

typedef uint32_t EPROGUPDATE_STATE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetProgUpdateState

E_CWORD33_Status ROM_AccessIf::GetResetCount (UI_32 * *p_reset_count*)

Summary:

Get reset count information.

Parameters:

out	*p_reset_count*	UI_32* - Destination buffer to store information of reset count.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetResetCount

E_CWORD33_Status ROM_AccessIf::GetSignature (UI_32 * *p_sig_value*)

Summary:

Get signature information.

Parameters:

out	*p_sig_value*	UI_32 - Destination buffer to store signature information

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetSignature

E_CWORD33_Status ROM_AccessIf::GetTransportMode (ECONTROL_MODE * *p_control_mode*)

Summary:

Get transport mode information.

Parameters:

out	*p_control_mode*	ECONTROL_MODE* - Destination buffer to store information of transport mode.

ECONTROL_MODE
DISABLE_MODE
ENABLE_MODE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

SetTransportMode

E_CWORD33_Status ROM_AccessIf::Initialize ()

Summary:

Synchronize SYS area (secondary storage area) and the mirror data in DRAM

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Synchronization success
e_CWORD33_StatusFail	Synchronization failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

EL_mem_exram_mmap

E_CWORD33_Status ROM_AccessIf::SetActiveFlashloader (EACTIVE_FLASHLOADER *active_flash_loader*)

Summary:

Set active flash loader information.

Parameters:

in	*active_flash_loader*	EACTIVE_FLASHLOADER - Active flash loader information which write to secondary storage area.

EBOOT_MODE
APPLICATION_MODE
PROGRAMMING_MODE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetBootMode

E_CWORD33_Status ROM_AccessIf::SetBootMode (EBOOT_MODE *boot_mode*)

Summary:

Set boot mode information.

Parameters:

in	*boot_mode*	EBOOT_MODE - Boot mode information which write to the secondary storage area.

EBOOT_MODE
APPLICATION_MODE
PROGRAMMING_MODE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetBootMode

E_CWORD33_Status ROM_AccessIf::SetDataResetMode (EDATARESET_MODE *data_reset_mode*)

Summary:

Set reserve data reset mode information.

Parameters:

in	*data_reset_mode*	EDATARESET_MODE - Reserve data reset mode information which write to the secondary storage

		area.

EDATARESET_MODE
DATARESET_NONE
DATARESET_USER
DATARESET_FACTORY

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetDataResetMode

E_CWORD33_Status ROM_AccessIf::SetErrLogCount (UI_32 *err_log_count*)

Summary:

Get error logging count information.

Parameters:

out	*err_log_count*	UI_32* - Destination buffer to store information of error logging count.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetErrLogCount

E_CWORD33_Status ROM_AccessIf::SetLastIlgReset (ELASTILGRESET_MODE *last_ilg_reset*)

Summary:

Set last illegal reset information.

Parameters:

in	*last_ilg_reset*	ELASTILGRESET_MODE - Last illegal reset information which write to the secondary storage area.

ELASTILGRESET_MODE
LAST_ILGRESET_NORMAL
LAST_ILGRESET_NG

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetLastIlgReset

E_CWORD33_Status ROM_AccessIf::SetLastUserMode (EUSER_MODE *user_mode*)

Summary:

Set last user mode information.

Parameters:

in	*user_mode*	EUSER_MODE - User mode information which write to the secondary storage area.

EUSER_MODE
USER_ON
USER_OFF

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetLastUserMode

E_CWORD33_Status ROM_AccessIf::SetLimpHomeCutOffReqMode (ECONTROL_MODE *control_mode*)

Summary:

Set limp home cut off request mode information.

Parameters:

in	*control_mode*	ECONTROL_MODE - Limp home cut off request mode information which write to the secondary storage area.

ECONTROL_MODE
DISABLE_MODE
ENABLE_MODE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetLimpHomeCutOffReqMode

E_CWORD33_Status ROM_AccessIf::SetNextWakeupType (ENEXT_WAKEUP_TYPE *next_wakeup_type*)

Summary:

Get next wakeup type information.

Parameters:

out	*next_wakeup_type*	ENEXT_WAKEUP_TYPE* - Destination buffer to store information of next wakeup type.

ENEXT_WAKEUP_TYPE
NEXT_WAKEUP_TYPE_NONE
NEXT_WAKEUP_TYPE_COLD
NEXT_WAKEUP_TYPE_HOT

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetNextWakeupType

E_CWORD33_Status ROM_AccessIf::SetProductionMode (ECONTROL_MODE *control_mode*)

Summary:

Set transport mode information.

Parameters:

in	*control_mode*	ECONTROL_MODE - Production mode information which write to the secondary storage area.

ECONTROL_MODE
DISABLE_MODE
ENABLE_MODE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetProductionMode

E_CWORD33_Status ROM_AccessIf::SetProductPrivate (UI_8 *buf*[ROM_PRODUCT_PRIVATE_MAX])

Summary:

Write data information depending on the product specification to the secondary storage area.

Parameters:

in	*buf[ROM_PRODUCT_PRIVATE_MAX]*	UI_8[] - Data information depending on the product specification

Definition of ROM_PRODUCT_PRIVATE_MAX
#define ROM_PRODUCT_PRIVATE_MAX 128

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetProductPrivate

E_CWORD33_Status ROM_AccessIf::SetProgUpdateState (EPROGUPDATE_STATE *prog_update_state*)

Summary:

Set program update status information.

Parameters:

in	*prog_update_state*	EPROGUPDATE_STATE - Program update status information which write to the secondary storage area.

typedef uint32_t EPROGUPDATE_STATE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetProgUpdateState

E_CWORD33_Status ROM_AccessIf::SetResetCount (UI_32 *reset_count*)

Summary:

Set reset count information.

Parameters:

in	*reset_count*	UI_32 - Reset count information which write to the secondary storage area.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetResetCount

E_CWORD33_Status ROM_AccessIf::SetSignature (UI_32 *sig_value*)

Summary:

Set signature information.

Parameters:

in	*sig_value*	UI_32 - Signature information which write to the secondary storage area.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetSignature

E_CWORD33_Status ROM_AccessIf::SetTransportMode (ECONTROL_MODE *control_mode*)

Summary:

Set transport mode information.

Parameters:

in	*control_mode*	ECONTROL_MODE - Transport mode information which write to the secondary storage area.

ECONTROL_MODE
DISABLE_MODE
ENABLE_MODE

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

GetTransportMode

E_CWORD33_Status ROM_AccessIf::SystemInitialize (DRAM_BACKUP_STATE *bkup_state*)

Summary:

Initialize the handles used for ROM access.

Parameters:

in	*bkup_state*	

DRAM_BACKUP_STATE
DRAM_BACKUP_STATE_OK
DRAM_BACKUP_STATE_NG

Return values:

e_CWORD33_StatusOK	Initialize the handles used for ROM access success
e_CWORD33_StatusFail	Initialize the handles used for ROM access failed

Precondition:

None

Change in the internal status:

The change in the internal status does not occur by this API.

Classification:

Public english

Type

Sync only(None communication)

See also:

EL_mem_exram_mmap

The documentation for this class was generated from the following file:

ss_sm_rom_access.h

T_SS_SM_INIT_HOOK Struct Reference

Version up mode, Callback function.

#include <ss_system_manager_conf.h>

Public Attributes

BOOL bIsVupMode

Version up mode.

E_CWORD33_Status(* cbRebootNoticeFunc)(HANDLE hApp)

Call back function.

Detailed Description

Version up mode, Callback function.

The documentation for this struct was generated from the following file:

ss_system_manager_conf.h

T_SS_SM_START_DataStruct Struct Reference

#include <ss_system_manager_if.h>

Public Member Functions

T_SS_SM_START_DataStruct ()
T_SS_SM_START_DataStruct (EPWR_WAKEUP_FACTOR_TYPE f_startupReason, BOOL f_isUserModeOn, ESMDataResetModeInfo f_dataResetMode, EPWR_SC_SECURITY_STATUS f_securityStatus, EPWR_SC_WAKEUP_TYPE f_wakeupType, ESMDramBackupStatus f_dramBackupStatus, ESMResetStatus f_resetStatus, UI_32 f_resetCount)

Public Attributes

EPWR_WAKEUP_FACTOR_TYPE **startupReason**
BOOL **isUserModeOn**
ESMDataResetModeInfo **dataResetMode**
EPWR_SC_SECURITY_STATUS **securityStatus**
EPWR_SC_WAKEUP_TYPE **wakeupType**
ESMDramBackupStatus **dramBackupStatus**
ESMResetStatus **resetStatus**
UI_32 **errResetCount**

Detailed Description

Parameters of the following message.

SS_SM_START '_CWORD33_OnStart'/'evStart'

SS_SM_PRE_START '_CWORD33_OnPreStart'/'evPreStart'

SS_SM_BACKGROUND_START '_CWORD33_OnBackgroundStart'/'evBackgroundStart'

Constructor & Destructor Documentation

T_SS_SM_START_DataStruct::T_SS_SM_START_DataStruct ()[inline]

Constructor

T_SS_SM_START_DataStruct::T_SS_SM_START_DataStruct (EPWR_WAKEUP_FACTOR_TYPE *f_startupReason*, BOOL *f_isUserModeOn*, ESMDataResetModeInfo *f_dataResetMode*, EPWR_SC_SECURITY_STATUS *f_securityStatus*, EPWR_SC_WAKEUP_TYPE *f_wakeupType*, ESMDramBackupStatus *f_dramBackupStatus*, ESMResetStatus *f_resetStatus*, UI_32 *f_resetCount*)[inline]

Constructor

The documentation for this struct was generated from the following file:

59 ss_system_manager_if.h

T_SS_SM_UserModeOnOffNotification_Struct Struct Reference

#include <ss_system_manager_if.h>

Public Member Functions

T_SS_SM_UserModeOnOffNotification_Struct (BOOL f_bUserMode, EPWR_WAKEUP_FACTOR_TYPE f_startupReason, EPWR_USER_MODE_CHANGE_REASON_TYPE f_changeReason)

Public Attributes

BOOL **isUserModeOn**
EPWR_WAKEUP_FACTOR_TYPE **startupReason**
EPWR_USER_MODE_CHANGE_REASON_TYPE **userModeChangeReason**

Detailed Description

Data passed in the NTFY_SSSystemMgrPowerOnOff and * NTFY_SSSystemMgrUserMode notification messages *

The documentation for this struct was generated from the following file:
60 ss_system_manager_if.h

T_SystemManagerCpuResetInfo Struct Reference

CPU Reset Info.

#include <ss_system_manager_if.h>

Public Attributes

ESMCpuResetReason **resetReason**
CHAR **messageStr** [SS_SM_RESET_MSG_STR_SIZE]
CHAR **suffixStr** [SS_SM_SUFFIX_STR_SIZE]

Detailed Description

CPU Reset Info.

The documentation for this struct was generated from the following file:
61 ss_system_manager_if.h

Timer Class Reference

Timer.

#include <ss_system_timer.h>

Public Member Functions

Timer ()
Timer (HANDLE hdl, UI_32 id, CbFuncPtr CbFn)
virtual ~Timer ()
BOOL Initialize (HANDLE hApp, UI_32 id, CbFuncPtr CbFn)
void Reinitialize (UI_32 id)
void Shutdown ()
BOOL SetTime (UI_32 ss, UI_32 sms, UI_32 rs, UI_32 rms)
BOOL Start (UI_32 ss, UI_32 sms, UI_32 rs, UI_32 rms)
BOOL Start ()
BOOL Stop ()

Detailed Description

Timer.

Brief Introduction

Class to provide Timer function

Constructor & Destructor Documentation

Timer::Timer ()

Summary

Default constructor of Timer class.

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

sync only

See also:

~Timer, Initialize

Timer::Timer (HANDLE *hdl*, UI_32 *id*, CbFuncPtr *CbFn*)

Summary

Constructor of Timer class.

Parameters:

in	*hdl*	HANDLE - HANDLE for Application
in	*id*	UI_32 - Timer ID corresponding to the callback function
in	*CbFn*	CbFuncPtr - Pointer to a callback function to be called when the timer expires

Return values:

None	

Preconditions

Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop etc) has been done.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

~Timer, Initialize

virtual Timer::~Timer ()[virtual]

~

Summary

Destructor of Timer class.

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

None

See also:

Timer

Member Function Documentation

BOOL Timer::Initialize (HANDLE *hApp*, UI_32 *id*, CbFuncPtr *CbFn*)

Summary

Initialization of the Timer class.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*id*	UI_32 - Timer ID corresponding to the callback function
in	*CbFn*	CbFuncPtr - Pointer to a callback function to be called when the timer expires

Return values:

TRUE	Initialization success
FALSE	Initialization failed

Preconditions

Generation/initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop etc) has been done.

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

~Timer, Shutdown

void Timer::Reinitialize (UI_32 *id*)

62 Initialize()

63 **Summary**

Reinitialize the timer by specifying a timer value information

64 **Parameters:**

in	*id*	UI_32 - timer id that you want to reinitialize

65 **Return values:**

None	

66 **Preconditions**

That has been initialized in the Initialize().

67 **Change of the internal state**

set m_tTi.iCmd with id
using the result of function McOpenSender(_CWORD33_GetAppName(m_hApp)) to set m_hSnd
create a timer and give the handle to m_hTmr
set m_bInit with TRUE

68 **Classification**

Public

69 **Type**

Method only

70 **Detail**

Reinitialize the timer by specifying a timer value information,
create a new timer if it is not initialize.

71 **See also:**

Initialize

72

BOOL Timer::SetTime (UI_32 *ss*, UI_32 *sms*, UI_32 *rs*, UI_32 *rms*)

Summary

Set the timer value information.

Parameters:

in	*ss*	UI_32 - Timeout expiration time(sec)
in	*sms*	UI_32 - Timeout expiration time(nano sec)
in	*rs*	UI_32 - Timeout expiration time at the time of repeat(sec)
in	*rms*	UI_32 - Timeout expiration time at the time of repeat(nano sec)

Return values:

TRUE	Setting success
FALSE	Setting failed

Preconditions

That has been initialized in the Initialize().

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

Timer

void Timer::Shutdown ()

Summary

The end of the Timer class.

Parameters:

None	

Return values:

None	

Preconditions

That has been initialized in the Initialize().

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Sync only(None communication)

See also:

Timer, Initialize

BOOL Timer::Start (UI_32 *ss*, UI_32 *sms*, UI_32 *rs*, UI_32 *rms*)

Summary

The start of the timer by specifying a timer value information.

Parameters:

in	*ss*	UI_32 - Timeout expiration time(sec)
in	*sms*	UI_32 - Timeout expiration time(nano sec)
in	*rs*	UI_32 - Timeout expiration time at the time of repeat(sec)
in	*rms*	UI_32 - Timeout expiration time at the time of repeat(nano sec)

Return values:

TRUE	Stop success
FALSE	Stop failed

Preconditions

That has been initialized in the Initialize().

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method only

See also:

Stop

BOOL Timer::Start ()

Summary

Stop the timer.

Parameters:

None	

Return values:

TRUE	Stop success
FALSE	Stop failed

Preconditions

That has been initialized in the Initialize().
Set the timeout value in SetTime().

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method only

See also:

SetTime, Stop

BOOL Timer::Stop ()

Summary

Stop the timer.

Parameters:

None	

Return values:

TRUE	Stop success
FALSE	Stop failed

Preconditions

That has been initialized in the Initialize().
The timer has been started

Change of the internal state

The internal state is not changed.

Classification

Public

Type

Method only

See also:

Start

The documentation for this class was generated from the following file:

73 ss_system_timer.h

TimerCtrl Class Reference

OSprocess.

#include <ss_system_timer.h>

Public Member Functions

TimerCtrl ()
TimerCtrl (UI_32 ntimers)
VOID Initialize (HANDLE hApp)
VOID Shutdown ()
UI_32 CreateTimer (CbFuncPtr CbFn)
UI_32 DeleteTimer (UI_32 id)
UI_32 SetTimer (UI_32 id, UI_32 ss, UI_32 sms, UI_32 rs, UI_32 rms)
UI_32 StartTimer (UI_32 id, UI_32 ss, UI_32 sms, UI_32 rs, UI_32 rms)
UI_32 StartTimerMulti (UI_32 id, UI_32 ss, UI_32 sms, UI_32 rs, UI_32 rms, UI_32 subId)
UI_32 StartTimer (UI_32 id)
UI_32 StopTimer (UI_32 id)

Detailed Description

OSprocess.

TimerCtrl.

Brief Introduction

This class is the System Manager OS process abstraction.

Brief Introduction

Class to provide TimerCtrl function

Constructor & Destructor Documentation

TimerCtrl::TimerCtrl ()

Summary

Default Constructor, called when the class is instantiated.

Parameters:

None	

Return values:

None	

Preconditions

Change of the internal state

set m_hApp as NULLm_nTimersMax as DEFAULT_NTIMERS

Classification

Public

Type

sync only

See also:

~TimerCtrl, Initialize

TimerCtrl::TimerCtrl (UI_32 *ntimers*)

Summary

Constructor, called when the class is instantiated.

Parameters:

in	*ntimers*	UI_32 - number of timers objects in the pool

Return values:

None	

Preconditions

Change of the internal state

initialize var m_hApp and m_nTimersMax

Classification

Public

Type

sync only

See also:

~TimerCtrl, Initialize

Member Function Documentation

UI_32 TimerCtrl::CreateTimer (CbFuncPtr *CbFn*)

Summary

get a timer control from pool

Parameters:

in	*CbFn*	CbFn - Pointer to a callback function to be called when the timer expires

Return values:

Timer	id - If id is in m_aTimers
0	- If id is not in m_aTimers

Preconditions

That has been initialized in the Initialize().

Change of the internal state

insert one Timer to m_rTimers

Classification

Public

Type

sync only

See also:

DeleteTimer

UI_32 TimerCtrl::DeleteTimer (UI_32 *id*)

Summary

delete a timer from m_rTimers using the param.

Parameters:

in	*id*	UI_32 - timer id that you want to delete

Return values:

Timer	id -If id is in m_aTimers
0	- If id is not in m_aTimers

Preconditions

That has been initialized in the Initialize().

Change of the internal state

delete one Timer to m_rTimers that id is the param.
and put id of the timer into m_aTimers.

Classification

Public

Type

sync only

See also:

CreateTimer

VOID TimerCtrl::Initialize (HANDLE *hApp*)

Summary

Initialize the class after instantiation.

Parameters:

in	*hApp*	HANDLE - HANDLE to you applications hApp.

Return values:

None	

Preconditions

Change of the internal state

initialize var m_hApp and m_aTimers

Classification

Public

Type

sync only

See also:

~TimerCtrl, Initialize

UI_32 TimerCtrl::SetTimer (UI_32 *id*, UI_32 *ss*, UI_32 *sms*, UI_32 *rs*, UI_32 *rms*)

Summary

Sets the timer timeout values

Parameters:

in	*id*	UI_32 - timer id that you want to set
in	*ss*	UI_32 - Start time in seconds
in	*sms*	UI_32 - Start time in milliseconds
in	*rs*	UI_32 - Repeat time in seconds
in	*rms*	UI_32 - Repeat time in milliseconds

Return values:

Timer	id - If id is in m_aTimers
0	- If id is not in m_aTimers

Preconditions

That has been initialized in the Initialize().

Change of the internal state

set the timer infomation in m_rTimers with ss, sms, rs, rms

Classification

Public

Type

sync only

See also:

VOID TimerCtrl::Shutdown ()

Summary

handle something when shut down timer control.

Parameters:

None	

Return values:

None	

Preconditions

That has been initialized in the Initialize()..

Change of the internal state

Clear m_aTimer,shut down timer in m_rTimer and delete it.clear m_rTimer.

Classification

Public

Type

sync only

See also:

~TimerCtrl, Initialize

UI_32 TimerCtrl::StartTimer (UI_32 *id*, UI_32 *ss*, UI_32 *sms*, UI_32 *rs*, UI_32 *rms*)

Summary

Sets the timer timeout values and starts the timer.

Parameters:

in	*id*	UI_32 - timer id that you want to start
in	*ss*	UI_32 - Timeout expiration time(sec)
in	*sms*	UI_32 - Timeout expiration time(nano sec)
in	*rs*	UI_32 - Timeout expiration time at the time of repeat(sec)
in	*rms*	UI_32 - Timeout expiration time at the time of repeat(nano sec)

Return values:

Timer	id - If id is in m_aTimers
0	- If id is not in m_aTimers

Preconditions

That has been initialized in the Initialize().

Change of the internal state

None

Classification

Public

Type

sync only

See also:

Stop

UI_32 TimerCtrl::StartTimer (UI_32 *id*)

Summary

Start the timer.

Parameters:

in	*id*	UI_32 - timer id that you want to start

Return values:

Timer	id - If id is in m_aTimers
0	- If id is in not m_aTimers

Preconditions

That has been initialized in the Initialize().
Set the timeout value in SetTimer().

Change of the internal state

None

Classification

Public

Type

sync only

See also:

SetTime, Stop

UI_32 TimerCtrl::StartTimerMulti (UI_32 *id*, UI_32 *ss*, UI_32 *sms*, UI_32 *rs*, UI_32 *rms*, UI_32 *subId*)

Summary

The start of the timer by specifying a timer value information and reinitialize timer id.

Parameters:

in	*id*	UI_32 - timer id that you want to start
in	*ss*	UI_32 - Timeout expiration time(sec)
in	*sms*	UI_32 - Timeout expiration time(nano sec)
in	*rs*	UI_32 - Timeout expiration time at the time of repeat(sec)
in	*rms*	UI_32 - Timeout expiration time at the time of repeat(nano sec)
in	*subId*	UI_32 - using subId to reinitialize timer id

Return values:

Timer	id - If id is in m_aTimers
0	- If id is not in m_aTimers

Preconditions

That has been initialized in the Initialize().

Change of the internal state

None

Classification

Public

Type

sync only

See also:

Stop

UI_32 TimerCtrl::StopTimer (UI_32 *id*)

Summary

Stop the timer.

Parameters:

in	*id*	UI_32 - timer id that you want to stop

Return values:

Timer	id - If id is in m_aTimers
0	- If id is not in m_aTimers

Preconditions

That has been initialized in the Initialize().
Set the timeout value in SetTimer().

Change of the internal state

None

Classification

Public

Type

sync only

See also:

SetTimer, StartTimer

The documentation for this class was generated from the following file:

ss_system_timer.h

TSKM_DATAINIT_t Struct Reference

Public Attributes

TSKM_DATAINIT_TYPE_t **type**
E_CWORD33_Status(* **onCompInit**)(HANDLE hApp)

The documentation for this struct was generated from the following file:
tskm_type.h

File Documentation

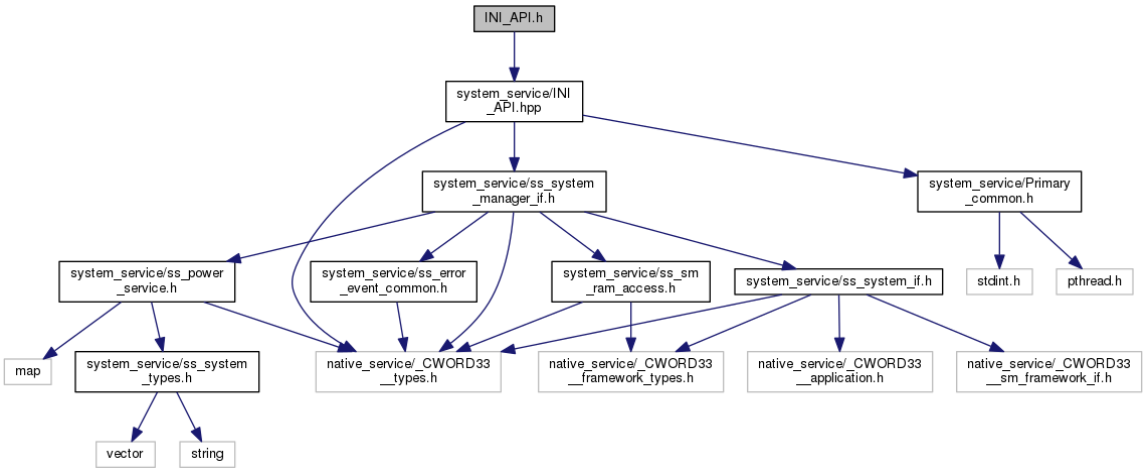
INI_API.h File Reference

This file include INI_API.hpp.

#include "system_service/INI_API.hpp"

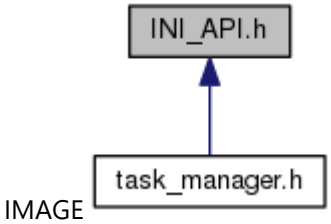
Include dependency graph for INI_API.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Detailed Description

This file include INI_API.hpp.

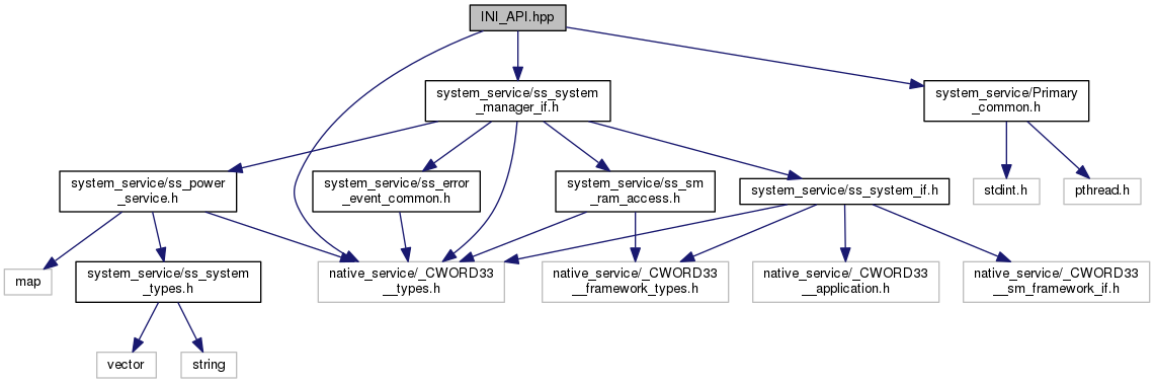
INI_API.hpp File Reference

This file provide api to operating INI.

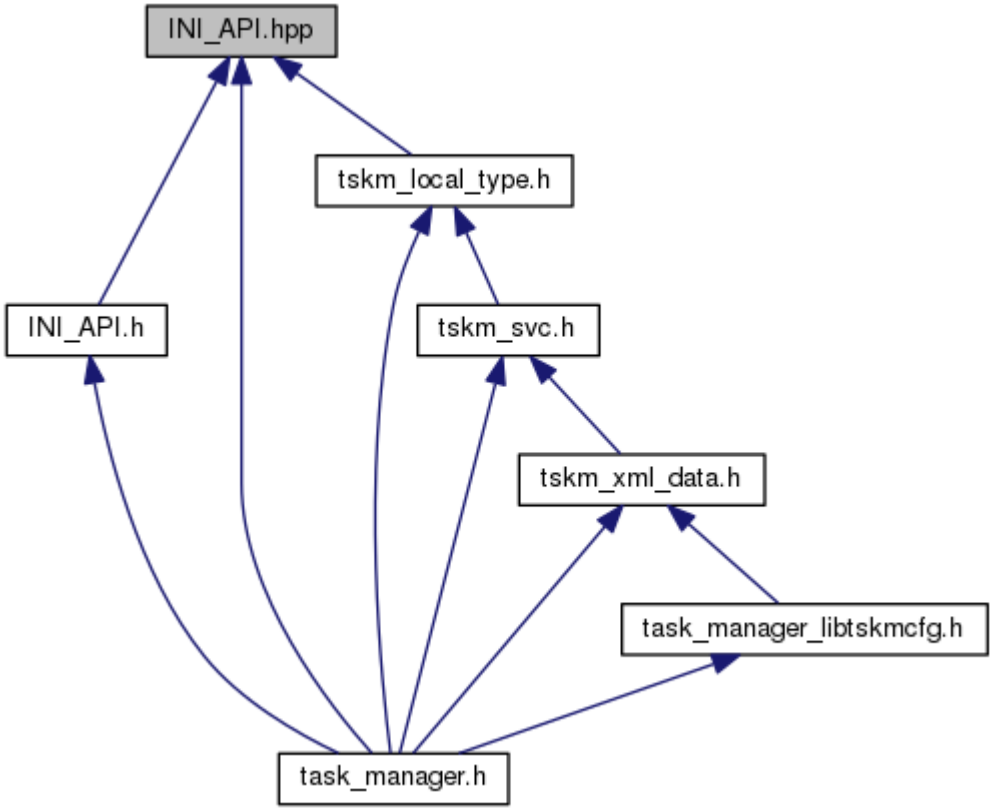
#include <native_service/_CWORD33__types.h>
#include <system_service/ss_system_manager_if.h>
#include "system_service/Primary_common.h"

Include dependency graph for INI_API.hpp:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _T_PRIM_PRM
struct T_PRIM_MONITOR_PRM

Macros

#define **INI_SUCCESS** 0 /* */
#define **INI_FALSE** -1 /* */
#define **INI_FD_MAX** 3
#define **INI_INITCOMP_NONE** 0x0000000000000000LLU
#define **INI_INITCOMP_ON_START** 0x0000000000000001LLU /* ON_START */
#define **INI_INITCOMP_NVM_ACCESS** 0x0000000000000002LLU /* NVM ACCESS */
#define **INI_INITCOMP_TEST0** 0x1000000000000000LLU /* TEST */
#define **INI_INITCOMP_TEST1** 0x2000000000000000LLU /* TEST */
#define **INI_INITCOMP_TEST2** 0x4000000000000000LLU /* TEST */
#define **INI_INITCOMP_TEST3** 0x8000000000000000LLU /* TEST */
#define **INI_TERMCOMP_NONE** 0x0000000000000000LLU
#define **INI_TERMCOMP_ACTIVITYMGR** 0x0000000000000001LLU /* ActivityManager */
#define **INI_TERMCOMP_RESIDENT** 0x0000000000000002LLU /* SVC */
#define **INI_TERMCOMP_TRANSIENT** 0x0000000000000004LLU /* SVC */
#define **INI_TERMCOMP_TEST0** 0x1000000000000000LLU /* TEST */
#define **INI_TERMCOMP_TEST1** 0x2000000000000000LLU /* TEST */
#define **INI_TERMCOMP_TEST2** 0x4000000000000000LLU /* TEST */
#define **INI_TERMCOMP_TEST3** 0x8000000000000000LLU /* TEST */
#define **INI_DEBUGDUMP**(args...) _INI_DEBUGDUMP(TRUE, ## args)
#define **INI_DEBUGDUMP_RAW**(args...) _INI_DEBUGDUMP(FALSE, ## args)

Typedefs

typedef struct _T_PRIM_PRM **T_PRIM_PRM**

Functions

int INI_Main (T_PRIM_PRM *p_prm, int argc, char *argv[])
void INI_ExitStart (void *rsv)
void INI_ExitDone (int status)
void * INI_GetPrivate ()
void * INI_GetHandle ()
int32_t INI_SetMonitorTimeout (uint32_t timeout)
int32_t INI_StepForkComp (uint64_t compId)
int32_t INI_AccOffComp (uint64_t compId)
int32_t INI_GetBootInfo (T_SS_SM_START_DataStructType *info)
int32_t INI_GetExtBootInfo (T_SS_SM_START_ExtDataStructType *info)
int32_t INI_Init (T_PRIM_PRM *p_prm, int argc, char *argv[], int *fdNum, int fdlist[INI_FD_MAX])
BOOL INI_Handler (fd_set *p_fds)
void INI_Term (void)
int32_t INI_SetMonitorState (T_PRIM_MONITOR_PRM *p_prm)
void **_INI_DEBUGDUMP** (BOOL bIsNeedSvcName, PCSTR f_cFormat,...)
E_CWORD33_Status _CWORD33_OnTouch (HANDLE hApp)

Detailed Description

This file provide api to operating INI.

interface_unified.h File Reference

include all interface_unified head files

```
#include "system_service/_CWORD33__system_application.h"
#include "system_service/_CWORD33__system_types.h"
#include "system_service/ss_boot_map.h"
#include "system_service/ss_client_names.h"
#include "system_service/ss_devicedetection_service.h"
#include "system_service/ss_devicedetection_service_ifc.h"
#include "system_service/ss_devicedetection_service_notifications.h"
#include "system_service/ss_devicedetection_service_protocol.h"
#include "system_service/ss_devicedetection_service_types.h"
#include "system_service/ss_error_event_common.h"
#include "system_service/ss_error_message.h"
#include "system_service/ss_heartbeat_if.h"
#include "system_service/ss_heartbeat_notifications.h"
#include "system_service/ss_heartbeat_service_protocol.h"
#include "system_service/ss_last_to_order.h"
#include "system_service/ss_logger_service.h"
#include "system_service/ss_logger_service_ifc.h"
#include "system_service/ss_logger_service_notifications.h"
#include "system_service/ss_logger_service_protocol.h"
#include "system_service/ss_services.h"
#include "system_service/ss_sm_client_if.h"
#include "system_service/ss_sm_thread_names.h"
#include "system_service/ss_string_maps.h"
#include "system_service/ss_sys_boot_area_map.h"
#include "system_service/ss_system_if.h"
#include "system_service/ss_system_manager_if.h"
#include "system_service/ss_system_manager_notifications.h"
#include "system_service/ss_system_manager_protocol.h"
#include "system_service/ss_system_process.h"
#include "system_service/ss_system_timer.h"
#include "system_service/ss_system_types.h"
#include "system_service/ss_templates.h"
#include "system_service/ss_test_clients.h"
#include "system_service/ss_version.h"
#include "system_service/ss_logger_store_logs.h"
#include "system_service/ss_devicedetection_service_local.h"
#include "system_service/ss_devicedetection_service_protocol_local.h"
#include "system_service/ss_devicedetection_service_types_local.h"
#include "system_service/ss_error_event_common_local.h"
```

```
#include "system_service/ss_sm_client_if_local.h"
#include "system_service/ss_sm_thread_names_local.h"
#include "system_service/ss_system_manager_if_local.h"
#include "system_service/ss_system_manager_notifications_local.h"
#include "system_service/ss_system_manager_protocol_local.h"
#include "system_service/ss_logger_service_local.h"
#include "system_service/ss_power_service_if.h"
#include "system_service/ss_power_service_notifications.h"
#include "system_service/ss_power_service_protocol.h"
#include "system_service/ss_power_service.h"
#include "system_service/ss_power_service_local.h"
#include "system_service/ss_power_service_notifications_local.h"
```

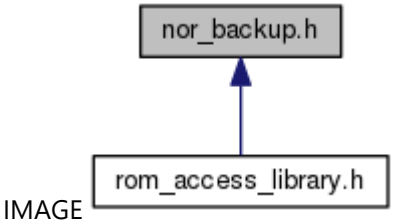
Detailed Description

include all interface_unified head files

nor_backup.h File Reference

This file provides API for get nor flash information.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define RET_DEV_NORMAL 0
#define RET_DEV_ERR_PARAM 3
#define RET_DEV_ERROR 4
#define RET_DEV_RD_ERR_OTHER 1
#define RET_DEV_RD_ERR 2
#define RET_DEV_RD_ERR_1 -1
#define RET_DEV_RD_ERR_2 -2
#define RET_DEV_RD_ERR_3 -3
#define RET_DEV_RD_ERR_4 -4
#define RET_DEV_WT_ERR_OTHER 1
#define RET_DEV_WT_ERR 2

Functions

int mtdn_backup_Read (const char *path_name, int i_id, int i_offset, int i_size, void *p_buff)
int mtdn_backup_Write (const char *path_name, int i_id, int i_offset, int i_size, void *p_buff)

Detailed Description

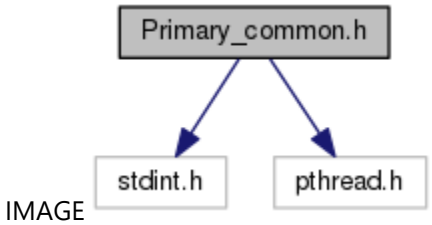
This file provides API for get nor flash information.

Primary_common.h File Reference

This file contains declaration of common enum and structures.

#include <stdint.h>
#include <pthread.h>

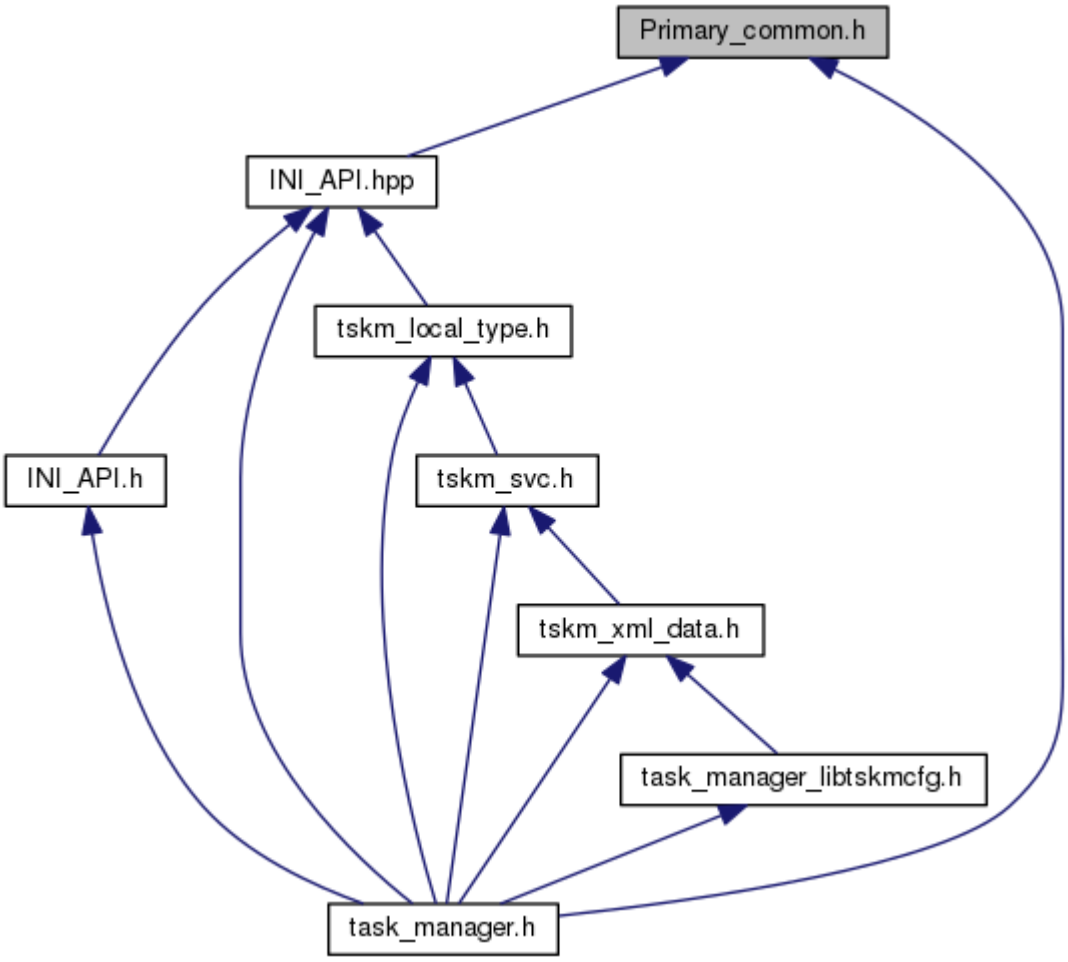
Include dependency graph for Primary_common.h:



IMAGE

This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct PRIM_SHAREDATA_TBL
struct PRIM_FORK_TBL
struct PRIM_EXFUNC_TBL
struct PRIM_BACKUPCHK_TBL
struct PRIM_ACCOFF_TBL

Macros

#define **SRAM_INIT** (int)0 /* */
#define **SRAM_CHK** (int)1 /* */
#define **SRAM_MAPVER** (int)2 /* Version */
#define **SRAM_MAPVER_CHK** (int)3 /* Version + */
#define **SRAM_SYSTEM_INIT** (int)4 /* #37 */
#define **SRAM_CONFIG_INIT** (int)5 /* #37 */
#define **SRAM_CHK_OK** (int)0 /* SRAM */
#define **SRAM_CHK_CHG** (int)1 /* SRAM */
#define **SRAM_CHK_NG** (int)-1 /* SRAM */
#define **PRIM_SHM_NAME_MAX** 32 /* */

Typedefs

typedef unsigned int **PRI_DWORD**

Enumerations

enum **_PRIM_FORK_STEP** { **PRIM_STEPFORK_FIRST** = ***, **PRIM_STEPFORK_SECOND**, **PRIM_STEPFORK_THIRD**, **PRIM_STEPFORK_FOURTH**, **PRIM_STEPFORK_FIFTH**, **PRIM_STEPFORK_SIXTH**, **PRIM_STEPFORK_SEVENTH**, **PRIM_STEPFORK_EIGHTH**, **PRIM_STEPFORK_NINTH**, **PRIM_STEPFORK_TENTH**, **PRIM_STEPFORK_MAX** }
enum **_PRIM_ACCOFF_STEP** { **PRIM_ACCOFF_FIRST** = ***, **PRIM_ACCOFF_SECOND**, **PRIM_ACCOFF_THIRD**, **PRIM_ACCOFF_FOURTH**, **PRIM_ACCOFF_FIFTH**, **PRIM_ACCOFF_MAX** }

Detailed Description

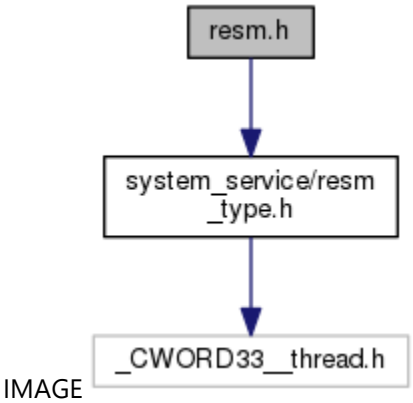
This file contains declaration of common enum and structures.

resm.h File Reference

This file contains declaration of APIs for resm library.

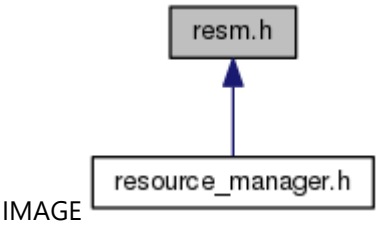
#include "system_service/resm_type.h"

Include dependency graph for resm.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

RESM_ERR_t RESM_Open (const RESM_RSV_t *p_prim, uint32_t *p_ssnId)
RESM_ERR_t RESM_Close (uint32_t ssnId)
RESM_ERR_t RESM_ReqEvent (uint32_t ssnId, const RESM_REQ_EVENT_t *p_reqEvent)
RESM_ERR_t RESM_GetEventFd (uint32_t ssnId, int *p_fd)
RESM_ERR_t RESM_GetEvent (uint32_t ssnId, RESM_EV_t *p_evFlag)
RESM_ERR_t RESM_GetStatus (uint32_t ssnId, RESM_STATUS_t *p_status)

Detailed Description

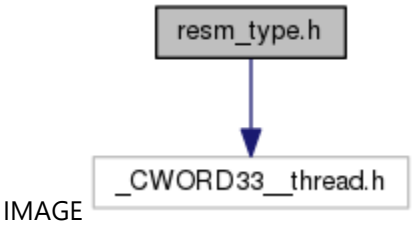
This file contains declaration of APIs for resm library.

resm_type.h File Reference

This file contains declaration of common enum and structures for resm library.

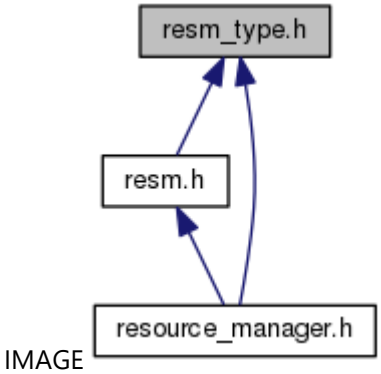
#include <_CWORD33__thread.h>

Include dependency graph for resm_type.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct RESM_REQ_EVENT_t
struct RESM_INET_STATUS_t
struct RESM_STATUS_t
struct RESM_REQ_EVENT_t.prm
struct RESM_INET_STATUS_t.ifInfo

Macros

#define **NTFY_ResourceMgr_Availability** MN_SS_RESOURCEMGR"/Availability"
#define **RESM_ERR_t** int32_t
#define **RESM_E_OK** (0)
#define **RESM_E_PAR** (-1)
#define **RESM_E_NG** (-2)
#define **RESM_EV_t** uint32_t
#define **RESM_EV_NOP** (0x00)
#define **RESM_EV_MEM** (0x01)
#define **RESM_EV_NAND_STATUS** (0x02)
#define **RESM_RSV_t** int32_t

#define **RESM_INET_IF_MAX** 5 /* tentative */
#define **HWADDR_LEN** (6)

Enumerations

enum **RESM_NAND_WRITE_STATUS_t** { **RESM_NAND_WRITE_ENABLE** = ***,
RESM_NAND_WRITE_DISABLE = -1 }

Detailed Description

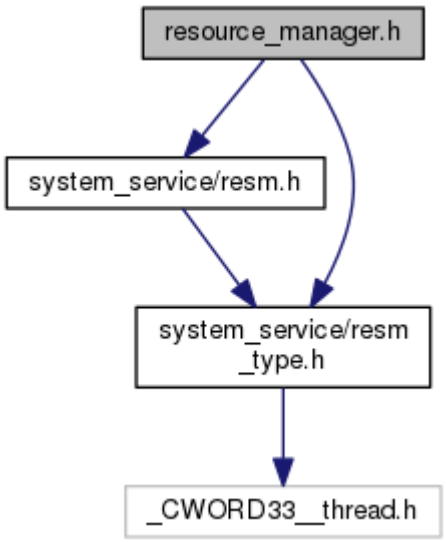
This file contains declaration of common enum and structures for resm library.

resource_manager.h File Reference

include all resm_library head files

#include "system_service/resm.h"
#include "system_service/resm_type.h"

Include dependency graph for resource_manager.h:



IMAGE

Detailed Description

include all resm_library head files

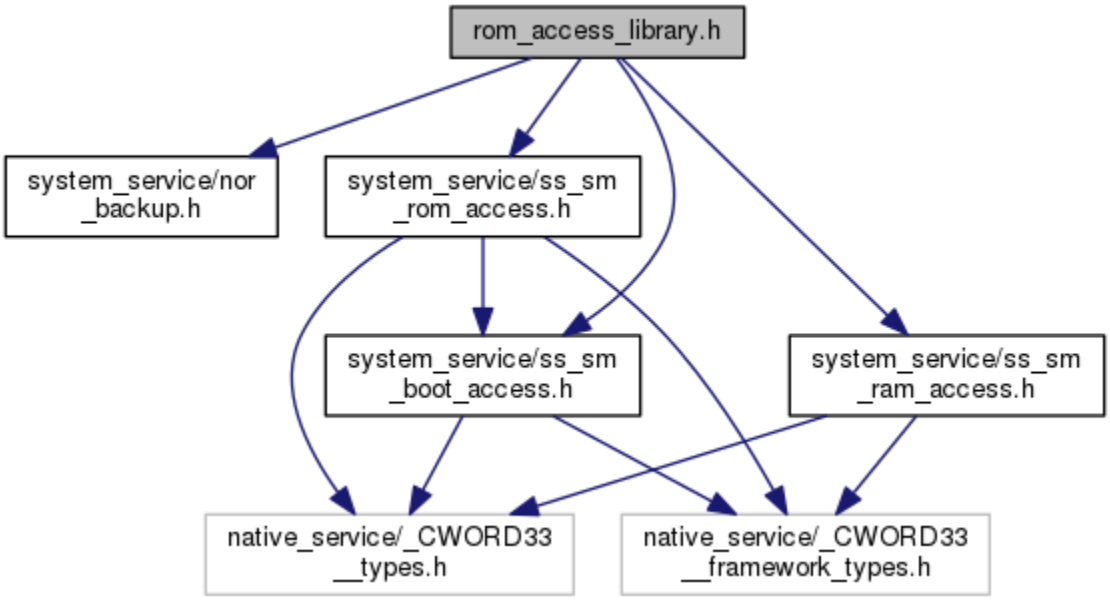
rom_access_library.h File Reference

This file include all head file of rom and nor access library.

#include "system_service/nor_backup.h"
#include "system_service/ss_sm_rom_access.h"
#include "system_service/ss_sm_boot_access.h"
#include "system_service/ss_sm_ram_access.h"

Include dependency graph for rom_access_library.h:

IMAGE



Detailed Description

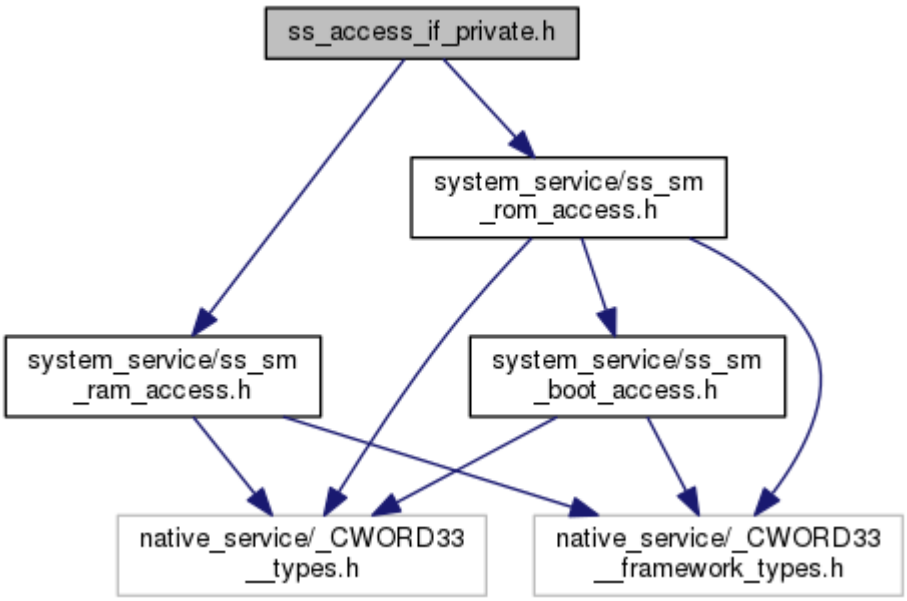
This file include all head file of rom and nor access library.

ss_access_if_private.h File Reference

This file contains declaration of structures.

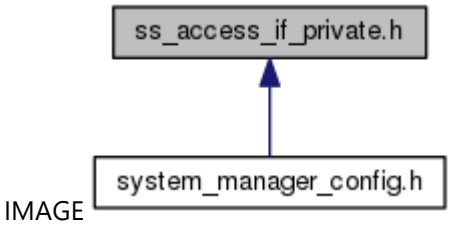
#include <system_service/ss_sm_rom_access.h>
#include <system_service/ss_sm_ram_access.h>

Include dependency graph for ss_access_if_private.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct RAM_AccessPrivate

RAM Access Private data define. *More...*

struct ROM_AccessPrivate

ROM Access Private data define. *More...*

union RAM_AccessPrivate.__unnamed__
struct RAM_AccessPrivate.__unnamed__.__unnamed__

CWORD80 error define *More...*

union ROM_AccessPrivate.__unnamed__
struct ROM_AccessPrivate.__unnamed__.__unnamed__

CWORD80 error count, connect state More...

Macros

#define **SS_CONN_NONE** 0
#define **SS_CONN_HAS__CWORD80_** 1
#define **SS_CONN_HASNOT__CWORD80_** 2

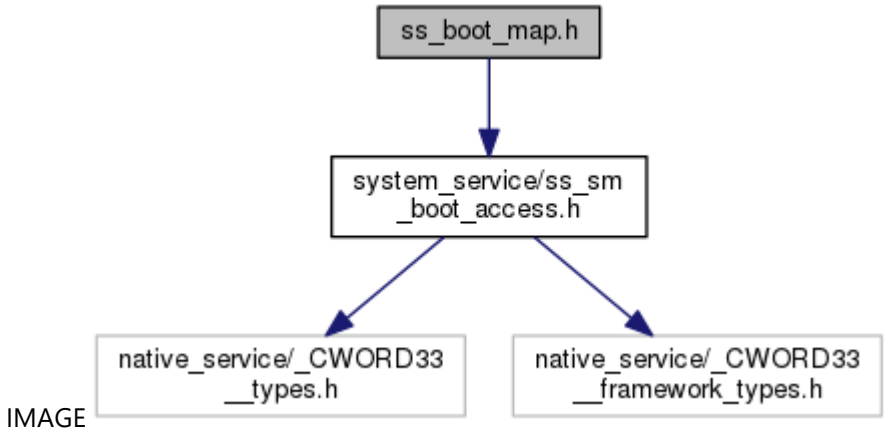
Detailed Description

This file contains declaration of structures.

ss_boot_map.h File Reference

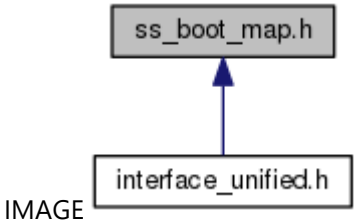
#include <system_service/ss_sm_boot_access.h>

Include dependency graph for ss_boot_map.h:



IMAGE

This graph shows which files directly or indirectly include this file:

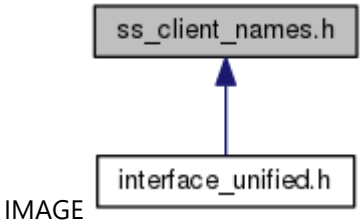


IMAGE

ss_client_names.h File Reference

This file supports contains the queue names of the client services that System Services needs to know about.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **SERVICE__CWORD69_** "_CWORD69_"
#define **SERVICE_AS_AUDIO** "AS_AudioService"
#define **SERVICE_AS_MODE** "AS_ModeService"
#define **SERVICE_BR_BROWSER** "BR_BrowserService"
#define **SERVICE_CAN_SHADOW** "VS_CANShadow"
#define **SERVICE__CWORD51__BT** "_CWORD51__BT"
#define **SERVICE__CWORD51__DR** "_CWORD51__DR"
#define **SERVICE__CWORD51___CWORD105_** "_CWORD51___CWORD105_"
#define **SERVICE_DAB** "RS_Master_Tuner"
#define **SERVICE_HMI** "HMI"
#define **SERVICE_HMI__CWORD46_** "HMI__CWORD46_Service"
#define **SERVICE_HMI__CWORD77_** "HMI__CWORD77_Service"
#define **SERVICE_IPC_MP_SHADOW** "PS_IPC_MP_Shadow"
#define **SERVICE_KEY_HANDLER** "PS_KeyHandler"
#define **SERVICE_MM_ICD** "MM_ICDService"
#define **SERVICE_MM_MEDIA** "MM_MediaService"
#define **SERVICE_MM__CWORD21_** "MM__CWORD21_Service"
#define **SERVICE_NAV** "NAV_NavigationService"
#define **SERVICE_NAV_LOC** "NAV_LocationService"
#define **SERVICE_NS_NPP** "NS_NPPService"
#define **SERVICE_NS_SHARED_MEM** "NS_SharedMem"
#define **SERVICE_NW_BT** "NW_BluetoothService"
#define **SERVICE_NW_CONNECTION** "NW_ConnectionService"
#define **SERVICE_NW_MESSAGING** "NW_MessagingService"
#define **SERVICE_NW_PHONE** "NW_PhoneService"
#define **SERVICE_NW_PHONE_BOOK** "NW_PhoneBookService"
#define **SERVICE_PSM_SHADOW** "PS_PSMShadow"
#define **SERVICE_PS_IPC** "PS_IPC"
#define **SERVICE_PS_SWDL_SHADOW** "PS_SoftwareUpdateShadow"
#define **SERVICE_SDARS** "RS_XSDARSService"
#define **SERVICE_TUNER** "RS_Radio"

#define **SERVICE_VS__CWORD4_** "VS__CWORD4_Service"
#define **SERVICE_VS_DIAG** "VS_DiagService"
#define **SERVICE_VS_DIAG_DISP** "DiagDispatcherService"
#define **SERVICE_VS__CWORD105_** "VS__CWORD105_Service"
#define **SERVICE_VS_RVC** "VS_RVCService"
#define **SERVICE_VS_VEHICLE** "VS_VehicleService"

Detailed Description

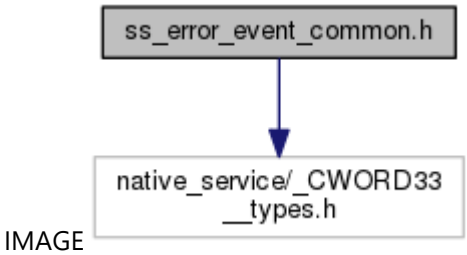
This file supports contains the queue names of the client services that System Services needs to know about.

ss_error_event_common.h File Reference

This file supports SM logging.

#include <native_service/_CWORD33__types.h>

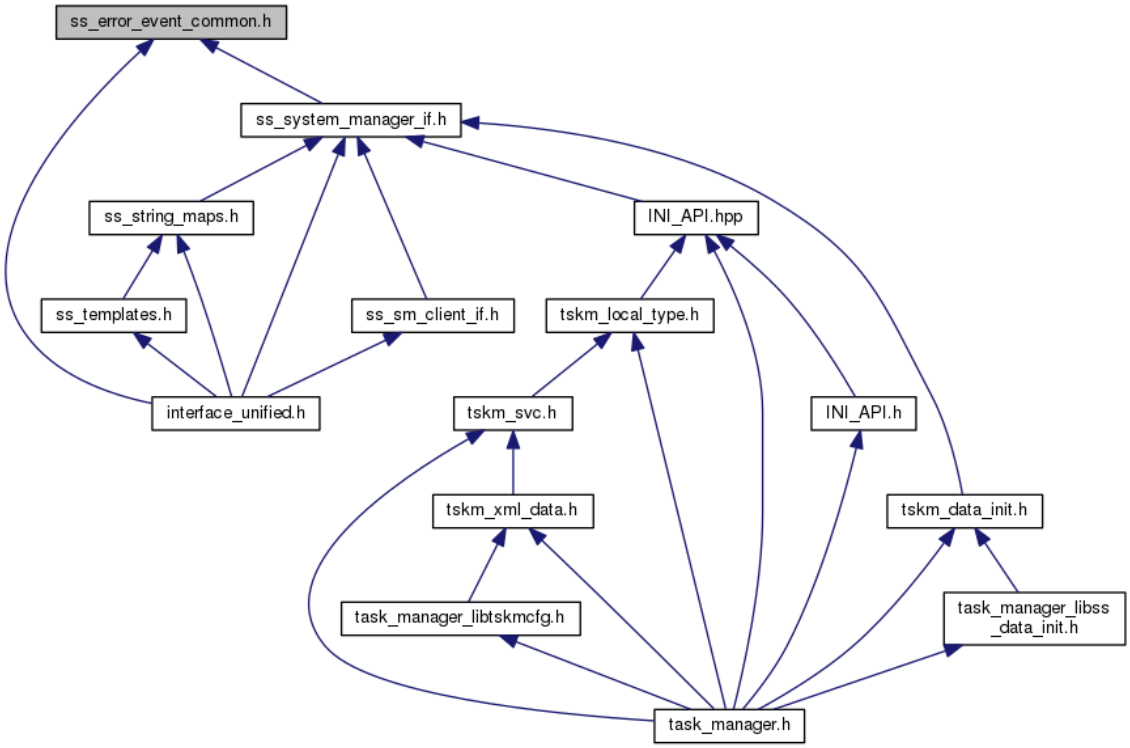
Include dependency graph for ss_error_event_common.h:



IMAGE

This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct ARTIFACT_RESPONSE

Enumerations

enum **EErrorEventType** { **eErrorEventTypeProcessCrash** = ***,
eErrorEventTypeHeartBeatFailure, **eErrorEventTypeSystemLowMemory**,

eErrorEventTypeReserved1, **eErrorEventTypeUserInvokedUserForceReset**,
eErrorEventTypeReserved2, **eErrorEventTypeReserved3**,
eErrorEventTypeUserInvokedCollectAllLogs, **eErrorEventTypeBootMicroReset**,
eErrorEventTypeProcessExit, **eErrorEventTypeUserInvokedCollectScreenShot**,
eErrorEventTypeUserInvokedCollect_CWORD33_Logs, **eErrorEventTypeEelExport**,
eErrorEventType_CWORD33_EmmcLogs, **eErrorEventTypeDiagEvent**,
eErrorEventTypeCanEvent, **eErrorEventTypeDtcEvent**, **eErrorEventTypeModConnFailed**,
eErrorEventTypeStartRespFailed, **eErrorEventTypeUserInvokedCollectDevLogs**,
eErrorEventTypeModuleInvokedResetRequest,
eErrorEventTypeModuleInvokedCollectDebugLogs,
eErrorEventTypeUserInvokedClearLogs, **eErrorEventTypeUserInvokedCollectNaviLog**,
eErrorEventTypeGroupRelaunch, **eErrorEventTypeMaxValue** }

enum **EArtifactId** { **eArtifactId_CWORD33_DebugLog** = ***, **eArtifactIdTransmitLog**,
eArtifactIdPerformanceLog, **eArtifactIdBootMicroLog**, **eArtifactIdSystemDataCsv**,
eArtifactIdShowMemTxt, **eArtifactIdProcessCore**, **eArtifactIdDebugDumpLog**,
eArtifactIdKernelLog, **eArtifactIdDRInitialLog**, **eArtifactIdDRLocationLog**,
eArtifactIdCpuHighLoadMonteCarloLogs, **eArtifactIdMetaCoreLogs**, **eArtifactIdCmsLogs**,
eArtifactIDInternalDTC, **eArtifactIdComDebugLog**, **eArtifactIdRadioDebugLog**,
eArtifactIdConnectivityDebugLog, **eArtifactIdKernelBootLog**,
eArtifactIdGraphicsDebugLog, **eArtifactIdSysLog**, **eArtifactIdPstoreLog**,
eArtifactIdClearAllLog, **eArtifactIdNaviLog**, **eArtifactIdWinSysLog**, **eArtifactIdAppFwLog**,
eArtifactIdTomoyoLog, **eArtifactIdScreenShot**, **eArtifactIdDebugFolder2Content**,
eArtifactIdDebugFolderContent, **eArtifactIdMaxValue** }

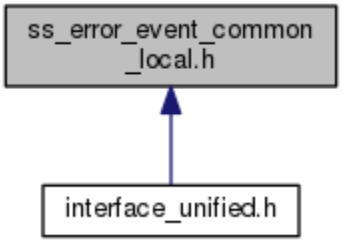
Detailed Description

This file supports SM logging.

ss_error_event_common_local.h File Reference

This file supports SM logging.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **SS_ERROR_EVENT_START_REQ_TO_SEC** (5)
#define **SS_ERROR_EVENT_DEBUG_DUMP_RSPN_TO_SEC** (1)
#define **SS_ERROR_EVENT_BOOT_MICRO_LOG_RESPONSE_TO_SEC** (2)
#define **SS_ERROR_EVENT_CORE_FILE_POLL_TO_MS** (100)

Enumerations

enum **EErrorEventPrio** { **eErrorEventPrioDiagLogRequest** = ***,
eErrorEventPrioUserInvokedCollectAllLogs = ***,
eErrorEventPrioUserInvokedCollectScreenShot = ***,
eErrorEventPrioUserInvokedCollect_CWORD33_Logs = ***,
eErrorEventPrioModuleInvokedCollectDebugLogs = ***,
eErrorEventPrioUserInvokedUserForceReset = ***, **eErrorEventPrioEelStorageFilePresent** = ***, **eErrorEventPrioEmmcLogsFilePresent** = ***, **eErrorEventPrioDefault** = *** }

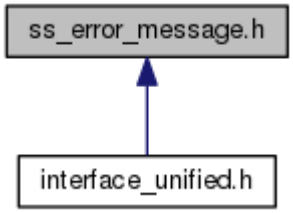
Detailed Description

This file supports SM logging.

ss_error_message.h File Reference

This file contains declaration of system error message.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **SS_ERROR_MSG_USB_DISCON** "UsbDisconnected"

#define **SS_ERROR_MSG__CWORD84__ABNORMAL_SHUTDOWN** "_CWORD84_AbnormalShutdown"

Detailed Description

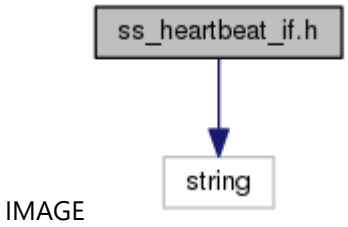
This file contains declaration of system error message.

ss_heartbeat_if.h File Reference

This file supports the System Manager Heartbeat Service.

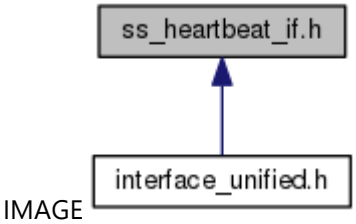
#include <string>

Include dependency graph for ss_heartbeat_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct THeartBeatSession
class CHeartBeatServiceIf

Detailed Description

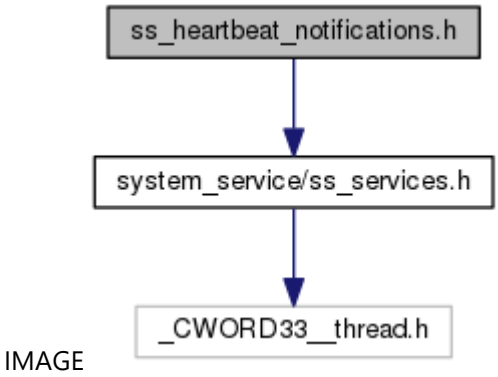
This file supports the System Manager Heartbeat Service.

ss_heartbeat_notifications.h File Reference

This file supports the System Manager Heartbeat Service.

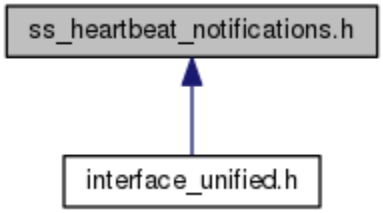
#include "system_service/ss_services.h"

Include dependency graph for ss_heartbeat_notifications.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define NTFY_HeartBeatAvailability SERVICE_HEARTBEAT"/Availability"

Indicates if HeartBeat is available or unavailable.

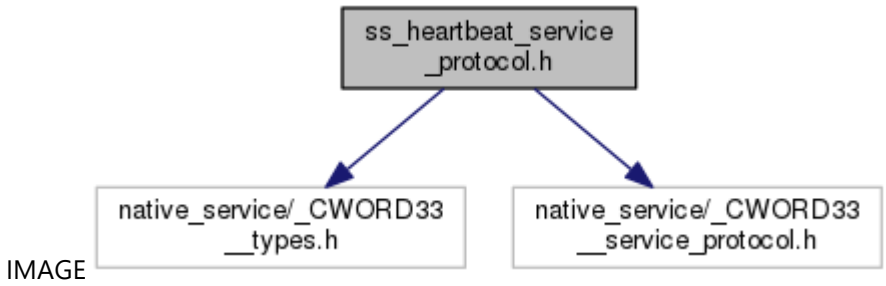
Detailed Description

This file supports the System Manager Heartbeat Service.

ss_heartbeat_service_protocol.h File Reference

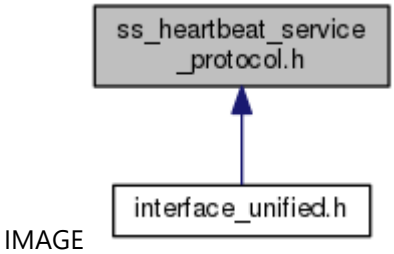
Handle HeartBeat Module Start and Stop notifications.

#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__service_protocol.h>
Include dependency graph for ss_heartbeat_service_protocol.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Typedefs

typedef enum T_SMHBProtocolMessages EHBProtocolMessages

HeatBeat Protocol Command heartbeat service protocol > define all protocol messages in and out of heart beat thread that are pertinent to the functionality offered by heart beat thread.

Enumerations

enum T_SMHBProtocolMessages { SS_HEARTBEAT_PRINT_CONNECTIONS = ***, SS_HEARTBEAT_PRINT_STACK = ***, SS_HEARTBEAT_PERIODIC_STATUS_REQ = ***, SS_HEARTBEAT_PERIODIC_RESP = ***, SS_HEARTBEAT_START = ***, SS_HEARTBEAT_STOP = ***, SS_HEARTBEAT_DELETE_MODULE_ENTRY = ***, SS_HEARTBEAT_MSG_CATEGORY_REPORT = ***, SS_HEARTBEAT_ERROR_DETECTED = ***, SS_HEARTBEAT_APPEND_MODULE_ENTRY = ***, SS_HEARTBEAT_AVAIL_CHECK_REQ = ***, SS_HEARTBEAT_AVAIL_CHECK_RESP = ***, SS_HEARTBEAT_RESPONSE = ***, SS_HEARTBEAT_REQUEST = *** }*HeatBeat Protocol Command heartbeat service protocol > define all protocol messages in and out of heart beat thread that are pertinent to the functionality offered by heart beat thread.*

Detailed Description

Handle HeartBeat Module Start and Stop notifications.

API Header for HeartBeat messages used by senders and receivers.

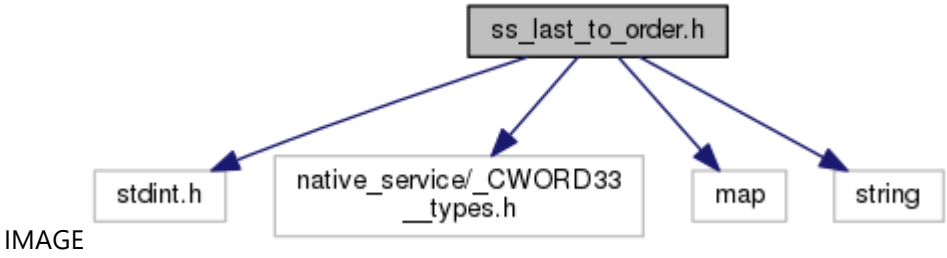
Declares the external Protocol messages to the HeartBeat.

ss_last_to_order.h File Reference

This file contains declaration of enum SS_CONT_CATEGORY_t and api convert lastinfo to order.

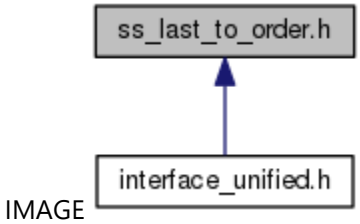
#include <stdint.h>
#include <native_service/_CWORD33__types.h>
#include <map>
#include <string>

Include dependency graph for ss_last_to_order.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Typedefs

typedef std::string **SS_CONTENT_NAME_t**
typedef std::map< SS_CONT_CATEGORY_t, SS_CONTENT_NAME_t > **SS_LAST_INFO_t**

Enumerations

enum **SS_CONT_CATEGORY_t** { **SS_CCAT_F_VIDEO**, **SS_CCAT_F_SUB_VIDEO**, **SS_CCAT_F_AUDIO**, **SS_CCAT_R_VIDEO**, **SS_CCAT_R_AUDIO**, **SS_CCAT_MAX** }

Functions

E_CWORD33_Status **SS_ConvLastInfoToOrder** (SS_LAST_INFO_t &curLastMode, std::string &order, const char *p_cfgPath=NULL)

Detailed Description

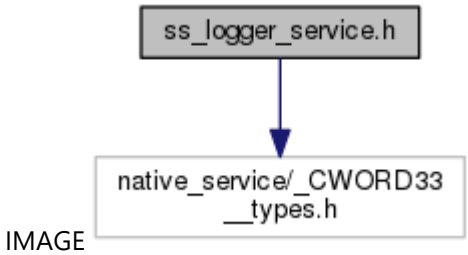
This file contains declaration of enum SS_CONT_CATEGORY_t and api convert lastinfo to order.

ss_logger_service.h File Reference

This file supports the the Logger Service.

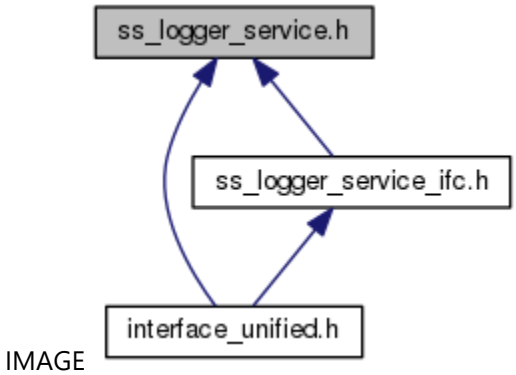
#include <native_service/_CWORD33__types.h>

Include dependency graph for ss_logger_service.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _TVINnumber

VIN numbers. struct _T_CWORD33_CANMileageInfo

CAN Diagnostic status data. More...

struct __CWORD62_DateAndTime

CAN current date and time. More...

struct _T_CWORD33_ScreenCaptureEvt

Screen Capture Event data. struct _SStatisticalCounter

Statistical counter. More...

struct _SEventLoggerCommonInfo

Event logger common information. More...

struct _SEventLoggerResetInfo

Event logger resets counter information. More...

struct _SEventLoggerEventInfo

Event logger CWORD56 events information. More...

struct _SEventCANLoggerEventInfo

Event logger CAN events information. *More...*

Macros

#define **MAX_STATISTICAL_BUFFER** 240

#define SHMEM_DRLOCATIONLOG "/DRLocationLog"

Shared Memory names.

#define SHMEM_DRINITIALLOG "/DRInitialLog"

Shared Memory names.

Typedefs

typedef struct _TVINnumber STVIN_NUMBER

VIN numbers.

typedef struct _T_CWORD33_CANMileageInfo STLOGGER_CANDIAGSTAT

CAN Diagnostic status data.

typedef struct __CWORD62_DateAndTime **STCanCurrentDateTime**

typedef enum _ELoggerDeviceTypes EDEV_TYPE

Logger Device Type.

typedef enum _ELoggerState ELOGGER_STAT

Logger State.

typedef enum _ELoggerErrorCodes **ELOGGERERR_CODES**

typedef struct _T_CWORD33_ScreenCaptureEvt STScreenCaptureEvt

Screen Capture Event data.

typedef enum _EEventLoggerUSBDeviceNumber EEvtLogUSBDevNumber

USB source number to store event logs.

typedef enum _EDevNumber EDevNumber

USB source number to store emergency error logs.

typedef enum _ECounterGroupID EStatCounterGroupID

Counter group ID.

typedef struct _SStatisticalCounter SStatisticalCounter

Statistical counter.

typedef enum _EEventLoggerSuccessCode **EEvtLoggerSuccessCode**

typedef enum _EEventLoggerErrorCode EEvtLoggerErrorCode

Event Logger error code.

typedef enum _EEmergencyLogErrorCode EELL_ErrorCode

Emergency Log error code.

typedef struct _SEventLoggerCommonInfo STEventLoggerCommonInfo

Event logger common information.

typedef struct _SEventLoggerResetInfo STEventLoggerResetInfo

Event logger resets counter information.

typedef struct _SEventLoggerEventInfo STEventLoggerEventInfo

Event logger CWORD56 events information.

typedef struct _SEventCANLoggerEventInfo STEventCANLoggerEventInfo

Event logger CAN events information.

Enumerations

enum _ELoggerDeviceTypes { **eDevUSB1**, **eDevUSB2**, **eDevSD**, **eEthernet**, **eTotalDevicesTypes**, **eInvalid_DevType** }*Logger Device Type.*

enum _ELoggerState { **eDeactivate**, **eActivate**, **eInvalid_LoggerState** }*Logger State.*

enum **_ELoggerErrorCodes** { **eSelectedDeviceNotFound** = ***, **eWriteToDeviceFailed** = ***, **eScreenCaptureFailed** = ***, **eScreenCaptureSaveTimeExpired** = ***, **eScreenCaptureStoreTimeExpired** = ***, **eNoLogToStore** = *** }

enum _EEventLoggerUSBDeviceNumber { **USB0** = ***, **USB1** = ***, **SD** = *** }*USB source number to store event logs.*

enum _EDevNumber { **eEEL_USB1** = ***, **eEEL_USB2** = ***, **eEEL_SD** = *** }*USB source number to store emergency error logs.*

enum _ECounterGroupID { **STARTUP_SHUTDOWN_COUNTER_GROUP** = ***, **_CWORD105__EVENTS_COUNTER_GROUP**, **APP_USAGE_COUNTER_GROUP**, **F_BLK_STABILITY_COUNTER_GROUP**, **MAX_COUNTER_GROUP** }*Counter group ID.*

enum **_EEventLoggerSuccessCode** { **COPY_EVT_USB_SUCCESS** = ***, **CLEAR_EVENT_LOG_SUCCESS** = ***, **STATISTICAL_COUNTER_READ_SUCCESS** = *** }

enum _EEventLoggerErrorCode { **USB_DEVICE_NOT_AVAILABLE** = ***, **USB_DEVICE_WRITE_ERROR** = ***, **CLEAR_EVENT_LOG_FAILED** = ***, **STATISTICAL_COUNTER_READ_FAILED** = ***, **NO_ERROR_INFO** = *** }*Event Logger error code.*

enum _EEmergencyLogErrorCode { **eDEVICE_NOT_AVAILABLE** = ***, **eDEVICE_WRITE_ERROR** = ***, **eNO_ERROR_INFO** = *** }*Emergency Log error code.*

Detailed Description

This file supports the the Logger Service.

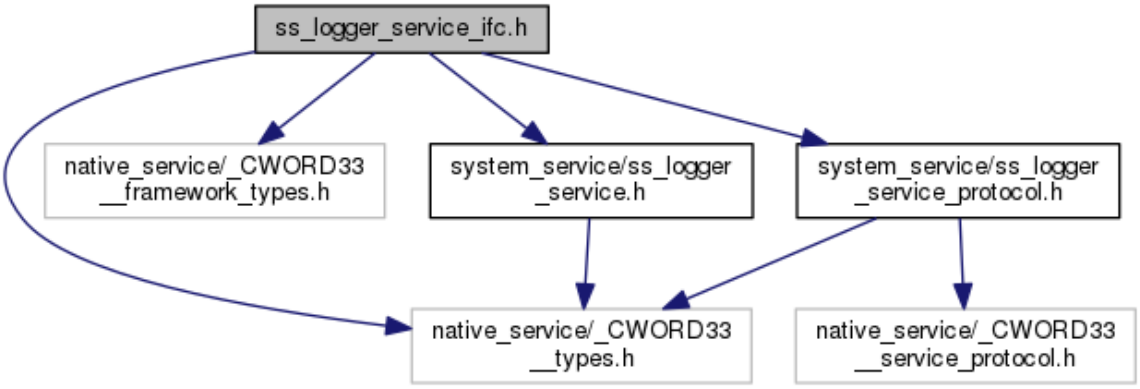
ss_logger_service_ifc.h File Reference

This file supports the the Logger Service.

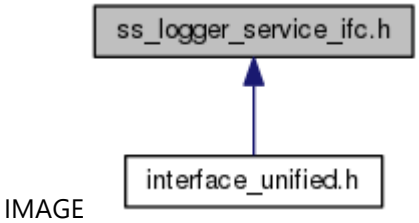
#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__framework_types.h>
#include "system_service/ss_logger_service.h"
#include "system_service/ss_logger_service_protocol.h"

Include dependency graph for ss_logger_service_ifc.h:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class LoggerServiceIf

logger_service

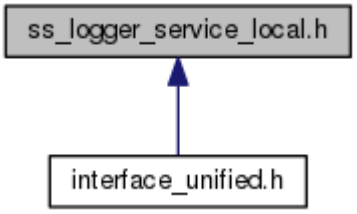
Detailed Description

This file supports the the Logger Service.

ss_logger_service_local.h File Reference

This file supports the the Logger Service.

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _T_CWORD33_SetLoggerParams

Screen Capture Event data. *More...*

struct _T_CWORD33_SetLoggerAllParams

struct _TErrorLogger_FolderInfo

union _uEvtLoggerCommonInfo

Event logger resets counter information. *More...*

struct _stLogEventss_

Logger Event information. *More...*

struct _T_CWORD33_EventLogPersistBuffer

Macros

#define MAX_EVT_RECORDS 20000

Macros definition for the Event Logs.

#define MAX_EVENTLOG_SIZE 10*MAX_EVT_RECORDS

Max size for event logging.

Typedefs

typedef struct _T_CWORD33_SetLoggerParams STLoggerSetParams

Screen Capture Event data.

typedef struct _T_CWORD33_SetLoggerAllParams **STLoggerSetAllParams**

typedef struct _TErrorLogger_FolderInfo **STLoggerFolderInfo**

typedef union _uEvtLoggerCommonInfo UEvtLoggerCommonInfo

Event logger resets counter information.

typedef struct _stLogEventss_ st_LogEvent_ss

Logger Event information.

typedef struct _T_CWORD33_EventLogPersistBuffer **STEventLogPersistBuffer**

Detailed Description

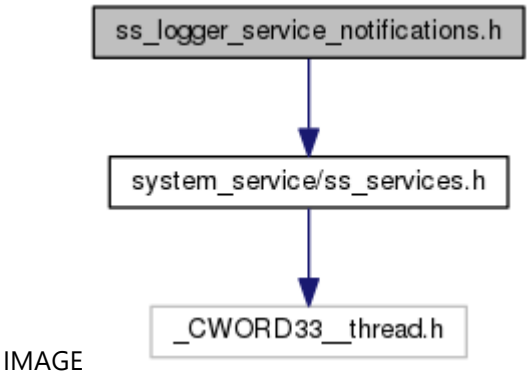
This file supports the the Logger Service.

ss_logger_service_notifications.h File Reference

This file supports the the Logger Service.

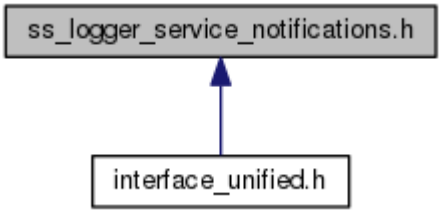
#include "system_service/ss_services.h"

Include dependency graph for ss_logger_service_notifications.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define NTFY_SS_LoggerService_Availability SERVICE_LOGGER"/Availability"

Logger Service Availability Notification.

Detailed Description

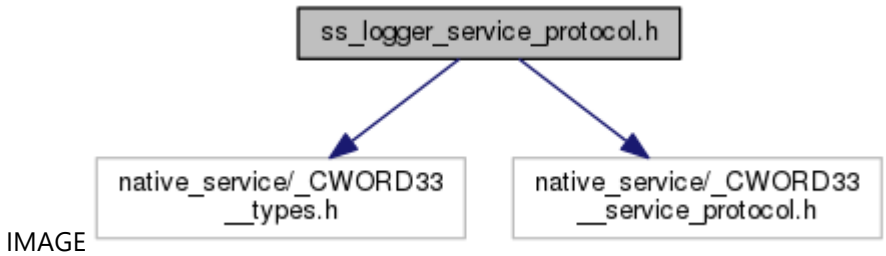
This file supports the the Logger Service.

ss_logger_service_protocol.h File Reference

This file supports the the Logger Service.

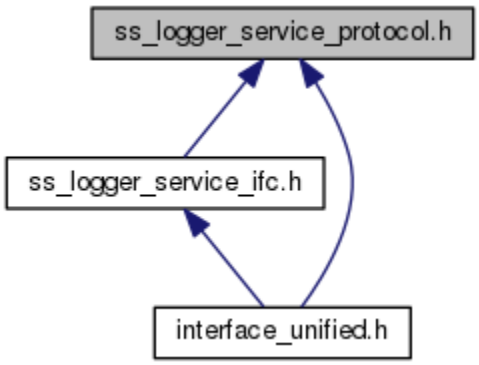
#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__service_protocol.h>

Include dependency graph for ss_logger_service_protocol.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Typedefs

typedef enum _LoggerServiceProtocol SS_LoggerServiceProtocol
Logger service event IDs.

typedef enum _SS_LOGGERSERVICEPROTOCOL SS_loggerserviceprotocol
Logger service event IDs.

typedef enum _LoggerServerEvents SS_LoggerServerEvents
Private events for the following categories of devices.

Enumerations

enum _LoggerServiceProtocol { **SS_LOGGER_MILEAGE_DATA** = ***, **SS_LOGGER_SET_PARAMS**, **SS_LOGGER_STORE_SCREENCAPTURE_AND_LOG**, **SS_LOGGER_STORE_SCREENCAPTURE**, **SS_LOGGER_STORE_LOG**, **SS_LOGGER_SCREENCAPTURE_EVT_ACK**, **SS_LOGGER_EVENT_COMMONINFO**, **SS_LOGGER__CWORD56__EVENT_INFO**, **SS_LOGGER__CWORD56__RESET_INFO**, **SS_LOGGER_SET_DATETIME**, **SS_LOGGER_SET_VIN**, **SS_LOGGER_UDP_LOGGING**,

SS_LOGGER_ERROR_EVENT_ARTIFACT_REQ,
SS_LOGGER_ERROR_EVENT_ARTIFACT_RESP,
SS_LOGGER__CWORD105_SHUTDOWN_COMPLETE,
SS_LOGGER_DIAGDTC_ACTIVE, **SS_LOGGER_SET_DIAGID**,
SS_LOGGER_ERROR_EVENT_TIMER_ID_LOGGING_START_RSPN,
SS_LOGGER_ERROR_EVENT_TIMER_ID_ARTIFACT_RESPONSE,
SS_LOGGER_ERROR_EVENT_TIMER_ID_SCREEN_CAPTURE_RSPN }*Logger service event IDs.*

enum _SS_LOGGERSERVICEPROTOCOL { **SS_LOGGERCOPYEVENTUSB** = ***,
SS_LOGGERCOPYEVENTUSB_SUCCESS_RESP,
SS_LOGGERCOPYEVENTUSB_ERROR_RESP, **SS_LOGGERCOPYEMERGENCYLOGS**,
SS_LOGGERCOPYEMERGENCYLOGS_SUCCESS_RESP,
SS_LOGGERCOPYEMERGENCYLOGS_ERROR_RESP, **SS_LOGGERCLEAREVENT**,
SS_LOGGERCLEAREVENT_SUCCESS_RESP, **SS_LOGGERCLEAREVENT_ERROR_RESP**,
SS_LOGGER_READ_STATL_COUNTER,
SS_LOGGER_READ_STATL_COUNTER_SUCCESS_RESP,
SS_LOGGER_READ_STATL_COUNTER_ERROR_RESP,
SS_LOGGER_RESET_STATL_COUNTER,
SS_LOGGER_RESET_STATL_COUNTER_SUCCESS_RESP,
SS_LOGGER_RESET_STATL_COUNTER_ERROR_RESP,
SS_LOGGER_ENG_READ_NUMOFEVENTS,
SS_LOGGER_ENG_READ_NUMOFEVENTS_RESP,
SS_LOGGER_ENG_READ_STATISTICAL_COUNTERS,
SS_LOGGER_ENG_READ_STATISTICAL_COUNTERS_RESP,
SS_LOGGER_UPLOAD_EVENTLOG, **SS_LOGGER_UPLOAD_EVENTLOG_RESP** }*Logger service event IDs.*

enum _LoggerServerEvents { **SS_LOGGER_SCREENCAPTURE_EVT** = ***,
SS_LOGGER_ERRORINFO_EVT, **SS_LOGGER_LOGINFO_EVT**,
SS_LOGGER_LOGSTARTED_EVT }*Private events for the following categories of devices.*

enum eSSLoggerCANProtocolID { **eSSLoggerCANProtocolIDCANTrigger** = ***,
eSSLoggerCANProtocolIDDTCTrigger }*eSSLoggerCANProtocolID*

enum eSSLoggerCANEvent { **eSSLoggerCANEventStart** = ***,
eSSLoggerCANEventError, eSSLoggerCANEventFinished }*eSSLoggerCANEvent*

Detailed Description

This file supports the the Logger Service.

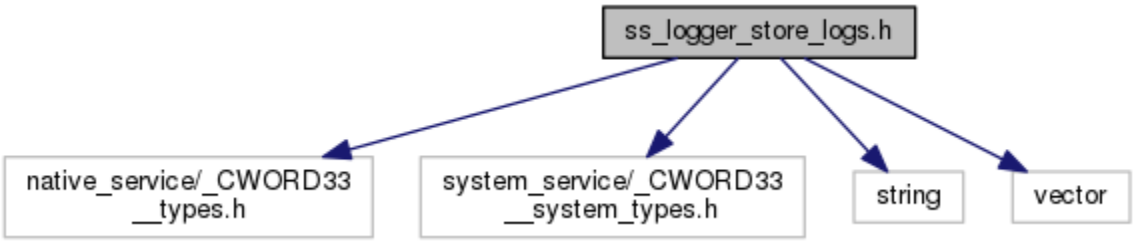
ss_logger_store_logs.h File Reference

This file supports the logger service interface for SystemManager.

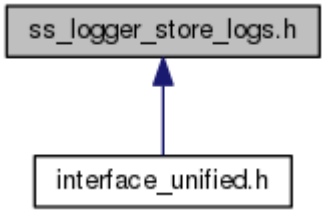
#include <native_service/_CWORD33__types.h>
#include "system_service/_CWORD33__system_types.h"
#include <string>
#include <vector>

Include dependency graph for ss_logger_store_logs.h:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define
SS_LOGGER_SAVE__CWORD33_LOG_FLAG "/ramd/log/_CWORD33_log/SS_LOGGER_SAVE__CWORD33_LOG_FLAG"

Enumerations

enum **SS_STORELOGS_OPE_TYPE** { **SS_STORELOGS__CWORD33_LOG** = ***, **SS_STORELOGS_ILLEGAL**, **SS_STORELOGS_ACCOFFON**, **SS_STORELOGS_SYS_ILLEGAL**, **SS_STORELOGS_ACCOFFON_PRESS**, **SS_STORELOGS_MAX** }

Functions

E_CWORD33_Status SS_LoggerStoreLogs (SS_STORELOGS_OPE_TYPE type)
SSLoggerSrvIfWriteDebugLogs.

E_CWORD33_Status **SS_LoggerStoreLogs_deleteOldLogAbnrm** (const std::string &log_path, std::vector< std::string > &l_vector, const std::string &f_archive_destination, SI_32 max_num, UI_32 abnrm_total)

E_CWORD33_Status **StartRtUsbLogThread** (HANDLE hApp)

E_CWORD33_Status **StopRtUsbLogThread** (HANDLE hApp)

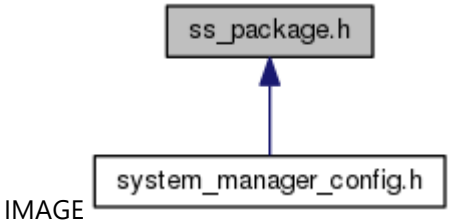
Detailed Description

This file supports the logger service interface for SystemManager.

ss_package.h File Reference

This file contains declaration of common package values.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **SS_PKG_SPI_LOADER** "SPI_LOADER"
#define **SS_PKG_UBOOT** "UBOOT"
#define **SS_PKG_SECURE_LOADER** "SECURE_LOADER"
#define **SS_PKG_SYS_UCOM** "SYS_UCOM"
#define **SS_PKG_CAN_UCOM** "CAN_UCOM"
#define **SS_PKG__CWORD52__UCOM** "_CWORD52__UCOM"
#define **SS_PKG__CWORD52__DATA** "_CWORD52__DATA"
#define **SS_PKG__CWORD52__FIXEQ** "_CWORD52__FIXEQ"
#define **SS_PKG_MAIN_EMMC** "MAIN_EMMC"
#define **SS_PKG_TARGETBOARD** "TARGETBOARD"
#define **SS_PKG_MAIN_PRODUCT_SI** "MAIN_PRODUCT_SI"
#define **SS_PKG_MAIN_COMP_SI** "MAIN_COMP_SI"
#define **SS_PKG_MAIN_LPF_SI** "MAIN_LPF_SI"
#define **SS_PKG_FACTORY_FROM** "FACTORY_FROM"
#define **SS_PKG_FACTORY_FROM2** "FACTORY_FROM2"
#define **SS_PKG_FACTORY_EEPROM** "FACTORY_EEPROM"
#define **SS_PKG_DECK** "DECK"
#define **SS_PKG_REAGION_CODE** "REAGION_CODE"
#define **SS_PKG__CWORD8__DB** "_CWORD8__DB"
#define **SS_PKG_CAPACITIVE_TP** "CAPACITIVE_TP"
#define **SS_PKG__CWORD52___CWORD32_** "_CWORD52___CWORD32_"
#define **SS_PKG_RADIO_PARAMETER** "RADIO_PARAMETER"
#define **SS_PKG_VR_DATA** "VR_DATA"
#define **SS_PKG_NAVI_GPS** "NAVI_GPS"
#define **SS_PKG_DTV_MAIN** "DTV_MAIN"
#define **SS_PKG_DTV_EEPROM** "DTV_EEPROM"
#define **SS_PKG_DTV_BOOT** "DTV_BOOT"
#define **SS_PKG_DTV_STATION_DB** "DTV_STATION_DB"
#define **SS_PKG_XM_TYPE_ID** "XM_TYPE_ID"
#define **SS_PKG_XM_HW** "XM_HW"
#define **SS_PKG_XM_SW** "XM_SW"
#define **SS_PKG_XM_SXI** "XM_SXI"

#define **SS_PKG_XM_BB** "XM_BB"
#define **SS_PKG_XM_HDEC** "XM_HDEC"
#define **SS_PKG_XM_RF** "XM_RF"
#define **SS_PKG_XM_SPL** "XM_SPL"
#define **SS_PKG_XM_AREA** "XM_AREA"
#define **SS_PKG_NAVI_PROGRAM** "NAVI_PROGRAM"
#define **SS_PKG_NAVI_MAP** "NAVI_MAP"
#define **SS_PKG_PHASE_INFO** "PHASE_INFO"
#define **SS_PKG_SERIES_INFO** "SERIES_INFO"
#define **SS_PKG_FPGA** "FPGA"
#define **SS_PKG_VP__CWORD63_** "VP__CWORD63_"
#define **SS_PKG_VP__CWORD31_** "VP__CWORD31_"
#define **SS_PKG_PREINSTALL_DATA** "PREINSTALL_DATA"
#define **MSG_PSM_REBOOT_NOTIFY** 0x11

Detailed Description

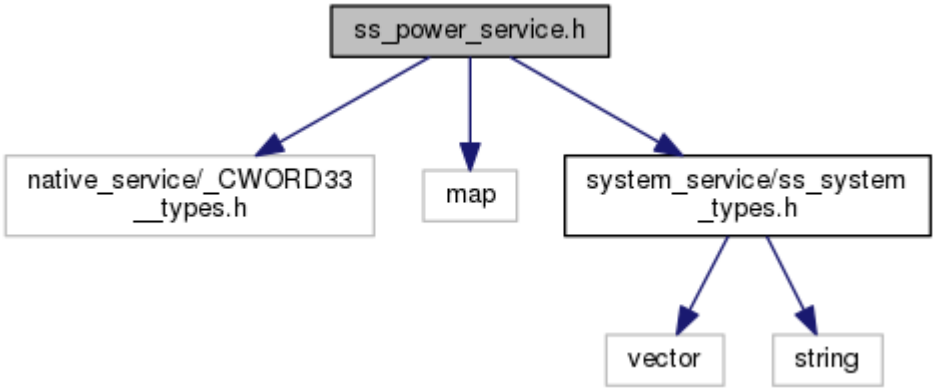
This file contains declaration of common package values.

ss_power_service.h File Reference

This file supports the Power Service interface.

#include <native_service/_CWORD33__types.h>
#include <map>
#include "system_service/ss_system_types.h"

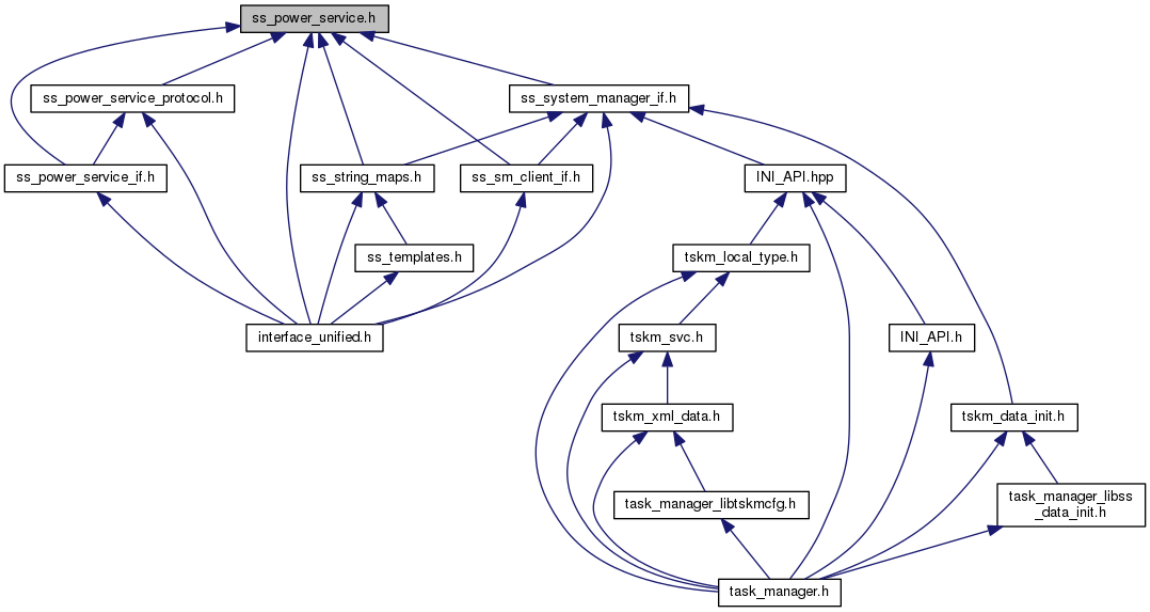
Include dependency graph for ss_power_service.h:



IMAGE

This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct _upinfo
struct _wakeinfo
struct _SystemModeInfo

struct StartupConfirmationMsgStrut
struct PowerRequestMsgStrut
struct PowerRequestMsgStrutWithUMCR
struct ShutdownRequestMsgStrut
struct EPWR_HB_REQ_MSG_STRUCT
struct SS_Pwr_CpuResetMsgStruct
struct Pwr_ServiceSetInterface
union Pwr_ServiceSetInterface::_data
struct Pwr_ServiceSetInterface::_data::_crank
struct Pwr_ServiceSetInterface::_data::_voltage
struct Pwr_ServiceSetInterface::_data::_ws_complete
struct Pwr_ServiceSetInterface::_data::_commwake
struct Pwr_ServiceSetInterface::_data::_UserMode
struct Pwr_ServiceSetInterface::_data::_ShutdownPopup
struct Pwr_ServiceSetInterface::_data::_PowerPopup

Macros

#define **PWR_RESET_MSG_STR_SIZE** 32

Typedefs

typedef enum _PwLevelTypes **PowerSrvLevelType**
typedef enum _PwLevelTypes **EPWR_LEVEL_TYPE**
typedef enum PowerSrvSessionType **EPWR_SESSION_TYPE**
typedef enum ePowerSrvCANStates **EPWR_CAN_STATE_TYPE**
typedef enum ePowerSrvWakeupLevels **EPWR_WAKEUP_LEVEL_TYPE**
typedef EPWR_WAKEUP_LEVEL_TYPE * **PEPWR_WAKEUP_LEVEL_TYPE**
typedef enum ePowerSrvPowerStates **EPWR_POWER_STATE_TYPE**
typedef enum ePowerSrvVoltageStates **EPWR_VOLTAGE_STATE_TYPE**
typedef EPWR_VOLTAGE_STATE_TYPE * **PEPWR_VOLTAGE_STATE_TYPE**
typedef enum ePowerSrvCrankStates **EPWR_CRANK_STATE_TYPE**
typedef enum ePowerSrvWakeupFactors **EPWR_WAKEUP_FACTOR_TYPE**
typedef EPWR_WAKEUP_FACTOR_TYPE * **PEPWR_WAKEUP_FACTOR_TYPE**
typedef enum ePwrServiceShutdownPopupType **EPWR_SHUTDOWN_POPUP_TYPE**
typedef enum ePwrServicePowerPopupType **EPWR_POWER_POPUP_TYPE**
typedef enum ePwrServiceLHCType **EPWR_LHC_TYPE**
typedef enum ePwrServiceProdModeType **EPWR_PROD_MODE_TYPE**
typedef enum ePwrServiceTransportModeType **EPWR_TRANSPORT_MODE_TYPE**
typedef enum ePwrServiceUserModeType **EPWR_USER_MODE_TYPE**
typedef enum ePwrServiceUserModeChangeReasonType
EPWR_USER_MODE_CHANGE_REASON_TYPE
typedef struct _upinfo **upInfo**
typedef struct _wakeinfo **wakeInfo**
typedef enum ePowerSrvSystemModeInfo **EPWR_SYSTEM_MODE_INFO**
typedef enum ePowerSrvStartupStageType **EPWR_STARTUP_STAGE_TYPE**
typedef struct _SystemModeInfo **SystemModeInfo**
typedef struct StartupConfirmationMsgStrut **EPWR_SC_MSG_STRUCT**
typedef enum ePowerSrv_CWORD102_StartupInfoType
EPWR__CWORD102__STARTUP_INFO_TYPE

typedef enum ePowerSrvPwrReqMsgCIDIOpeningType **EPWR_OPENING_TYPE**
typedef enum ePowerSrvPwrManResetInfoType **EPWR_MAN_RESET_INFO_TYPE**
typedef enum ePowerSrvTypeLastSysRetReq **EPWR_LAST_SYS_RET_REQ_TYPE**
typedef struct PowerRequestMsgStrut **EPWR_POWER_REQUEST_MSG_STRUCT**
typedef struct PowerRequestMsgStrutWithUMCR
EPWR_POWER_REQUEST_MSG_STRUCT_WITH_UMCR
typedef enum ePowerSrvONSType **EPWR_ONS_TYPE**
typedef enum ePowerSrvPwrShutdownTriggerType **EPWR_SHUTDOWN_TRIGGER_TYPE**
typedef struct ShutdownRequestMsgStrut **EPWR_SHUTDOWN_REQUEST_MSG_STRUCT**

Enumerations

enum **_PwLevelTypes** { **epspltUNKNOWN**, **epspltWAKEUP** = ***, **epspltNORMAL**, **epspltSHUTDOWN**, **epspltEMERGENCY** }
enum **PowerSrvSessionType** { **epsstUNKNOWN**, **epsstBASIC**, **epsstSUPERVISOR**, **epsstSYSTEM** }
enum **ePowerSrvCANStates** { **epscnINVALID** = ***, **epscnCANWAKEUP** = ***, **epscnCANSLEEP** = *** }
enum **ePowerSrvWakeupLevels** { **epswlINVALID** = ***, **epswlFULLRUN** = ***, **epswlLOCALWAKEUP** = *** }
enum **ePowerSrvPowerStates** { **epswsINVALID** = ***, **epswsPWRON** = ***, **epswsPWROFF** = *** }
enum **ePowerSrvVoltageStates** { **epsvsINVALID** = ***, **epsvsNORMAL** = ***, **epsvsLVI1** = ***, **epsvsLVI2** = *** }
enum **ePowerSrvCrankStates** { **epscsINVALID** = ***, **epscsENTRY** = ***, **epscsEXIT** = *** }
enum **ePowerSrvWakeupFactors** { **epswfINVALID** = ***, **epswfTESTACC** = ***, **epswfON_KEY** = ***, **epswfIGN_ACC** = ***, **epswfDR_OPEN_CLOSE** = ***, **epswfDX_ACTIVATION** = ***, **epswfETHERNET** = ***, **epswfPASS_ACTIVATION** = ***, **epswfSPVACTIVATION** = ***, **epswfUSER_DATA_RESET** = *** }
enum **ePwrServiceShutdownPopupType** { **epsspPowerSave1** = ***, **epsspPowerSave2** = ***, **epsspPowerSave3** = ***, **epsspPowerSaveClr** = ***, **epsspLowVoltage1** = ***, **epsspLowVoltage2** = ***, **epsspLowVoltage3** = ***, **epsspLowVoltageClr** = ***, **epsspBattCouplingSW1** = ***, **epsspBattCouplingSW2** = ***, **epsspBattCouplingSW3** = ***, **epsspBattCouplingSWClr** = ***, **epsspAbnormalTemp_1st** = ***, **epsspAbnormalTemp_Clr** = ***, **epsspLimpHome_1st** = ***, **epsspLimpHome_2nd** = ***, **epsspLimpHome_3rd** = ***, **epsspLimpHome_Clr** = ***, **epsspProdMd_1st** = ***, **epsspProdMd_Clr** = ***, **epsspTransMd_1st** = ***, **epsspTransMd_Clr** = ***, **epsspAllClr** = *** }
enum **ePwrServicePowerPopupType** { **epssppNormal** = ***, **epssppCritical** = ***, **epssppAppCritical** = ***, **epssppAllClr** = *** }
enum **ePwrServiceLHCType** { **epslhcINVALID** = ***, **epslhcDISABLED** = ***, **epslhcENABLED** = *** }
enum **ePwrServiceProdModeType** { **epspmINVALID** = ***, **epspmDISABLED** = ***, **epspmENABLED** = *** }
enum **ePwrServiceTransportModeType** { **epstmINVALID** = ***, **epstmDISABLED** = ***, **epstmENABLED** = *** }
enum **ePwrServiceUserModeType** { **epsumINVALID** = ***, **epsumOFF** = ***, **epsumON** = *** }

enum ePwrServiceUserModeChangeReasonType { epsumcrNOT_AVAILABLE = ***,
epsumcrON_KEY = ***, epsumcrPARKING_B, epsumcrPRE_BA, epsumcrNORMAL,
epsumcrBACKGROUND_BA }
enum **ePowerSrvSystemModeInfo** { **epssinfINVALID** = ***, **epssinfNORMAL** = ***,
epssinfPROGRAMMING = *** }
enum **ePowerSrvStartupStageType** { **epssusfINVALID** = ***,
epssusSYSTEM_SERVICES_STARTED = ***, **epssusALL_SERVICES_LAUNCHED** = *** }
enum **EPWR_SC__CWORD56__BOOT_MODE_TYPE** { **eps_CWORD56_bmINVALID** = ***,
eps_CWORD56_bmAPPLICATION_MODE = ***,
eps_CWORD56_bmPROGRAMMING_MODE = *** }
enum **EPWR_SC_WAKEUP_TYPE** { **epsstINVALID** = ***, **epsstWARMSTART** = ***,
epsstCOLDSTART = *** }
enum **EPWR_SC_COLD_START_REQ_TYPE** { **epsscrtINVALID** = ***, **epsscrtNOT_REQUIRED** =
***, **epsscrtREQUIRED** = *** }
enum **EPWR_SC_SECURITY_STATUS** { **epsssINVALID** = ***, **epsssUNLOCK** = ***, **epsssLOCK** =
*** }
enum ePowerSrv_CWORD102_StartupInfoType { **epsprm_CWORD102_siINVALID** = ***,
epsprm_CWORD102_si_CWORD102__STARTUP_NORMAL = ***,
epsprm_CWORD102_si_CWORD102__STARTUP_AFTER_RESET = *** }
enum ePowerSrvPwrReqMsgCIDIOpeningType { **epsprmotINVALID** = ***,
epsprmotTIMING_TYPE_1 = ***, epsprmotTIMING_TYPE_2 = *** }
enum ePowerSrvPwrManResetInfoType { **epsprmriINVALID** = ***, **epsprmriNORMAL** = ***,
epsprmriWAKEUP_AFTER_RESET = *** }
enum ePowerSrvTypeLastSysRetReq { **epsprlsrrINVALID** = ***, **epsprlsrrLAST_SYSTEM_OFF** =
***, **epsprlsrrLAST_SYSTEM_ON** = *** }
enum **ePowerSrvONSType** { **epssdmonsINVALID** = ***, **epssdmonsNO_ONS** = ***,
epssdmonsONS = *** }
enum **ePowerSrvPwrShutdownTriggerType** { **epssdmsdtINVALID** = ***,
epssdmsdtTESTACC_OFF = ***, **epssdmsdtON_KEY** = ***, **epssdmsdtIGN_LOCK** = ***,
epssdmsdtPWR_SAVE = ***, **epssdmsdtTMP_STARTUP** = ***,
epssdmsdtDIAG_DEACTIVATION = ***, **epssdmsdtABNORMAL_VOLTAGE** = ***,
epssdmsdtABNORMAL_TEMP = ***, **epssdmsdtBATTERYCUTOFF** = ***,
epssdmsdtLIMPHOME = ***, **epssdmsdtHU_CAN_ERROR** = ***,
epssdmsdtBODY_CAN_ERROR = ***, **epssdmsdtTRANSPORT_MODE** = ***,
epssdmsdtPRODUCTION_MODE = ***, **epssdmsdtIGN_OFF** = ***,
epssdmsdtGENERIC_ERROR_RESET = ***, **epssdmsdtFATAL_ERROR_RESET** = ***,
epssdmsdtUSER_DATA_RESET = ***, **epssdmsdtFACTORY_DATA_RESET** = ***,
epssdmsdtFAST_SLEEP_MODE = ***, **epssdmsdtNORMAL_RESET** = ***,
epssdmsdtPROGUPDATE_RESET = *** }
enum **epsCpuResetReason** { **epsCpuResetReasonGeneric** = ***, **epsCpuResetReasonFatalError**,
epsCpuResetReasonUserDataReset, **epsCpuResetReasonFactoryDataReset**,
epsCpuResetReasonUserForceReset, **epsCpuResetReasonShipmentModeReset**,
epsCpuResetReasonHeadUnitReset, **epsCpuResetReasonNormalReset**,
epsCpuResetReasonProgupdateReset }

Detailed Description

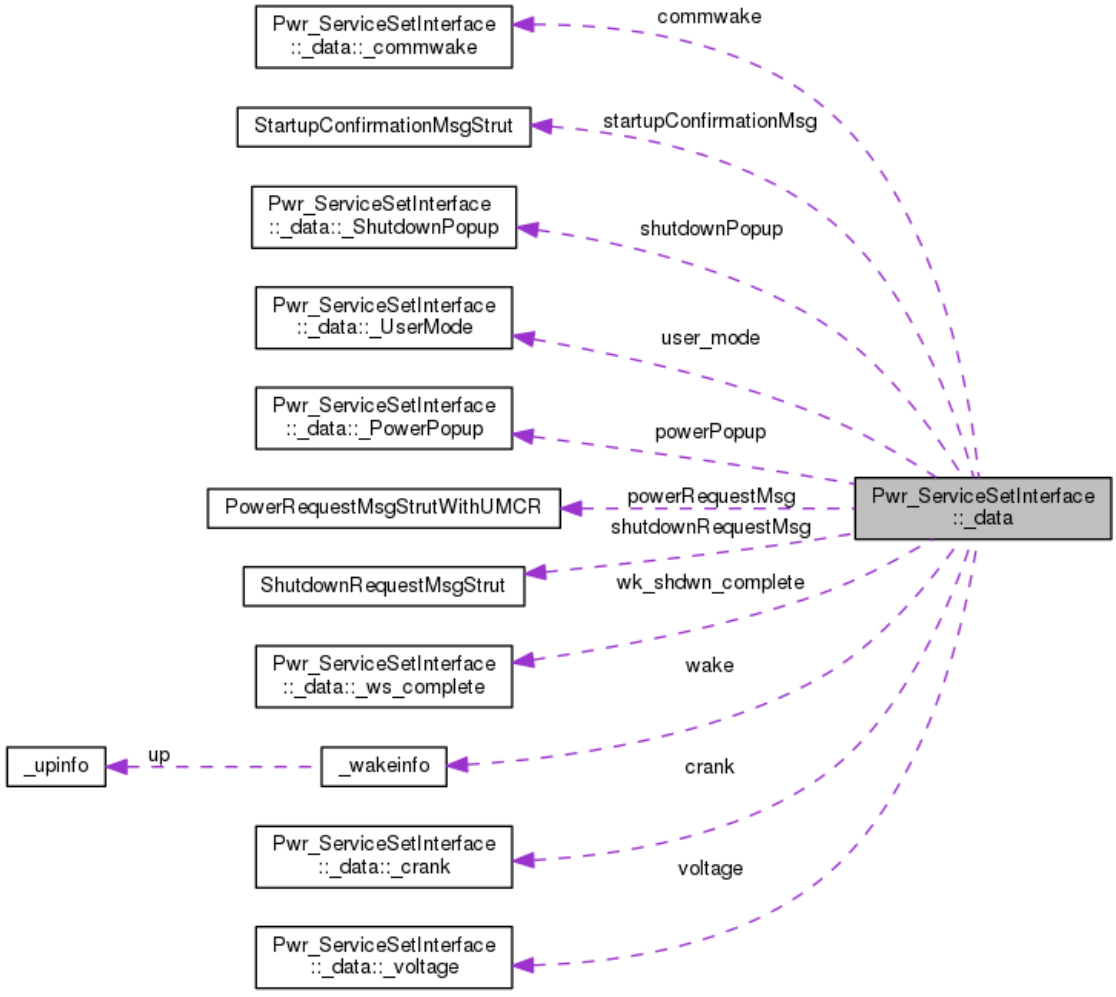
This file supports the Power Service interface.

Class Documentation

union Pwr_ServiceSetInterface::_data

Collaboration diagram for Pwr_ServiceSetInterface::_data:

IMAGE



Class Members:

struct _commwake	commwake	
struct _crank	crank	
struct _PowerPopup	powerPopup	
PowerRequestMsgStrutWithUMCR	powerRequestMsg	
struct _ShutdownPopup	shutdownPopup	
ShutdownRequestMsgStrut	shutdownRequestMsg	
StartupConfirmationMsgStrut	startupConfirmationMsg	
struct _UserMode	user_mode	

struct _voltage	voltage	
wakeInfo	wake	
struct _ws_complete	wk_shdwn_complete	

struct Pwr_ServiceSetInterface::_data::_crank

Class Members:

EPWR_CRANK_STATE_TYPE	state	

struct Pwr_ServiceSetInterface::_data::_voltage

Class Members:

EPWR_VOLTAGE_STATE_TYPE	state	

struct Pwr_ServiceSetInterface::_data::_ws_complete

Class Members:

EPWR_WAKEUP_FACTOR_TYPE	factor	
EPWR_WAKEUP_LEVEL_TYPE	level	

struct Pwr_ServiceSetInterface::_data::_commwake

Class Members:

EPWR_CAN_STATE_TYPE	state	

struct Pwr_ServiceSetInterface::_data::_UserMode

Class Members:

EPWR_USER_MODE_TYPE	mode	

struct Pwr_ServiceSetInterface::_data::_ShutdownPopup

Class Members:

EPWR_SHUTDOWN_POPUP_TYPE	shutdownPopupEvent	

struct Pwr_ServiceSetInterface::_data::_PowerPopup

Class Members:

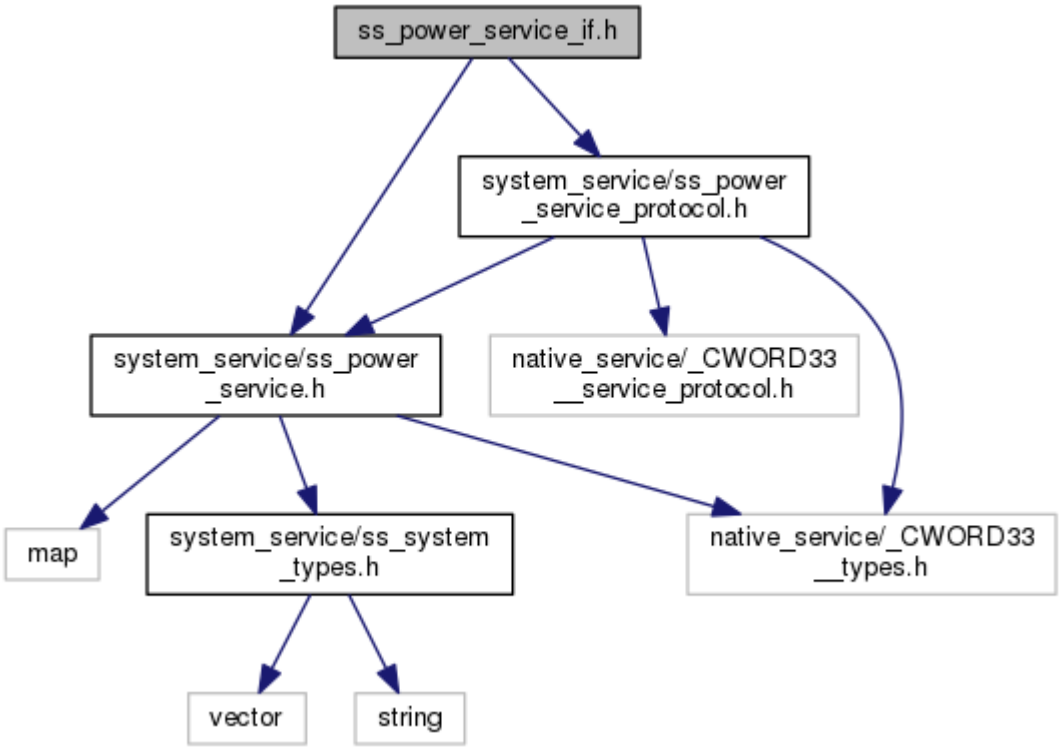
EPWR_POWER_POPUP_TYPE	powerPopupEvent	

ss_power_service_if.h File Reference

This file supports the Power Service client interface.

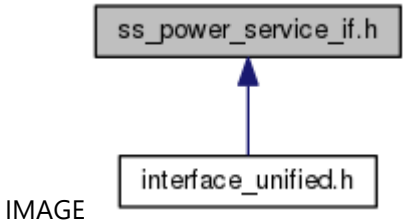
#include "system_service/ss_power_service.h"
#include "system_service/ss_power_service_protocol.h"

Include dependency graph for ss_power_service_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

HANDLE OpenPowerService (HANDLE f_hApp)
E_CWORD33_Status ClosePowerService (HANDLE f_hApp, HANDLE f_hService)
E_CWORD33_Status PwrServiceOpenSessionRequest (HANDLE f_hService, EPWR_SESSION_TYPE f_eSessionType)
E_CWORD33_Status PwrServiceCloseSessionRequest (HANDLE f_hService, HANDLE f_hSession)

E_CWORD33_Status PwrServiceDecodeOpenSessionResponse (HANDLE f_hApp, HANDLE &f_hSession)
E_CWORD33_Status PwrServiceSetVoltageState (HANDLE f_hSession, EPWR_VOLTAGE_STATE_TYPE f_eVoltage_state)
E_CWORD33_Status PwrServiceSetCrankState (HANDLE f_hSession, EPWR_CRANK_STATE_TYPE f_eCrank_state)
E_CWORD33_Status PwrServiceSystemModeInfoRequest (HANDLE f_hSession)
E_CWORD33_Status PwrServiceSendInitCompReport (HANDLE f_hSession)
E_CWORD33_Status PwrServiceSendSetShutdownPopupRequest (HANDLE f_hSession, EPWR_SHUTDOWN_POPUP_TYPE f_eShutdownPopup)
E_CWORD33_Status PwrServiceSendStartupConfirmationMsgRequest (HANDLE f_hSession, EPWR_SC_MSG_STRUCT f_eState)
E_CWORD33_Status PwrServiceSendPowerRequest (HANDLE f_hSession, EPWR_POWER_REQUEST_MSG_STRUCT_WITH_UMCR &f_eState)
E_CWORD33_Status PwrServiceSendShutdownRequest (HANDLE f_hSession, EPWR_SHUTDOWN_REQUEST_MSG_STRUCT &f_eState)
E_CWORD33_Status PwrServiceSendHeartBeatRequest (HANDLE f_hSession, EPWR_HB_REQ_MSG_STRUCT *f_eHbReqMsg)

Detailed Description

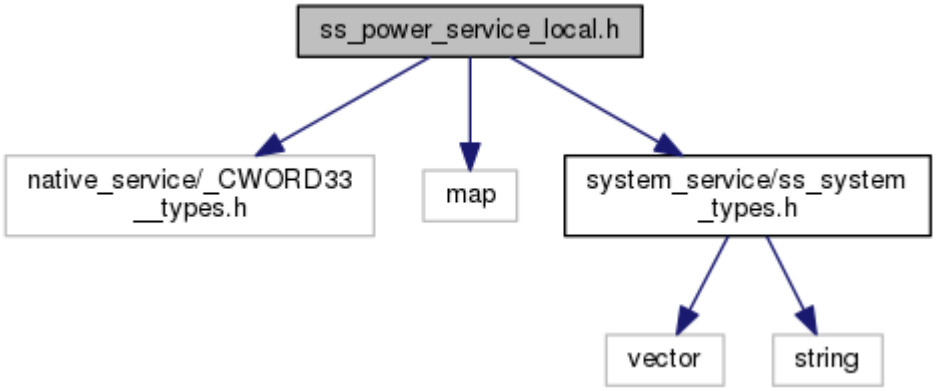
This file supports the Power Service client interface.

ss_power_service_local.h File Reference

This file supports the Power Service interface.

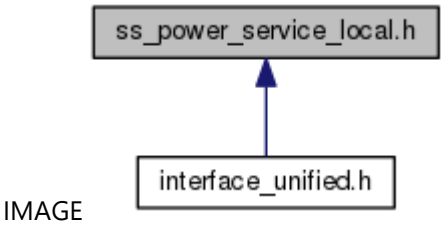
#include <native_service/_CWORD33__types.h>
#include <map>
#include "system_service/ss_system_types.h"

Include dependency graph for ss_power_service_local.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _PwCompleteAck
struct _SS_PSCurrentState

Typedefs

typedef enum _PwLVIStatus **PowerSrvLVIStatus**
typedef enum _PwLVIStatus **EPWER_LVI_STATUS**
typedef struct _PwCompleteAck **StartCompleteAck**
typedef struct _SS_PSCurrentState **SS_PSCurrentState**

Enumerations

enum **_PwLVIStatus** { **ePwSrvLVI_Status_InActive**, **ePwSrvLVI_Status_Active** }

Variables

const UI_32 **MaxTestCaseName** = ***

const UI_32 **MaxRespMsg** = ***

Detailed Description

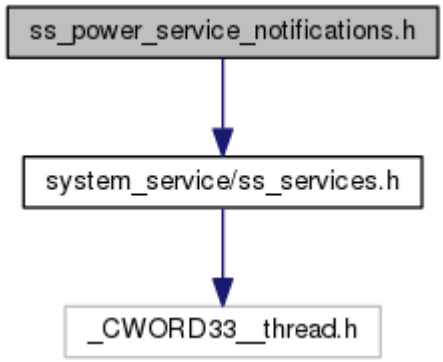
This file supports the Power Service interface.

ss_power_service_notifications.h File Reference

This file supports the Power Service interface.

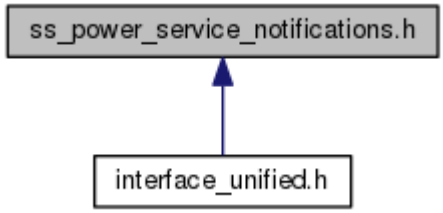
#include "system_service/ss_services.h"

Include dependency graph for ss_power_service_notifications.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define szNTFY_PowerAvailability SERVICE_POWER"/Availability"

Indicated if Power Service is available or not portion of this message is BOOL.

Detailed Description

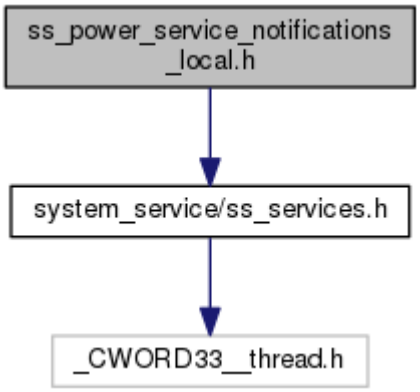
This file supports the Power Service interface.

ss_power_service_notifications_local.h File Reference

This file supports the Power Service interface.

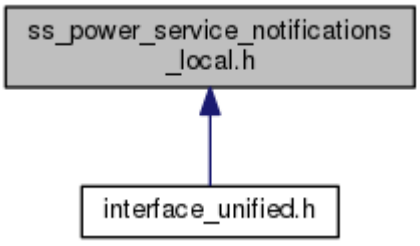
#include "system_service/ss_services.h"

Include dependency graph for ss_power_service_notifications_local.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define szNTFY_PowerLVI1 SERVICE_POWER"/LVI1"

Depreciated !!

#define szNTFY_PowerLVI2 SERVICE_POWER"/LVI2"

Depreciated !!

#define szNTFY_PowerLevel SERVICE_POWER"/Level"

Indicates the current power level state.

#define szNTFY_ShutdownPopup SERVICE_POWER"/ShutdownPopup"

#define szNTFY_PowerPopup SERVICE_POWER"/PowerPopup"

Detailed Description

This file supports the Power Service interface.

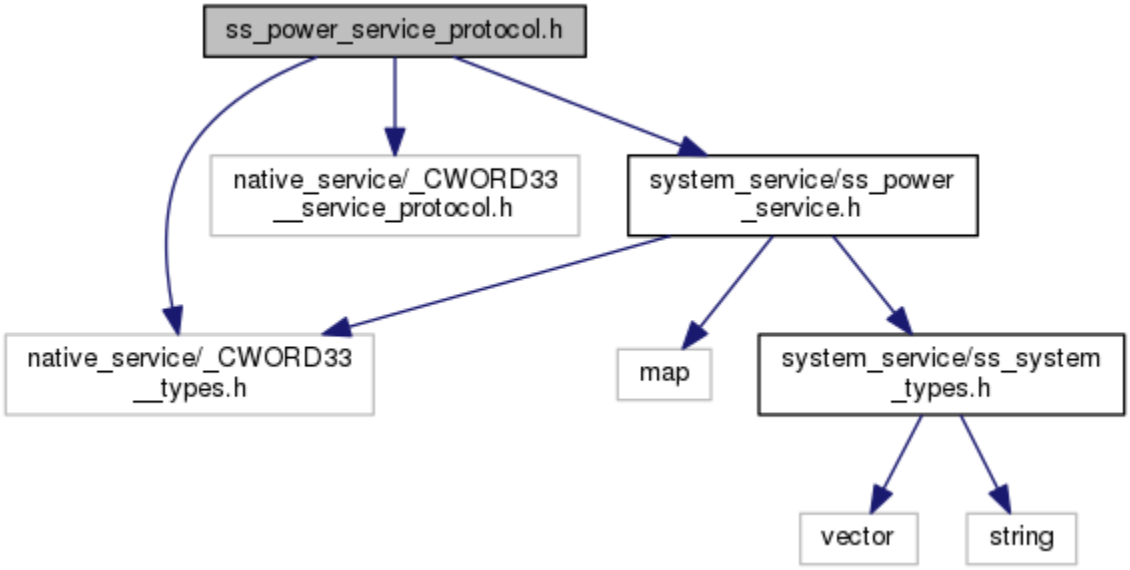
ss_power_service_protocol.h File Reference

This file supports the Power Service interface.

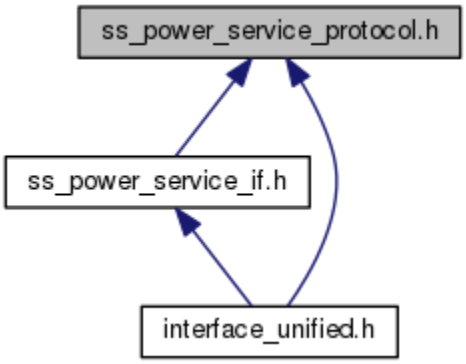
#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__service_protocol.h>
#include "system_service/ss_power_service.h"

Include dependency graph for ss_power_service_protocol.h:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Typedefs

typedef enum _SS_PowerServiceProtocol SS_PowerServiceProtocol

Enumerations

enum _SS_PowerServiceProtocol { SS_POWER_PRINT_CONNECTIONS = ***,
SS_POWER_PRINT_STACK = ***, SS_POWER_WAKEUP_COMPLETE = ***,
SS_POWER_SHUTDOWN_COMPLETE = ***, SS_POWER_VOLTAGE_STATE = ***,
SS_POWER_LVI2_SHUTDOWN_COMPLETE = ***, SS_POWER_CRANK_STATE = ***,
SS_POWER_WAKEUP_MODULES_CMPL_RSPN = ***, SS_POWER_SYSTEM_LAUNCH_COMPLETE = ***, SS_POWER_STATE_CHANGE_REQ = ***, SS_POWER_STATE_CHANGE_RESP = ***,
SS_POWER_COMM_WAKEUP = ***, SS_POWER_SHUTDOWN_REQ = ***,
SS_POWER_SHUTDOWN_RESP = ***, SS_POWER_CRNT_STATE_QUERY = ***,
SS_POWER_CRNT_STATE_QUERY_RSPN = ***, SS_POWER_SYSTEM_MODE_INFO_REQ = ***,
SS_POWER_SYSTEM_MODE_INFO_RESP = ***, SS_POWER_INITCOMP_REP = ***,
SS_POWER_PUBLISH_POWER_POPUP_REQ, SS_POWER_PUBLISH_POWER_POPUP_RESP,
SS_POWER_PUBLISH_SHUTDOWN_CONDITION_REQ,
SS_POWER_PUBLISH_SHUTDOWN_CONDITION_RESP,
SS_POWER_FWD_START_CONFIRMATION_MSG_REQ,
SS_POWER_FWD_START_CONFIRMATION_MSG_RESP, SS_POWER_POWER_REQUEST_MSG,
SS_POWER_SHUTDOWN_REQUEST_MSG, SS_POWER_USER_MODE_SET_RESP,
SS_POWER_HEARTBEAT_REQ, SS_POWER_HEARTBEAT_RESP, SS_POWER_HARD_RESET_REQ }

Detailed Description

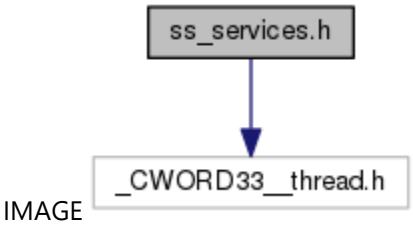
This file supports the Power Service interface.

ss_services.h File Reference

This file supports contains the queue names of all System Services processes.

#include <_CWORD33__thread.h>

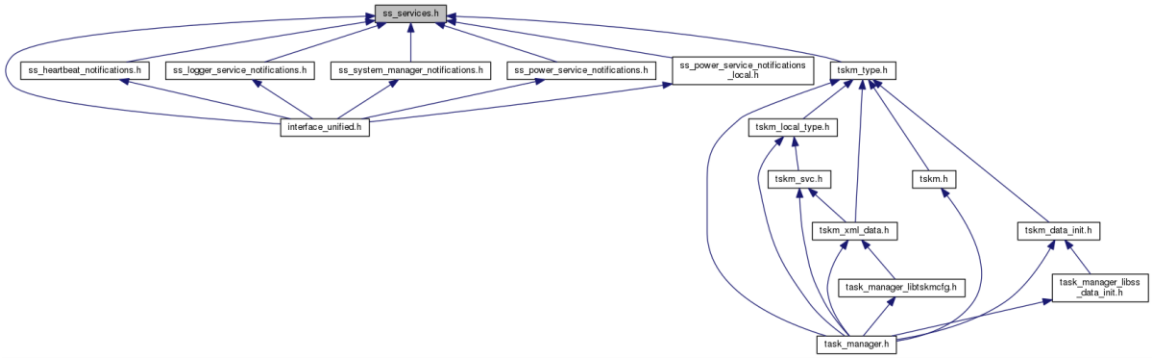
Include dependency graph for ss_services.h:



IMAGE

This graph shows which files directly or indirectly include this file:

IMAGE



Macros

#define **SS_DEV_DETECT_THD** "SS_Dev_Detect_Thd"
#define **SS_DEV_DETECT_SRV** MN_SS_DEVDETECTSRV
#define **SS_DEV_DETECT_BSP_THD** "SS_DeviceDetectErr"
#define **SS_GROUP_LAUNCH_TRIGGER** "SS_GroupLaunchTrigger"
#define **SERVICE_HEARTBEAT** "SS_HeartBeatService"
#define **SERVICE_LOGGER** MN_SS_LOGGERSRV
#define **SS_UDEV_DEV_DETECT_DLL** "SS_UDEV_DeviceDetect"
#define **SS_PLM_SERVICE** "SS_PLMService"
#define **SERVICE_POWER** MN_SS_POWERSERVICE
#define **SS_RESOURCE_MONITOR_SERVICE** "SS_ResourceMonitorService"
#define **SERVICE_SOFTWAREUPDATE** "SS_SofUpdateSrv"
#define **SERVICE_SYSMANAGER** MN_SS_SYSMANAGER
#define **SS_WINSYS** MN_SS_WINSYS
#define **SS_RESOURCE_MANAGER** MN_SS_RESOURCEMGR
#define **SS_TASK_MANAGER** MN_SS_TASKMANAGER
#define **SERVICE_VUP** MN_VUPSERVICE
#define **SS_UPDATESERVICE** MN_SS_UPDATESERVICE

Detailed Description

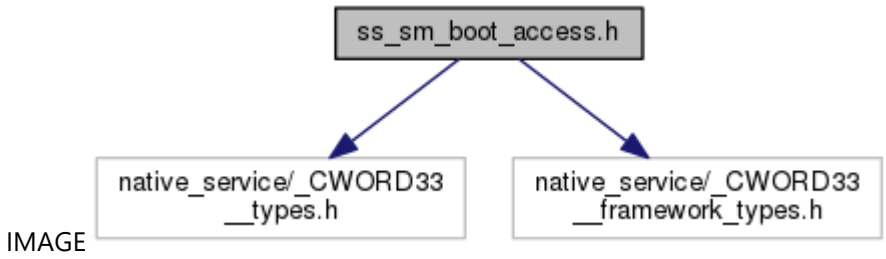
This file supports contains the queue names of all System Services processes.

ss_sm_boot_access.h File Reference

This file provides API for get boot information from nor flash.

#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__framework_types.h>

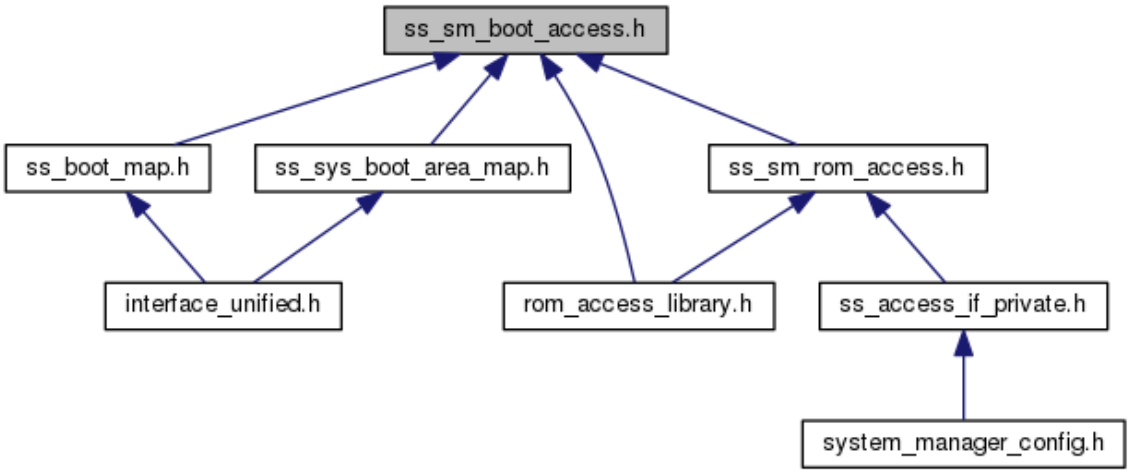
Include dependency graph for ss_sm_boot_access.h:



IMAGE

This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct LBM_boot_wpinfo_t
struct LBM_boot_t
struct _tmb_ram_t
class BOOT_AccessIf

***BOOT_AccessIf*. Macros**

#define **LBM_UINT32** unsigned int
#define **LBM_UINT16** uint16_t
#define **LBM_UINT8** unsigned char
#define **LBM_INT64** int64_t
#define **LBM_INT32** unsigned int
#define **LBM_INT16** uint16_t

```
#define LBM_INT8  unsigned char
#define LBM_NOR_MIRROR_SIZE  0x10000
#define LBM_DRAM_SIZE  0x10000
#define LBM_NOR_BOOT_SIZE  128
#define LBM_NOR_TSKM_SIZE  128
#define LBM_NOR_VUP_SIZE  1024
#define LBM_NOR_ANA_SIZE  32
#define LBM_NOR_VEHI_SIZE  32
#define LBM_NOR_DEVMGR_SIZE  32
#define SYSUP_DRAM_NORMAL  (LBM_UINT32)0x4E524E52  /* normal value
    NRNR*/
#define SYSUP_DRAM_SFTERRSR  (LBM_UINT32)0x73667366  /* abnormal Soft
    Reset(softwareabnormality) sfsf*/
#define SYSUP_DRAM_DISCARD  (LBM_UINT32)0x44434443  /* need to discard backup
    DCDC*/
#define SYSUP_CND_FACTRESET  (LBM_UINT16)0x434c  /* COLD START */
#define SYSUP_CND_NORMAL  (LBM_UINT16)0x4853  /* HOT START  */
#define SYSUP_MODE_NORMAL  (LBM_UINT32)0x4E524E52  /* normal mode           */
#define SYSUP_MODE_VERUP  (LBM_UINT32)0x56555655  /* versionup mode        */
#define SUBCPU_STS_NOMAL  (LBM_UINT32)0x00000000  /* Communication normal    */
#define SUBCPU_STS_COMNG  (LBM_UINT32)0x88888888  /* Communication abnormal  */
#define SDRAM_STS_NOMAL  (LBM_UINT32)0x11111111  /* backup power supply is normal  */
#define SDRAM_STS_BUPNG  (LBM_UINT32)0x99999999  /* backup power supply is abnormal */
#define HWDT_STS_NOMAL  (LBM_UINT32)0x22222222  /* WDT normal                */
#define HWDT_STS_WDTTO  (LBM_UINT32)0xAAAAAAAA  /* WDT time-out outbreak      */
#define SWDT_STS_NOMAL  HWDT_STS_NOMAL        /* SWDT normal               */
#define SWDT_STS_WDTTO  HWDT_STS_WDTTO        /* SWDT time-out outbreak     */
#define IFS_PROG1_CODE  (LBM_UINT8)0xAA  /* Side PROG1            */
#define IFS_PROG2_CODE  (LBM_UINT8)0x55  /* Side PROG2            */
#define UPTBLE_ID_MAX  (LBM_UINT8)0x20  /* LBM_UPTBLINFO Maximum value */
#define UPTBLE_ID_NORBOOT  (LBM_UINT8)0x00  /* NOR-BOOT ID              */
#define UPTBLE_ID_SCRLDR  (LBM_UINT8)0x01  /* SecureLoader ID          */
#define UPTBLE_ID_NANDBOOT  (LBM_UINT8)0x02  /* NAND-BOOT ID             */
#define UPTBLE_ID_ROOTFS  (LBM_UINT8)0x03  /* Rootfs1 ID               */
#define UPTBLE_ID_ROOTF_CWORD98_  (LBM_UINT8)0x04  /* Rootf_CWORD98_ ID               */
#define STUP_STS_NORMAL  (LBM_UINT32)0x00000000  /* normal status  */
#define STUP_STS_ERROR_DETEC  (LBM_UINT32)0x22222222
#define STUP_STS_UPDATING  (LBM_UINT32)0x33333333  /* updating status */
#define STUP_STS_SIGNVRFY_ERROR  (LBM_UINT32)0x11111111
#define UPDATE_NOTI_NONE  (LBM_UINT32)0x00000000 /* Notify is NOT required. */
#define UPDATE_NOTI_EXIST  (LBM_UINT32)0xEEEEEEEE /* Notify is required.    */
#define RAMJUDGE_STS_NOMAL  (LBM_UINT32)0x44444444 /* RAM Judge Port is High */
#define RAMJUDGE_STS_NG  (LBM_UINT32)0xDDDDDDDD /* RAM Judge Port is Low  */
#define SELF_REFRESH_OK  (LBM_UINT32)0x55555555 /* DRAM Self Refresh is OK */
#define SELF_REFRESH_NG  (LBM_UINT32)0xCCCCCCCC /* DRAM Self Refresh is NG */
#define PRODUCTROM  (LBM_UINT32)0x11111111
#define RELEASEROM  (LBM_UINT32)0x22222222
#define RELEASEROM_REMAIN_USBBOOT  (LBM_UINT32)0x44444444
```

#define **DEBUGROM** (LBM_UINT32)0x88888888
#define **WRITE_NONE** (LBM_UINT32)0x00000000
#define **WRITE_DONE** (LBM_UINT32)0x11111111
#define **DEFAULT_OPIMAGE_DRAM_BACKUPED** (LBM_UINT32)0x00000000
#define **DEFAULT_OPIMAGE_FROM_EMMC** (LBM_UINT32)0xAAAAAAAA
#define **SPECIFIC_OPIMAGE_DRAM_BACKUPED** (LBM_UINT32)0x11111111
#define **SPECIFIC_OPIMAGE_FROM_EMMC** (LBM_UINT32)0xBBBBBBBB
#define **SPECIFIC_OPIMAGE_FLAG_NONE** (LBM_UINT32)0x00000000
#define **SPECIFIC_OPIMAGE_FILE_NONE** (LBM_UINT32)0x11111111
#define **SPECIFIC_OPIMAGE_FILESIZE_0** (LBM_UINT32)0x22222222
#define **SPECIFIC_OPIMAGE_FILE_EXIST** (LBM_UINT32)0xEEEEEEEE
#define **SS_SYS_AREA_BOOT_MAX_SIZE** (0x00010000UL)
#define **SS_SYS_AREA_ROM_OFFSET** (0x00010000UL)
#define **SS_SYS_AREA_ROM_MAX_SIZE** (0x00001000UL)
#define **SS_SYS_AREA_RAM_OFFSET** (0x00011000UL)
#define **SS_SYS_AREA_RAM_MAX_SIZE** (0x00001000UL)

Typedefs

typedef struct LBM_boot_wpinfo_t **LBM_UPTBLINFO**
typedef struct LBM_boot_t **LBM_NOR_t**
typedef struct _tmb_ram_t **LBM_RAM_t**

Enumerations

enum BAI_OPEN_t { **BAI_OPEN_RO**, **BAI_OPEN_RW** }

Detailed Description

This file provides API for get boot information from nor flash.

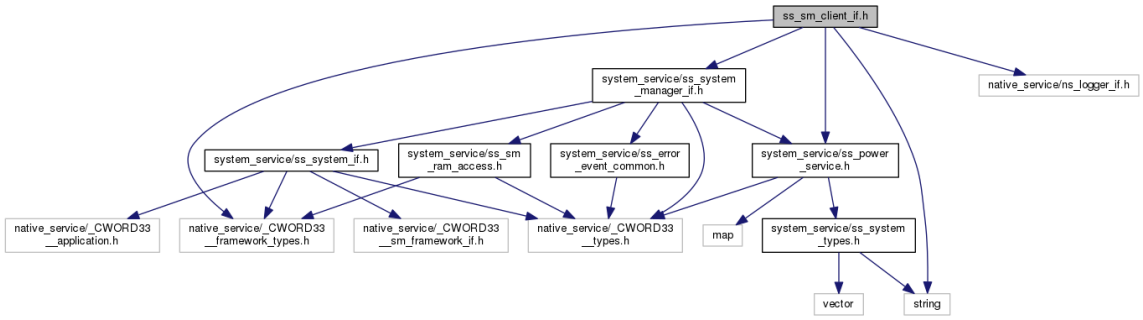
ss_sm_client_if.h File Reference

This file supports the _CWORD3_002 framework client interface to System manager.

#include <native_service/_CWORD33__framework_types.h>
#include <native_service/ns_logger_if.h>
#include <string>
#include "system_service/ss_power_service.h"
#include "system_service/ss_system_manager_if.h"

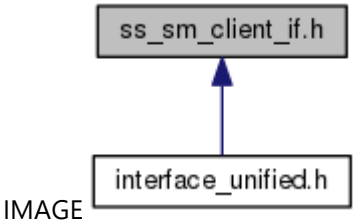
Include dependency graph for ss_sm_client_if.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Macros

#define SSDEBUGDUMP(args...) (SendDebugDumpResponseToSystemManager(TRUE, ## args))

Redirect either formatted, or unformatted response output to System Manager.

#define **SSDEBUGDUMP_RAW**(args...) (SendDebugDumpResponseToSystemManager(FALSE, ## args))

Functions

E_CWORD33_Status RegisterSMSessionAckCallback (E_CWORD33_Status(*CallbackPtr)(HANDLE))
E_CWORD33_Status SendBootModeSetRequestToSystemManager (ESMBootModeInfo hostBootMode)
E_CWORD33_Status SetDataResetModeToSystemManager (ESMDataResetModeInfo dataResetMode)
E_CWORD33_Status SetProgUpdateStateToSystemManager (SMProgUpdateState progUpdateState)

E_CWORD33_Status SendCpuResetRequestToSystemManager (ESMCpuResetReason l_eCpuResetReason, std::string f_messageStr="", std::string f_suffixStr="")

E_CWORD33_Status SetBootLoaderInfoRequestToSystemManager (ESMBootInfoType f_type, UI_32 f_value)

E_CWORD33_Status GetBootLoaderInfoRequestToSystemManager (void *p_info)

E_CWORD33_Status SetWakeupOrderToSystemManager (std::string f_orderName)

E_CWORD33_Status SetNextWakeupTypeToSystemManager (ESMNextWakeupType f_wakeupType)

E_CWORD33_Status SendVarCodeDataToSystemManager (UI_32 f_uiLength, void *f_data)

VOID Set_UseStopCompleteNotificationVs_CWORD33_OnStopFnc_StateVar (BOOL f_SetTrue)

E_CWORD33_Status SendUserInvokedLoggingRequestToSystemManager (eSMUserLogType f_userInvokedLogType, std::string f_messageStr="", std::string f_suffixStr="")

E_CWORD33_Status Send_CWORD33_OnStopResponseToSystemManager (E_CWORD33_Status f_eStatus)

E_CWORD33_Status Get_CWORD33_OnStartExtInfo (T_SS_SM_START_ExtDataStructType &f_info)

E_CWORD33_Status Get_CWORD33_OnStopExtInfo (T_SS_SM_STOP_ExtDataStructType &f_info)

void SendDebugDumpResponseToSystemManager(BOOL f_bFormatStrRequired, PCSTR f_cFormat,...) E_CWORD33_Status Send_CWORD33_EmmcLogsRequestToSystemManager(std E_CWORD33_Status SendClearLogsRequestToSystemManager (TSystemManagerClearLogsInfo *f_info)

E_CWORD33_Status AttachCallbackToSystemManager (HANDLE hApp, UI_32 iCmd, CbFuncPtr fpOnCmd)

AttachCallbackToSystemManager.

E_CWORD33_Status DetachCallbacksFrom_CWORD33_Dispatcher (HANDLE hApp, PCSTR pServiceName, const _CWORD33_ProtocolCallbackHandler *pMsgHandler, UI_32 uiHandlerCount, HANDLE hSession=NULL)

Set of complementary detach functions that accept same arguments as the _CWORD33_Attach_XYZ_ToDispatcher() functions do.

E_CWORD33_Status DetachParentCallbacksFrom_CWORD33_Dispatcher (HANDLE hChildApp, const _CWORD33_ProtocolCallbackHandler *pMsgHandler, UI_32 uiHandlerCount)

HANDLE GetSystemManagerSessionHandle (void)

GetSystemManagerSessionHandle.

E_CWORD33_Status SendShutdownToSystemManager (Pwr_ServiceSetInterface *pData)

SendShutdownToSystemManager.

E_CWORD33_Status SendSystemModeRequestToSystemManager (void)

SendSystemModeRequestToSystemManager.

E_CWORD33_Status SendInitCompReportToSystemManager (void)

SendInitCompReportToSystemManager.

E_CWORD33_Status SendWakeUpToSystemManager (wakeInfo *pData)

SendWakeUpToSystemManager.

E_CWORD33_Status SendStartupConfirmationToSystemManager (StartupConfirmationMsgStrut &l_startupConfirmationMsg)

SendStartupConfirmationToSystemManager.

E_CWORD33_Status SendSystemErrorToSystemManager (E_CWORD33_SystemError f_systemError)

SendSystemErrorToSystemManager.

E_CWORD33_Status Send_CWORD56_HeartBeatRequestToSystemManager (EPWR_HB_REQ_MSG_STRUCT f_HbReq)

Send_CWORD56_HeartBeatRequestToSystemManager.

E_CWORD33_Status SendBootMicroResetNotificationToSystemManager (eSMBootMicroResetReason ResetReason)

SendBootMicroResetNotificationToSystemManager.

E_CWORD33_Status SendDiagLoggingRequestToSystemManager (std::string f_copyDestPathStr)

SendDiagLogEventRequestToSystemManager.

E_CWORD33_Status SendCANLoggingRequestToSystemManager (void)

SendCANLoggingRequestToSystemManager.

E_CWORD33_Status SendDTCLoggingRequestToSystemManager (UI_32 f_dtc)

SendDTCLoggingRequestToSystemManager.

E_CWORD33_Status RegisterBootMicroLogRequestCb (HANDLE hApp, CbFuncPtr fpOnCmd)

RegisterBootMicroLogRequestCb.

E_CWORD33_Status SendBootMicroLogResponseToSystemManager (std::string f_logString)

SendBootMicroLogResponseToSystemManager.

E_CWORD33_Status SendLogStartRequestToSystemManager (EErrorEventType f_errorEventType)

SendLogStartRequestToSystemManager.

E_CWORD33_Status SendLogArtifactRequestToSystemManager (EArtifactId f_artifactId)

SendLogArtifactRequestToSystemManager.

E_CWORD33_Status SendLogCompleteRequestToSystemManager (E_CWORD33_Status f_eStatus=e_CWORD33_StatusOK)

SendLogCompleteRequestToSystemManager.

E_CWORD33_Status SendEelExportRequestToSystemManager (std::string f_path)

SendEelExportRequestToSystemManager.

E_CWORD33_Status OnSystemManagerDebugDump (HANDLE hApp)

DebugDump request callback function.

Detailed Description

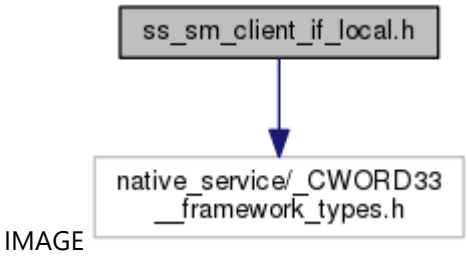
This file supports the _CWORD3_002 framework client interface to System manager.

ss_sm_client_if_local.h File Reference

This file supports the api of debug dump response to System Manager.

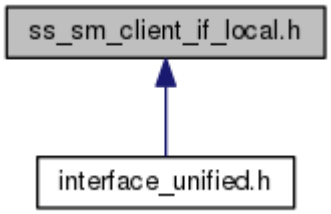
#include <native_service/_CWORD33__framework_types.h>

Include dependency graph for ss_sm_client_if_local.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

VOID **SendDebugDumpResponseToSystemManager** (std::string &f_messageStr)

Detailed Description

This file supports the api of debug dump response to System Manager.

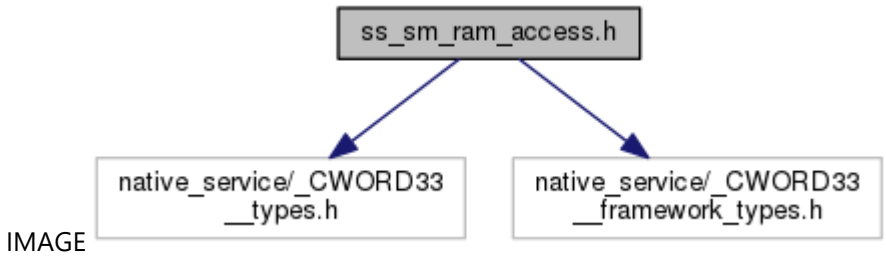
ss_sm_ram_access.h File Reference

This file provides API for get ram information from extension memory.

#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__framework_types.h>

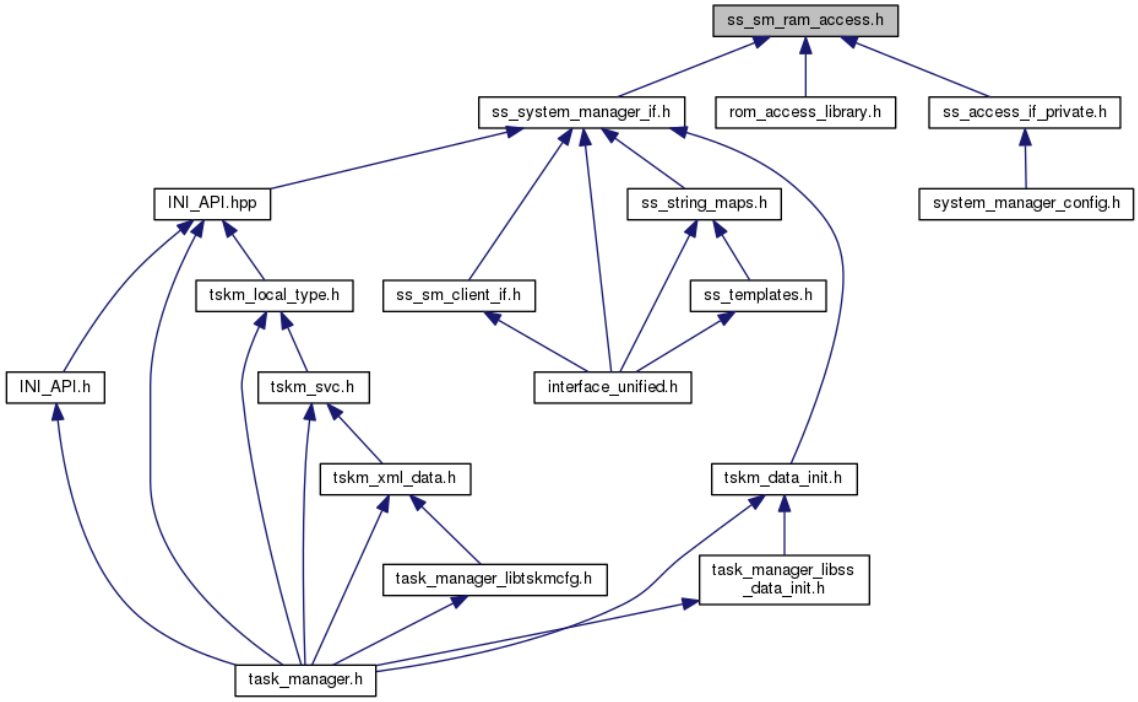
Include dependency graph for ss_sm_ram_access.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct RAM_SM_INFO_t
class RAM_AccessIf

***RAM_AccessIf*. Macros**

#define RAM_PRODUCT_PRIVATE_MAX (96)
#define SS_SM_ORDER_NAME_MAX (32)

Enumerations

enum RAM_WAKEUP_STATE { **RAM_WAKEUP_STATE_DONT_CARE** = ***,
RAM_WAKEUP_STATE_BACKUP_NG, **RAM_WAKEUP_STATE_BATTERY_DOWN** }

Detailed Description

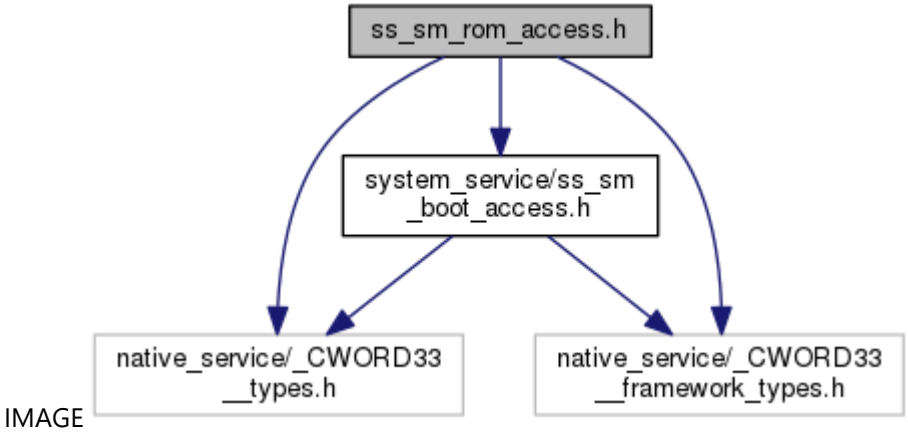
This file provides API for get ram information from extension memory.

ss_sm_rom_access.h File Reference

This file provides API for get rom information from extension memory.

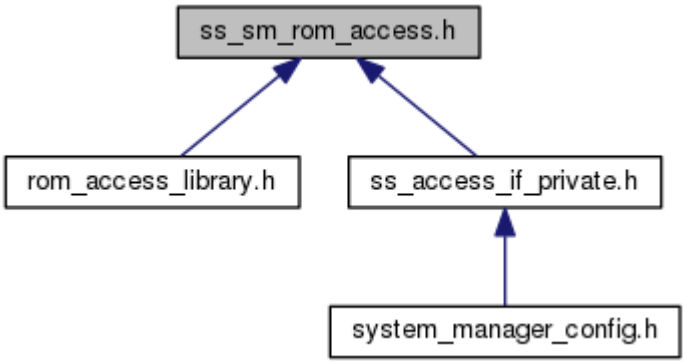
#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__framework_types.h>
#include "system_service/ss_sm_boot_access.h"

Include dependency graph for ss_sm_rom_access.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class ROM_AccessIf

***ROM_AccessIf*. Macros**

#define ROM_PRODUCT_PRIVATE_MAX 128

Typedefs

typedef uint32_t EPROGUPDATE_STATE

Enumerations

enum EBOOT_MODE { **APPLICATION_MODE** = ***, **PROGRAMMING_MODE** }

enum EACTIVE_FLASHLOADER { **NEW_FLASHLOADER** = ***, **OLD_FLASHLOADER** }
enum EUSER_MODE { **USER_OFF** = ***, **USER_ON** }
enum ECONTROL_MODE { **DISABLE_MODE** = ***, **ENABLE_MODE** }
enum EDATARESET_MODE { **DATARESET_NONE** = ***, **DATARESET_USER**,
DATARESET_FACTORY }
enum ELASTILGRESET_MODE { **LAST_ILGRESET_NORMAL** = ***, **LAST_ILGRESET_NG** }
enum ENEXT_WAKEUP_TYPE { **NEXT_WAKEUP_TYPE_NONE** = ***, **NEXT_WAKEUP_TYPE_COLD**,
NEXT_WAKEUP_TYPE_HOT }
enum DRAM_BACKUP_STATE { **DRAM_BACKUP_STATE_OK** = ***, **DRAM_BACKUP_STATE_NG** }

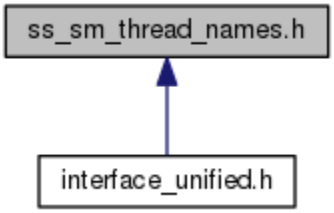
Detailed Description

This file provides API for get rom information from extension memory.

ss_sm_thread_names.h File Reference

This file supports System Manager thread naming.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **SS_SMHeartbeat** "SM.Heartbeat"

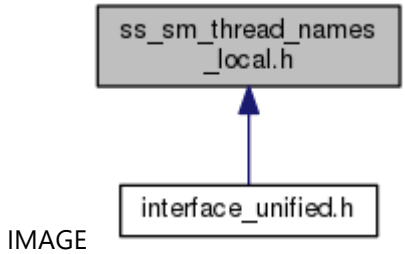
Detailed Description

This file supports System Manager thread naming.

ss_sm_thread_names_local.h File Reference

This file supports System Manager thread naming.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **SS_SMLauncher** "SM.ProcLaunch"
#define **SS_SMLowMemMonitor** "SM.LowMemMon"

Detailed Description

This file supports System Manager thread naming.

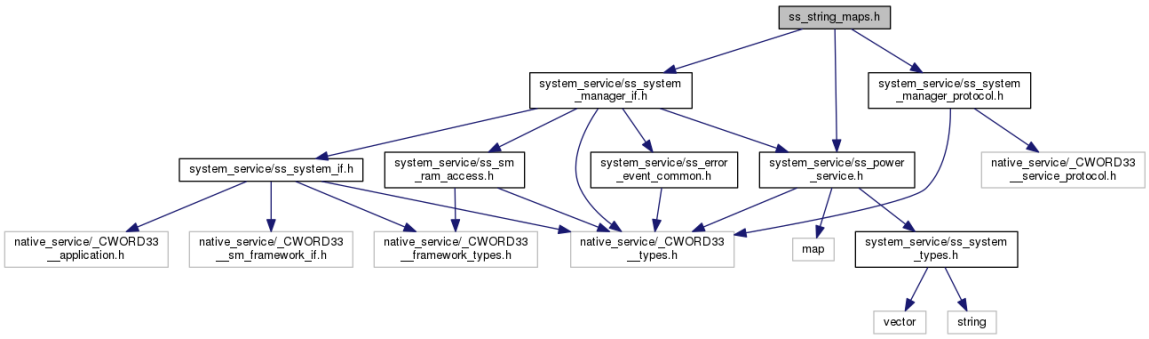
ss_string_maps.h File Reference

This file supports the a generic std::map<std::string> wrapper.

#include "system_service/ss_power_service.h"
#include "system_service/ss_system_manager_if.h"
#include "system_service/ss_system_manager_protocol.h"

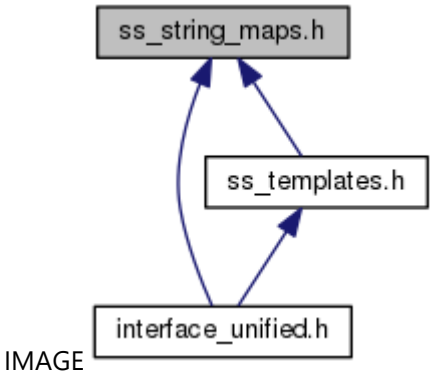
Include dependency graph for ss_string_maps.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Functions

SS_String GetStr (BOOL f_enum)
SS_String GetStr (EErrorEventType f_enum)
SS_String GetStr (E_CWORD33_SystemError f_enum)
SS_String GetStr (EPWR_LHC_TYPE f_enum)
SS_String GetStr (EPWR_PROD_MODE_TYPE f_enum)
SS_String GetStr (EPWR_SC_COLD_START_REQ_TYPE f_enum)
SS_String GetStr (EPWR_SC__CWORD56__BOOT_MODE_TYPE f_enum)
SS_String GetStr (EPWR_SHUTDOWN_TRIGGER_TYPE f_enum)
SS_String GetStr (EPWR_STARTUP_STAGE_TYPE f_enum)
SS_String GetStr (EPWR_SYSTEM_MODE_INFO f_enum)
SS_String GetStr (EPWR_TRANSPORT_MODE_TYPE f_enum)

SS_String GetStr (EPWR_USER_MODE_CHANGE_REASON_TYPE f_enum)
SS_String GetStr (EPWR_USER_MODE_TYPE f_enum)
SS_String GetStr (EPWR_WAKEUP_FACTOR_TYPE f_enum)
SS_String GetStr (E_CWORD33_Status f_enum)
SS_String GetStr (ESMBootModeInfo f_enum)
SS_String GetStr (ESMCpuResetReason f_enum)
SS_String GetStr (SS_SystemManagerProtocol f_enum)

Detailed Description

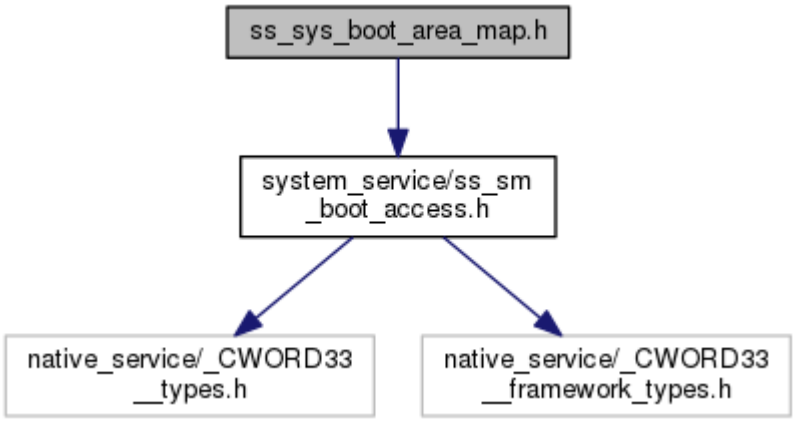
This file supports the a generic std::map<std::string> wrapper.

ss_sys_boot_area_map.h File Reference

This file supports system area type definitions.

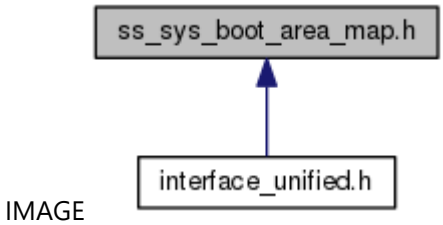
#include <system_service/ss_sm_boot_access.h>

Include dependency graph for ss_sys_boot_area_map.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Detailed Description

This file supports system area type definitions.

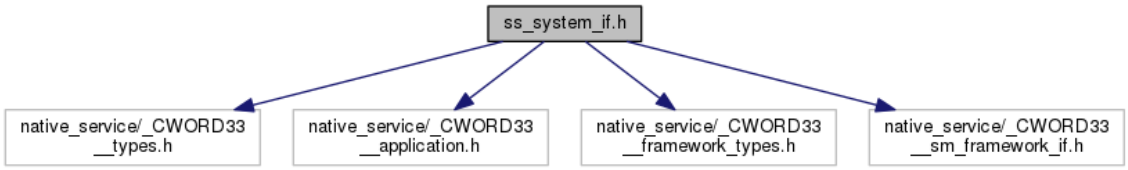
ss_system_if.h File Reference

This file contains api build the session of heartbeat observation system manager.

#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__application.h>
#include <native_service/_CWORD33__framework_types.h>
#include <native_service/_CWORD33__sm_framework_if.h>

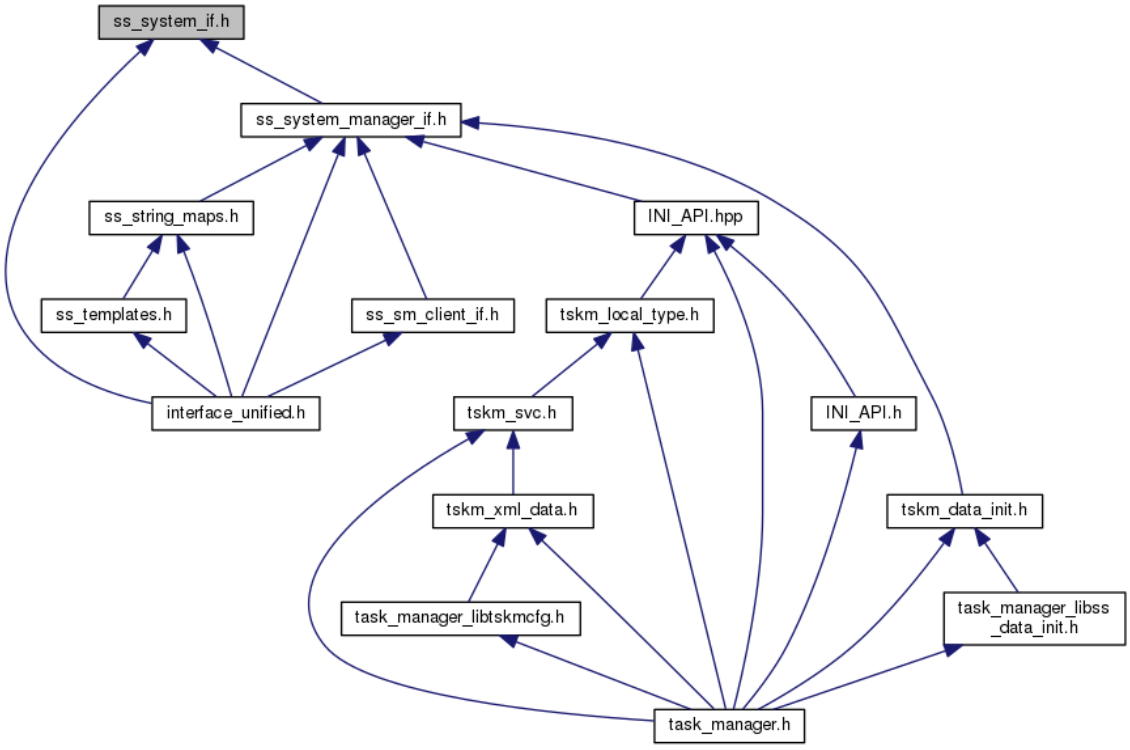
Include dependency graph for ss_system_if.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Macros

#define **_CWORD33__MAKE_DEFAULT_CALLBACK**(x)

Functions

E_CWORD33_Status **_CWORD33_SSFrameworkInterface** (HANDLE hApp)

Detailed Description

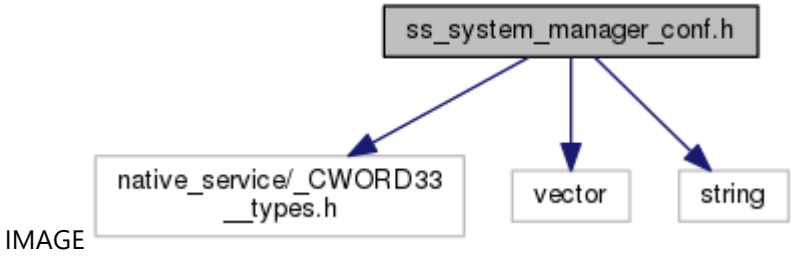
This file contains api build the session of heartbeat observation system manager.

ss_system_manager_conf.h File Reference

This file contains declaration of structures T_SS_SM_INIT_HOOK_IN_PARAM, T_SS_SM_INIT_HOOK_OUT_PARAM and api initial Hook.

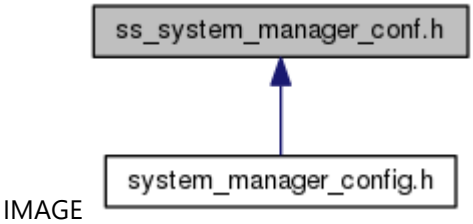
#include <native_service/_CWORD33__types.h>
#include <vector>
#include <string>

Include dependency graph for ss_system_manager_conf.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct T_SS_SM_INIT_HOOK

Version up mode, Callback function. struct T_SS_SM_INIT_HOOK_OUT_PARAM

Not in OOM Killer's service object, Group Relaunch service. *More...*

Typedefs

typedef struct T_SS_SM_INIT_HOOK **T_SS_SM_INIT_HOOK_IN_PARAM**

Functions

E_CWORD33_Status ss_sm_initHook (HANDLE hApp, const T_SS_SM_INIT_HOOK_IN_PARAM *inPrm, T_SS_SM_INIT_HOOK_OUT_PARAM *outPrm)

Detailed Description

This file contains declaration of structures T_SS_SM_INIT_HOOK_IN_PARAM, T_SS_SM_INIT_HOOK_OUT_PARAM and api initial Hook.

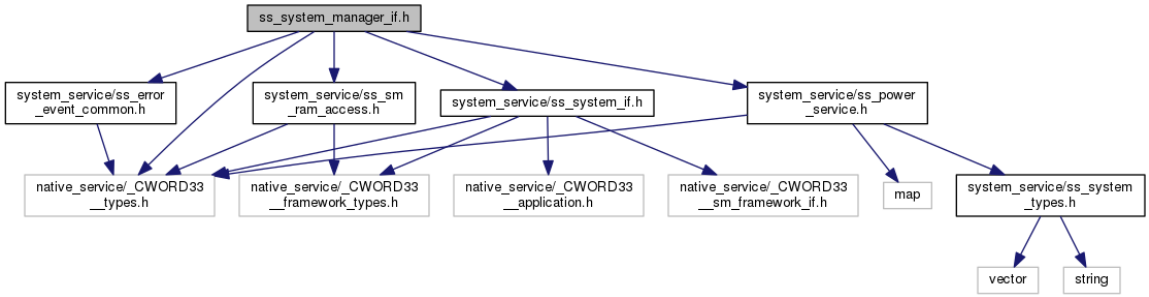
ss_system_manager_if.h File Reference

This file supports the System Manager client interface.

```
#include <native_service/_CWORD33__types.h>
#include <system_service/ss_sm_ram_access.h>
#include "system_service/ss_system_if.h"
#include "system_service/ss_error_event_common.h"
#include "system_service/ss_power_service.h"
```

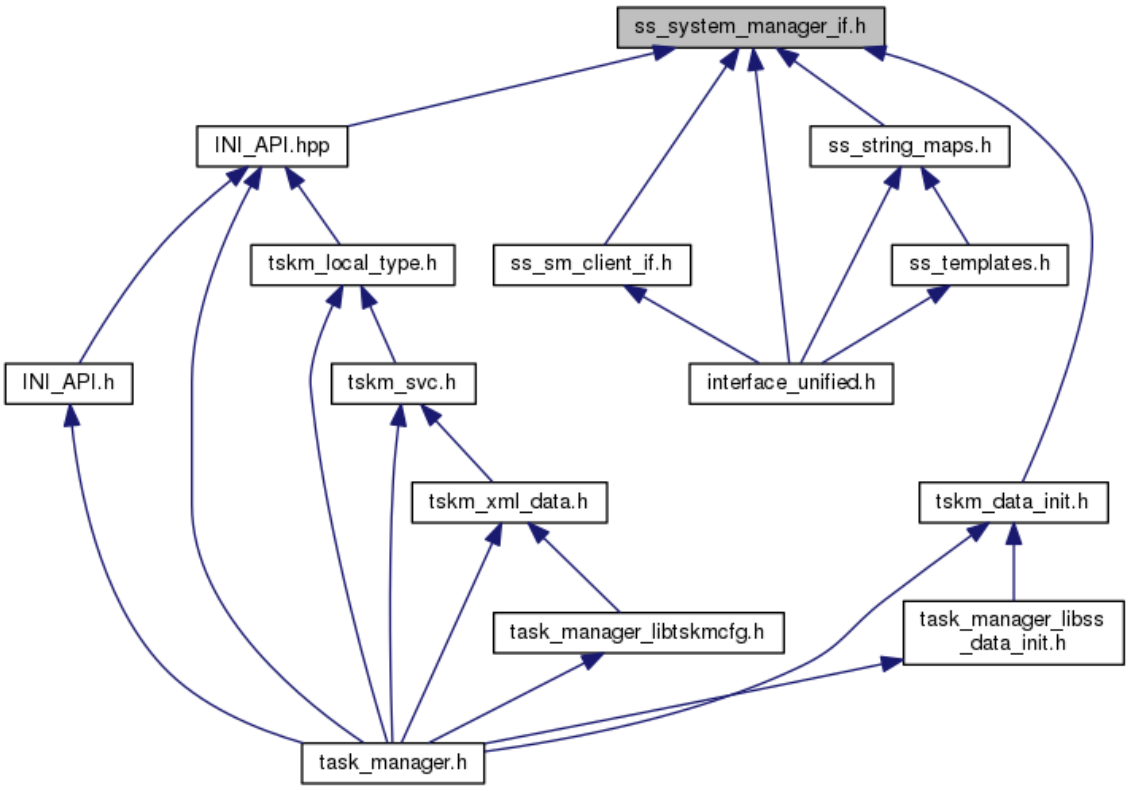
Include dependency graph for ss_system_manager_if.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct T_SystemManagerCpuResetInfo

CPU Reset Info. struct T_SystemManagerBootModeRequestResponse

struct TSystemManagerBootInfoStruct

struct T_SS_BootModeListStruct

Data Reset Reason. *More...*

struct T_SystemManagerWakeupOrderInfo

struct T_SS_SM_UserModeOnOffNotification_Struct

struct T_SS_SM_START_DataStruct

struct T_SS_SM_STOP_DataStruct

struct T_SS_SM_START_ExtDataStructType

struct T_SS_SM_STOP_ExtDataStructType

struct SMErrorEventNtfData

struct T_SystemManagerLoggingRequestInfo

*Error Event Logging Info *.* *More...*

struct T_SystemManagerClearLogsInfo

*Error Event Clear Logs Info *.* *More...*

Macros

#define **SS_SM_PROG_UPDATE_STATE_NONE** 0x0000

#define **SS_SM_PROG_UPDATE_STATE_UPDATED** 0x0001

#define **SS_SM_PROG_UPDATE_STATE_MAP_UPDATED** 0x0002

#define **SS_SM_PROG_UPDATE_STATE_MAPDIFF_UPDATED** 0x0004

#define **SS_SM_RESET_MSG_STR_SIZE** 64

#define **SS_SM_SUFFIX_STR_SIZE** 32

#define **SS_SM_START_EXT_INFO_SIZE** 64

#define **SS_SM_START_EXT_BODY_SIZE**

#define **SS_SM_START_EXT_RSV_SIZE**

#define **SS_SM_STOP_EXT_INFO_SIZE** 64

#define **SS_SM_STOP_EXT_BODY_SIZE** (sizeof(BOOL))

#define **SS_SM_STOP_EXT_RSV_SIZE**

#define **SM_MODULE_NAME_SIZE** 64

#define **SS_SM_LOG_MSG_STR_SIZE** 64

Typedefs

typedef UI_32 SMProgUpdateState

Program update status.

typedef struct T_SystemManagerCpuResetInfo TSystemManagerCpuResetInfo

CPU Reset Info.

typedef struct T_SystemManagerBootModeRequestResponse

TSystemManagerBootModeRequestResponse

typedef struct T_SS_BootModeListStruct TSS_BootModeListStruct

Data Reset Reason.

typedef struct T_SystemManagerWakeupOrderInfo **TSystemManagerWakeupOrderInfo**

typedef struct T_SS_SM_UserModeOnOffNotification_Struct
T_SS_SM_UserModeOnOffNotification_StructType
typedef struct T_SS_SM_START_DataStruct T_SS_SM_START_DataStructType
typedef struct T_SS_SM_STOP_DataStruct T_SS_SM_STOP_DataStructType
typedef struct T_SystemManagerLoggingRequestInfo TSystemManagerLoggingRequestInfo
*Error Event Logging Info *.*
typedef struct T_SystemManagerClearLogsInfo TSystemManagerClearLogsInfo
*Error Event Clear Logs Info *.*

Enumerations

enum ESMBootModeInfo { **e_SS_SM_BOOT_MODE_INVALID** = -1, **e_SS_SM_BOOT_MODE_APPLICATION**, **e_SS_SM_BOOT_MODE_PROGRAMMING** }*Boot mode information.*

enum ESMDataResetModeInfo { **e_SS_SM_DATA_RESET_MODE_NONE**, **e_SS_SM_DATA_RESET_MODE_USER**, **e_SS_SM_DATA_RESET_MODE_FACTORY**, **e_SS_SM_DATA_RESET_MODE_PROGUPDATE** }*Data reset mode information.*

enum ESMNextWakeupType { **e_SS_SM_NEXT_WAKEUP_TYPE_NONE** = ***, **e_SS_SM_NEXT_WAKEUP_TYPE_COLD**, **e_SS_SM_NEXT_WAKEUP_TYPE_HOT** }*Set Next Wakeup type information.*

enum ESMDramBackupStatus { **e_SS_SM_DRAM_BACKUP_UNSET** = -1, **e_SS_SM_DRAM_BACKUP_OK**, **e_SS_SM_DRAM_BACKUP_NG** }*DRAM backup status.*

enum ESMResetStatus { **e_SS_SM_RESET_STATUS_UNSET** = -1, **e_SS_SM_RESET_STATUS_NONE**, **e_SS_SM_RESET_STATUS_NG**, **e_SS_SM_RESET_STATUS_IMMEDIATE** }*Reset status.*

enum EMRelaunchStatus { **e_SS_SM_RELAUNCH_STATUS_NONE**, **e_SS_SM_RELAUNCH_STATUS_SAFE**, **e_SS_SM_RELAUNCH_STATUS_ERR** }*Relaunch status.*

enum ESMCpuResetReason { **e_SS_SM_CPU_RESET_REASON_INVALID** = -1, **e_SS_SM_CPU_RESET_REASON_NORMAL** = ***, **e_SS_SM_CPU_RESET_REASON_DATA_RESET**, **e_SS_SM_CPU_RESET_REASON_USER_FORCE_RESET**, **e_SS_SM_CPU_RESET_REASON_GENERIC**, **e_SS_SM_CPU_RESET_REASON_GENERIC_ERR** = e_SS_SM_CPU_RESET_REASON_GENERIC, **e_SS_SM_CPU_RESET_REASON_DSP_ERR**, **e_SS_SM_CPU_RESET_REASON_IMMRESET_NORMAL**, **e_SS_SM_CPU_RESET_REASON_CRITICAL_ERR** }*CPU Reset Reason.*

enum ESMDataResetType { **e_SS_SM_DATA_RESET_TYPE_USER** = ***, **e_SS_SM_DATA_RESET_TYPE_FACTORY**, **e_SS_SM_DATA_RESET_TYPE_CONFIGURATION**, **e_SS_SM_DATA_RESET_TYPE_PROGUPDATE** }*Data Reset Reason.*

enum **ESMServiceWakeupStatus** { **e_SS_SM_SVC_WAKEUP_STATUS_COMPLETE** = *** }

enum **ESMBootInfoType** { **e_SS_SM_BOOT_INFO_MDUPDATE**,
e_SS_SM_BOOT_INFO_SDUPDATE, **e_SS_SM_BOOT_INFO_OPDTUPDATE** }
enum **eSMBootMicroResetReason** { **SS_SM_BOOT_MICRO_RESET_REASON_SELF_RESET** = ***,
SS_SM_BOOT_MICRO_RESET_REASON_USER_FORCE_RESET,
SS_SM_BOOT_MICRO_RESET_REASON_DSP_RESET,
SS_SM_BOOT_MICRO_RESET_REASON_HB_TIMEOUT }
enum eSMUserLogType { e_SS_SM_CAPTURE_ALL_LOGS = ***, e_SS_SM_SCREEN_CAPTURE,
e_SS_SM_CAPTURE_CWORD33_LOGS, e_SS_SM_USER_FORCE_RESET,
e_SS_SM_CAPTURE_DEV_LOGS, e_SS_SM_CAPTURE_MODULE_LOGS,
e_SS_SM_CAPTURE_DTC_LOGS, e_SS_SM_CAPTURE_NAVI_LOGS,
e_SS_SM_CAPTURE_GROUP_RELAUNCH }

Detailed Description

This file supports the System Manager client interface.

Class Documentation

struct TSystemManagerBootInfoStruct

Class Members:

ESMBootInfoType	type	
UI_32	value	

struct T_SystemManagerWakeupOrderInfo

Class Members:

char	order_name[SS_SM_ORDER_NAME_MAX]	

struct SMErrorEventNtfData

Class Members:

EErrorEventType	EventType	
BOOL	isNeedReboot	
CHAR	ModuleName[SM_MODULE_NAME_SIZE]	

Macro Definition Documentation

#define SS_SM_START_EXT_BODY_SIZE

Value:(sizeof(BOOL) \
+ sizeof(EMRelaunchStatus) \
+ sizeof(BOOL) \

```
                + sizeof(BOOL) \
                )
```

#define SS_SM_START_EXT_RSV_SIZE

Value:SS_SM_START_EXT_INFO_SIZE \
- SS_SM_START_EXT_BODY_SIZE

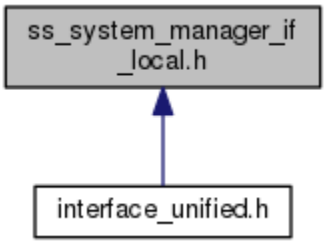
#define SS_SM_STOP_EXT_RSV_SIZE

Value:SS_SM_STOP_EXT_INFO_SIZE \
- SS_SM_STOP_EXT_BODY_SIZE

ss_system_manager_if_local.h File Reference

This file supports the System Manager client interface.

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct T_SMRequestMsg

Macros

#define **SS_MAX_NUM_MODULES** (128)
#define **SS_SM_EXIT_RELAUNCH** (2)

Typedefs

typedef struct T_SMRequestMsg TSMRequestMessage

Detailed Description

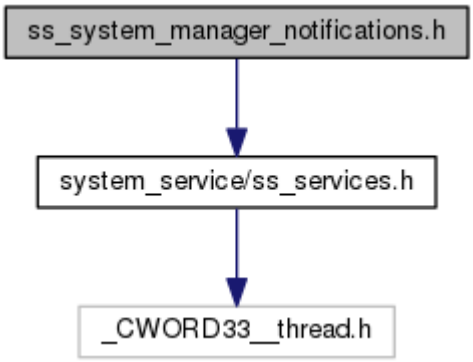
This file supports the System Manager client interface.

ss_system_manager_notifications.h File Reference

This file supports the System Manager client interface.

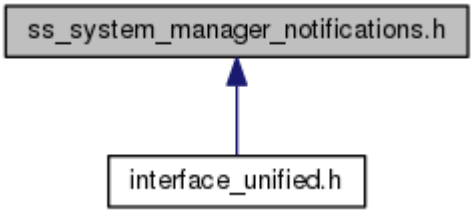
#include "system_service/ss_services.h"

Include dependency graph for ss_system_manager_notifications.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define NTFY_SSSystemMgrAvailability SERVICE_SYSMANAGER"/Availability"

System Manager availability -.

#define NTFY_SSSystemMgrState SERVICE_SYSMANAGER"/State"

System Manager state - Init,.

#define NTFY_SSSystemMgrPowerOnOff SERVICE_SYSMANAGER"/PowerOnOffState"

Power ON notification for.

#define NTFY_SSSystemMgrStartUpType SERVICE_SYSMANAGER"/StartUpType"

Notification for start up type.

#define NTFY_SSHUBootModes SERVICE_SYSMANAGER"/HeadUnitBootModes"

Notification for boot modes.

#define NTFY_SSServiceWakeupStatus SERVICE_SYSMANAGER"/ServiceWakeupStatus"

Service Wakeup state.

#define NTFY_SSNeedAplRestart SERVICE_SYSMANAGER"/NeedAplRestart"

Notification for Need APL Restart.

#define NTFY_SSSystemMgrShutdownStarted SERVICE_SYSMANAGER"/ShutdownStarted"

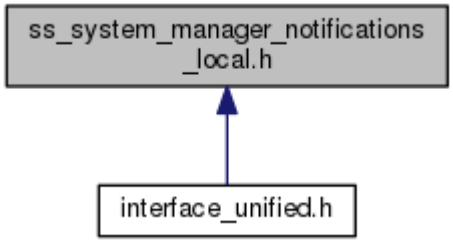
Detailed Description

This file supports the System Manager client interface.

ss_system_manager_notifications_local.h File Reference

This file supports the System Manager client interface.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define NTFY_SSSystemMgrUserMode SERVICE_SYSMANAGER"/UserMode"

#define NTFY_SSSystemMgrDataReset SERVICE_SYSMANAGER"/DataReset"

Notification for data reset.

#define NTFY_SSLastSourceType SERVICE_SYSMANAGER"/LastSourceType"

Notification used by Audio and System Services for Dynamic Launch.

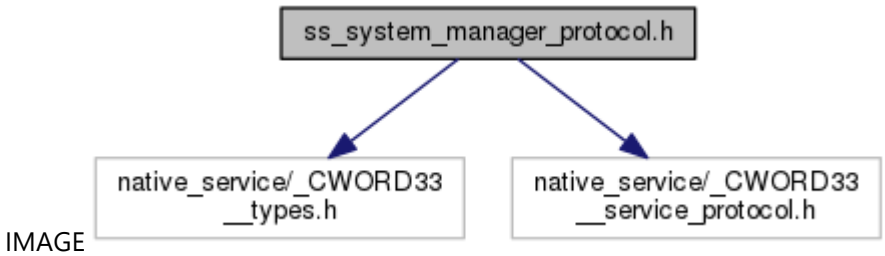
Detailed Description

This file supports the System Manager client interface.

ss_system_manager_protocol.h File Reference

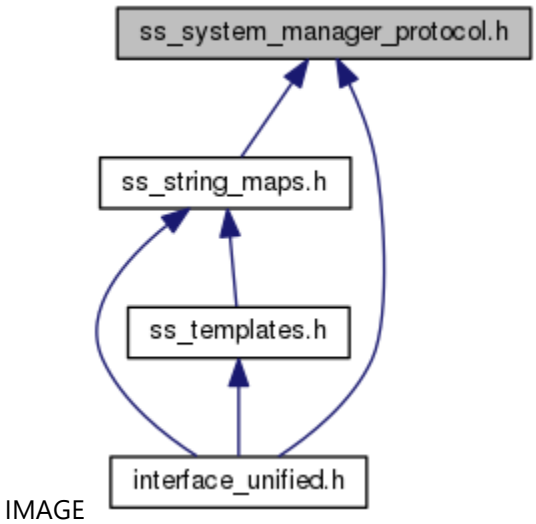
This file supports the System Manager client interface.

#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__service_protocol.h>
Include dependency graph for ss_system_manager_protocol.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _SS_SMCurrentState
struct _SMCompleteAck
struct _SMGroupLaunchTrigger

Macros

#define **SS_SM_MAX_MODULE_NAME_LENGTH** (256)
#define **SS_SM_MAX_TEST_CASE_NAME_SIZE** (128)
#define **SS_SM_MAX_RESP_MSG_SIZE** (2048)

Typedefs

typedef enum _SS_SystemManagerProtocol SS_SystemManagerProtocol

typedef struct _SS_SMCurrentState **SS_SMCurrentState**
typedef struct _SMCompleteAck **SMStopCompleteAck**
typedef struct _SMGroupLaunchTrigger **SMGroupLaunchTrigger**

Enumerations

enum _SS_SystemManagerProtocol { **SS_SYSTEM_MANAGER_PROTOCOL_BEGINNING_INDEX** = ***, SS_SM_NOTIFY_SYSTEM_LAUNCH_COMPLETE = SS_SYSTEM_MANAGER_PROTOCOL_BEGINNING_INDEX, **SS_SM_PROTOCOL_INTERFACE_END**, SS_SM_START, SS_SM_START_COMPL_RSPN, SS_SM_STOP, SS_SM_STOP_COMPL_RSPN, SS_SM_WAKEUP_MODULES_CMPL_RSPN, SS_SM_SHUTDOWN_MODULES_CMPL_RSPN, SS_SM_WAKEUP_MODULES, SS_SM_POWER_OFF_MODULES, SS_SM_SHUTDOWN_MODULES, SS_SM_CRNT_STATE_QUERY, SS_SM_CRNT_STATE_QUERY_RSPN, SS_SM_SYSTEM_MODE_INFO_REQ, SS_SM_SYSTEM_MODE_INFO_RSPN, SS_SM_INITCOMP_REP, **SS_SM_PROTOCOL_OPEN_SESSION_REQ**, **SS_SM_PROTOCOL_OPEN_SESSION_ACK**, SS_SM_GET_START_EXT_INFO, SS_SM_GET_STOP_EXT_INFO, SS_SM_BOOT_MODE_SET_REQ, SS_SM_BOOT_MODE_SET_RESP, SS_SM_DATA_RESET_MODE_SET_REQ, SS_SM_PROG_UPDATE_STATE_SET_REQ, **SS_SM_CPU_RESET_REQ**, **SS_SM_LOCAL_DATA_RESET_REQ**, **SS_SM_REMOTE_DATA_RESET_REQ**, SS_SM_FWD_STARTUP_CONFIRMATION_MSG_REQ, SS_SM_FWD_START_CONFIRMATION_MSG_RESP, SS_SM_POWER_REQUEST_MSG, SS_SM_POWER_REQUEST_MSG_RESP, SS_SM_USER_MODE_SET_RESP, **SS_SM_DEBUG_VAR_CODE_DATA_SET_REQ**, **SS_SM_SET_BOOTLOADER_INFO**, **SS_SM_GET_BOOTLOADER_INFO**, **SS_SM_WAKEUP_ORDER_SET_REQ**, **SS_SM_NEXT_WAKEUP_TYPE_SET_REQ**, SS_SM_EVENT_ERROR, SS_SM_EVENT_ERROR_TO_SSL, SS_SM_ERROR_EVENT_LOGGING_START_REQ, SS_SM_ERROR_EVENT_LOGGING_START_RSPN, SS_SM_ERROR_EVENT_ARTIFACT_REQ, SS_SM_ERROR_EVENT_ARTIFACT_RSPN, SS_SM_ERROR_EVENT_LOGGING_COMPLETE, SS_SM_ERROR_EVENT_LOGGING_COMPLETE_RSPN, **SS_SM_ERROR_EVENT_TIMER_ID_LOGGER_START_REQ**, **SS_SM_ERROR_EVENT_TIMER_ID_DEBUG_DUMP_RSPN**, **SS_SM_ERROR_EVENT_TIMER_ID_BOOT_MICRO_LOG_RSPN**, **SS_SM_ERROR_EVENT_TIMER_ID_CORE_FILE_POLL**, SS_SM_USER_INVOKED_LOG_REQ, SS_SM_ERROR_EVENT_EEL_EXPORT_REQ, SS_SM_ERROR_EVENT__CWORD33__EMMC_LOGS_REQ, SS_SM_ERROR_EVENT__CWORD33__EMMC_LOGS_RSPN, SS_SM_ERROR_EVENT_CLR_LOGS_REQ, SS_SM_ERROR_EVENT_PROCESS_EXIT, SS_SM_ERROR_EVENT_DIAG_LOG_REQ, SS_SM_ERROR_EVENT_CAN_LOG_REQ, SS_SM_ERROR_EVENT_DTC_LOG_REQ, **SS_SM_GROUP_LAUNCH_TRIGGER**, SS_SM_DEBUG_DUMP, SS_SM_DEBUG_DUMP_RSPN, SS_SM_MODULE_RELAUNCH_REQ, SS_SM_CPU_HIGH_LOAD_DETECTED, SS_SM_PROPAGATE_SYSTEM_ERROR, SS_SM__CWORD56__HEARTBEAT_REQ, SS_SM__CWORD56__HEARTBEAT_RSPN, SS_SM_EVENT_USER_DATA_RESET, SS_SM_BOOT_MICRO_RESET_NTF, SS_SM_BOOT_MICRO_LOG_REQ, SS_SM_BOOT_MICRO_LOG_RSP, SS_SM_PRE_START, SS_SM_PRE_START_COMPL_RSPN, SS_SM_PRE_STOP, SS_SM_PRE_STOP_COMPL_RSPN, SS_SM_BACKGROUND_START, SS_SM_BACKGROUND_START_COMPL_RSPN, SS_SM_BACKGROUND_STOP, SS_SM_BACKGROUND_STOP_COMPL_RSPN,

SS_SYSTEM_MANAGER_PROTOCOL_ENDING_INDEX =
SS_SM_BACKGROUND_STOP_COMPL_RSPN }

Detailed Description

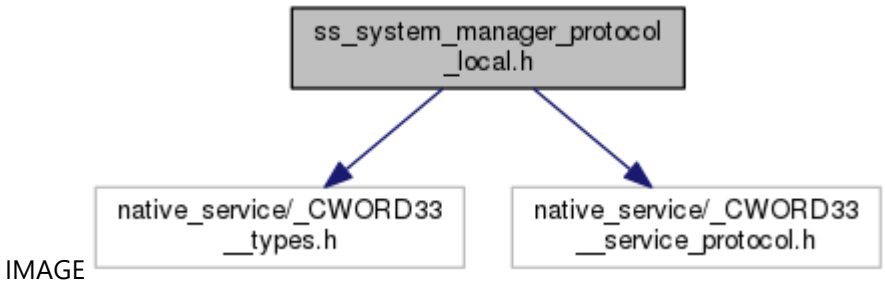
This file supports the System Manager client interface.

ss_system_manager_protocol_local.h File Reference

This file supports the System Manager client interface.

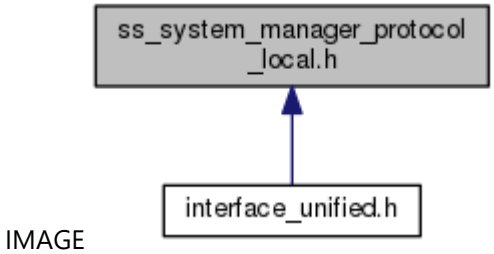
#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__service_protocol.h>

Include dependency graph for ss_system_manager_protocol_local.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **SS_SM_MAX_MODULE_LOG_MASK_LENGTH** (50)
#define **SS_SM_MAX_MODULE_PATH_LENGTH** (256)
#define **SS_SM_MAX_MODULE_ARGS_LENGTH** (512)

Typedefs

typedef UI_64 **SS_VersionNumberType**

Detailed Description

This file supports the System Manager client interface.

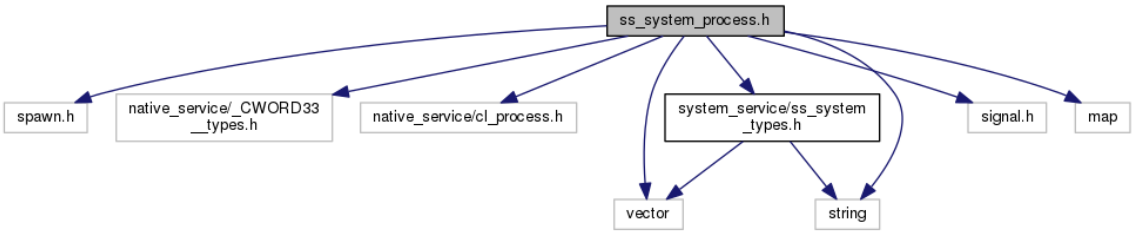
ss_system_process.h File Reference

This file supports the System Manager OS process abstraction.

#include <spawn.h>
#include <native_service/_CWORD33__types.h>
#include <native_service/cl_process.h>
#include <system_service/ss_system_types.h>
#include <signal.h>
#include <vector>
#include <string>
#include <map>

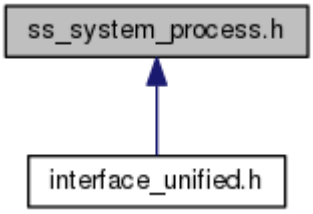
Include dependency graph for ss_system_process.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

class Process

Typedefs

typedef std::map< SS_String, Process * > **ProcessMap**

Variables

const long **IProcess_VALIDATION_VALUE** = ***
const int **iProcess_DEFAULT_PROCESS_PRIORITY** = ***
const int **iProcess_MAXIMUM_PROCESS_PRIORITY** = ***
const long **iProcess_DEFAULT_PROCESS_FLAGS**
const int **iProcess_MAXIMUM_NUMBER_OF_PROCESS_ARGUMENTS** = ***
const int **iProcess_MAXIMUM_PROCESS_NAME_LENGTH** = ***

Detailed Description

This file supports the System Manager OS process abstraction.

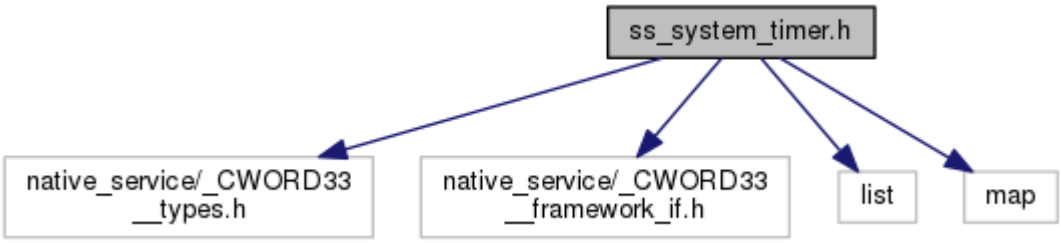
ss_system_timer.h File Reference

brief This file supports a generic timer abstraction.

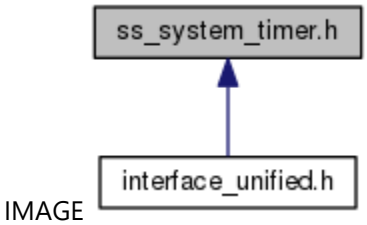
#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__framework_if.h>
#include <list>
#include <map>

Include dependency graph for ss_system_timer.h:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class Timer

Timer. class TimerCtrl

***OSprocess*. Variables**

const PCSTR **TIMER_SERVICE_NAME** = "TIMER"

Detailed Description

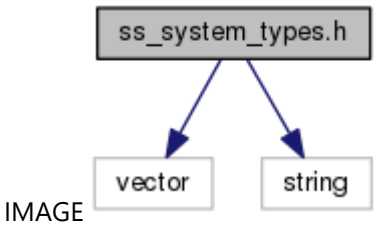
brief This file supports a generic timer abstraction.

ss_system_types.h File Reference

This file supports string vector type definitions.

#include <vector>
#include <string>

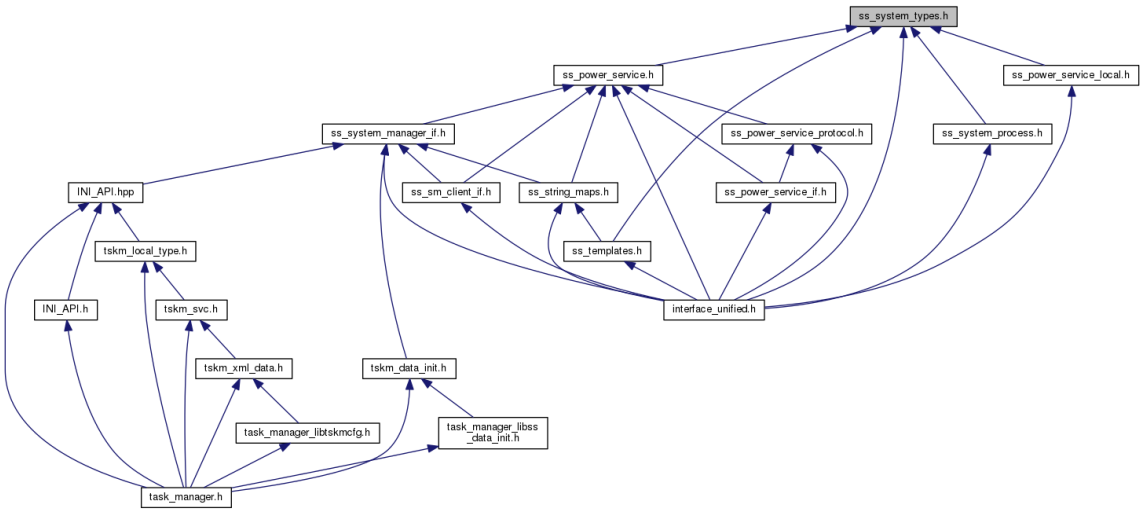
Include dependency graph for ss_system_types.h:



IMAGE

This graph shows which files directly or indirectly include this file:

IMAGE



Typedefs

typedef std::string **SS_String**
typedef std::vector< std::string > **StringList**
typedef std::vector< std::string >::iterator **StringListIter**

Detailed Description

This file supports string vector type definitions.

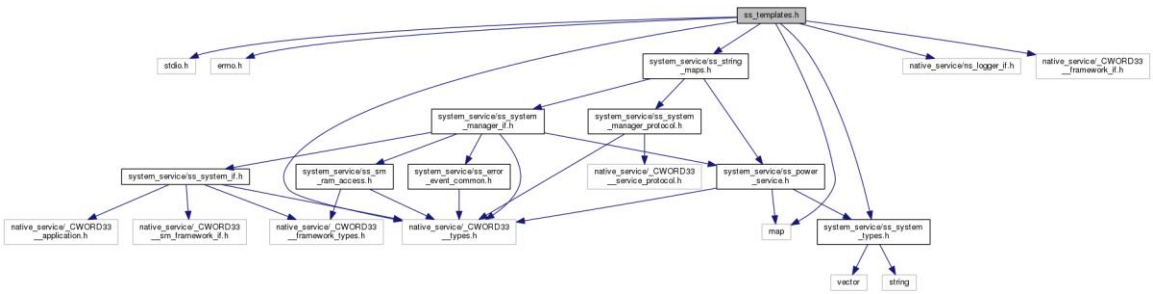
ss_templates.h File Reference

This file supports templates for high use common tasks.

```
#include <stdio.h>
#include <errno.h>
#include <native_service/_CWORD33__types.h>
#include <native_service/ns_logger_if.h>
#include <native_service/_CWORD33__framework_if.h>
#include <map>
#include "system_service/ss_system_types.h"
#include "system_service/ss_string_maps.h"
```

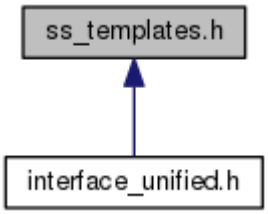
Include dependency graph for ss_templates.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

class EnumStringMap< enumT, fp >

***EnumStringMap*. Macros**

```
#define ZONE_ERR  ZONEMASK(31)
#define LOG_ERROR(pStr)
#define LOG_SUCCESS(pStr)  _CWORD33_LOG(ZONE_INFO, __FUNCTION__, " %s successful",
    pStr);
#define LOG_STATUS(l_eStatus,  pStr)
#define CALL_AND_LOG_STATUS(fnc)
#define LOG_STATUS_IF_ERRORED(l_eStatus,  pStr)
#define CALL_AND_LOG_STATUS_IF_ERRORED(fnc)
#define MAP_ENTRY(f_map,  f_enum)  f_map[ f_enum ] = #f_enum
```

#define _CWORD33_LOG_RECEIVED_FROM(hApp) _CWORD33_LOG(ZONE_INFO, __FUNCTION__, " Received from %s", _CWORD33_GetMsgSrc(hApp))
#define _CWORD33_LOG_WHEN_COMPILED
#define **SS_ASERT**(x)
#define **SS_ASERT_ERRNO**(x)
#define **SS_ASERT_LOG**(x, fmt, ...)
#define **SS_STATIC_ASERT**(expr)

Functions

template<typename T > E_CWORD33_Status ReadMsg (HANDLE hApp, T &Data, ESMRetrieveTypes f_eRetrieveMethod=eSMRRelease)

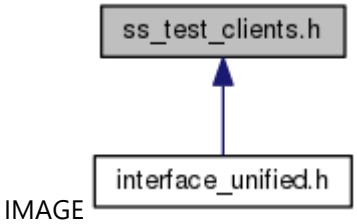
Detailed Description

This file supports templates for high use common tasks.

ss_test_clients.h File Reference

This file supports test client queue names.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **TC_Power** "PwrTestClient"
#define **TC_SysManager** "SMTestClient"

Detailed Description

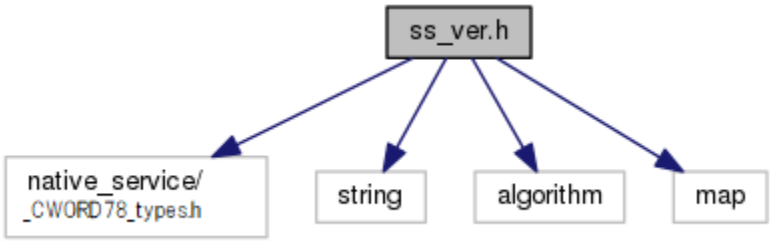
This file supports test client queue names.

ss_ver.h File Reference

This file supports CSSVer class, This class is a generic CSSVer abstraction.

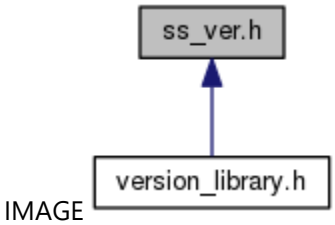
#include <native_service/_CWORD78__types.h>
#include <string>
#include <algorithm>
#include <map>

Include dependency graph for ss_ver.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct SSVER_PkgInfo

The structure of version package information. *More...*

class CSSVer

Version information set-up/acquisition. **Macros**

#define **SSVER_VERCHAR_MAX** (64)
#define **SSVER_DATE_MAX** (16)

Typedefs

typedef std::map< std::string, SSVER_PkgInfo > **SSVerPkgList**
typedef SSVerPkgList::const_iterator **SSVerPkgListIter**

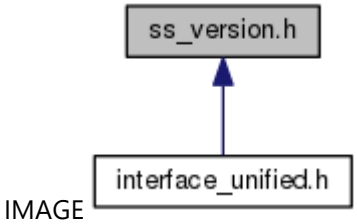
Detailed Description

This file supports CSSVer class, This class is a generic CSSVer abstraction.

ss_version.h File Reference

This file supports System Manager version number management.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **MAJORNO** 0x01
#define **MINORNO** 0x00
#define **REVISION** 0x01

Detailed Description

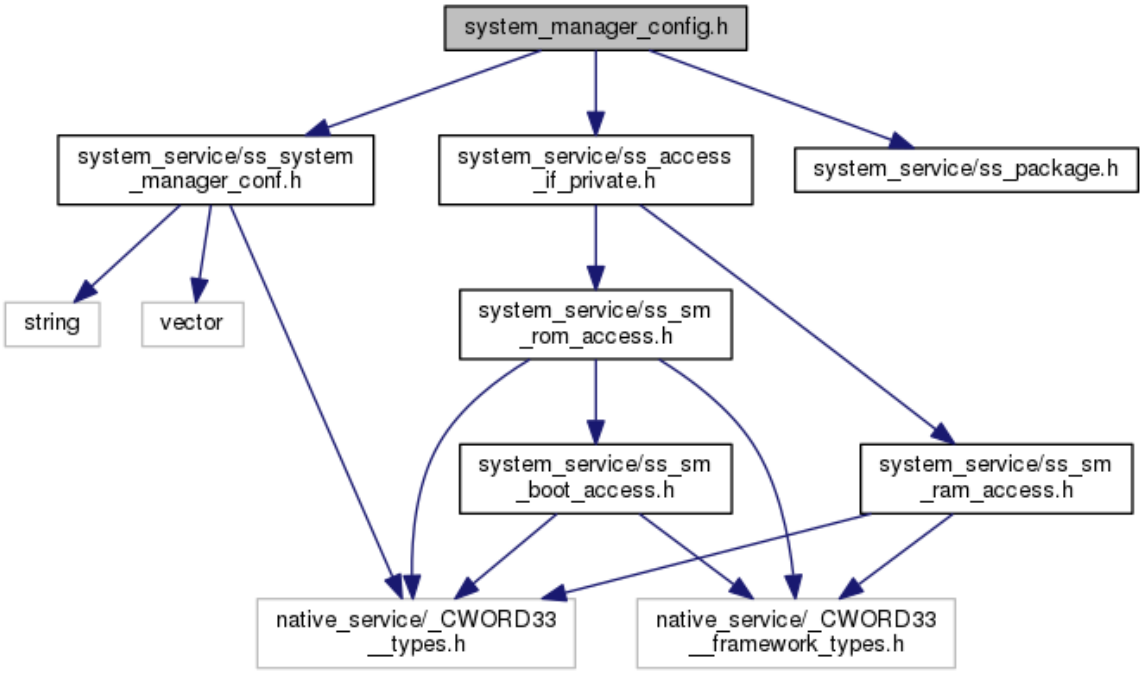
This file supports System Manager version number management.

system_manager_config.h File Reference

#include <system_service/ss_access_if_private.h>
#include <system_service/ss_package.h>
#include <system_service/ss_system_manager_conf.h>

Include dependency graph for system_manager_config.h:

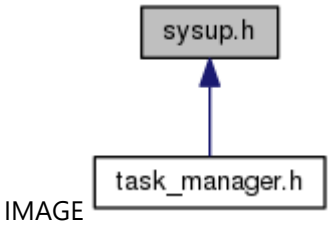
IMAGE



sysup.h File Reference

This file contains declaration of system status(SYS_RESET, SYS_ONLY_RESET)

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **SYS_RESET** (int32)1 /* */
#define **SYS_ONLY_RESET** (int)2 /* #9# */

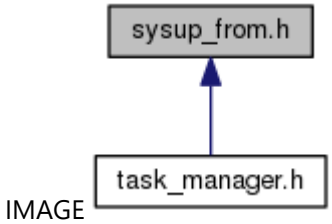
Detailed Description

This file contains declaration of system status(SYS_RESET, SYS_ONLY_RESET)

sysup_from.h File Reference

empty file

This graph shows which files directly or indirectly include this file:



IMAGE

Detailed Description

empty file

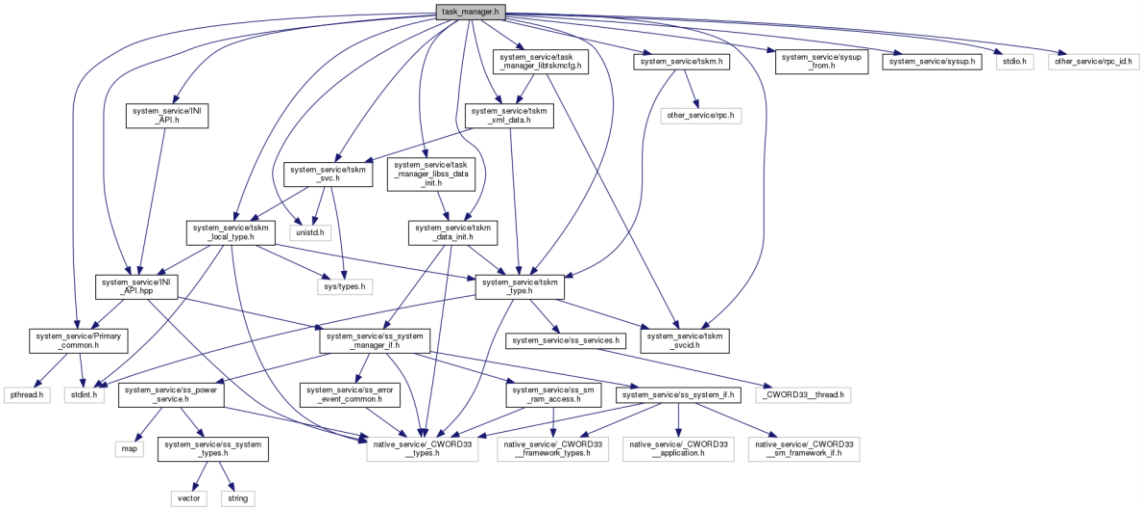
task_manager.h File Reference

include all task_manager_library head files

#include "system_service/INI_API.h"
#include "system_service/INI_API.hpp"
#include "system_service/Primary_common.h"
#include "system_service/sysup_from.h"
#include "system_service/sysup.h"
#include "system_service/tskm_svc.h"
#include "system_service/tskm_type.h"
#include "system_service/tskm.h"
#include "system_service/tskm_local_type.h"
#include "system_service/tskm_xml_data.h"
#include "system_service/tskm_svcid.h"
#include "system_service/task_manager_libtskmcfg.h"
#include "system_service/tskm_data_init.h"
#include "system_service/task_manager_libss_data_init.h"

Include dependency graph for task_manager.h:

IMAGE



Detailed Description

include all task_manager_library head files

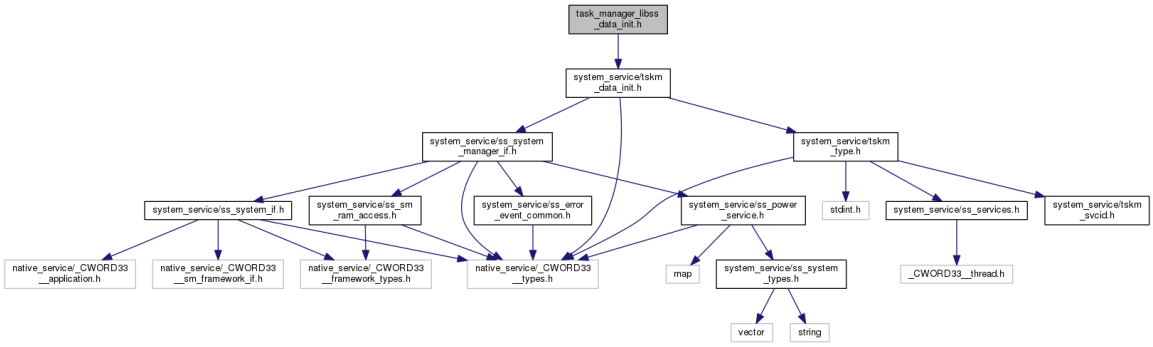
task_manager_libss_data_init.h File Reference

This file include tskm_data_init.h.

#include "system_service/tskm_data_init.h"

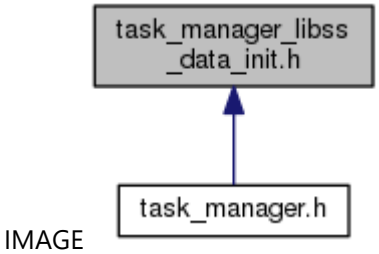
Include dependency graph for task_manager_libss_data_init.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Detailed Description

This file include tskm_data_init.h.

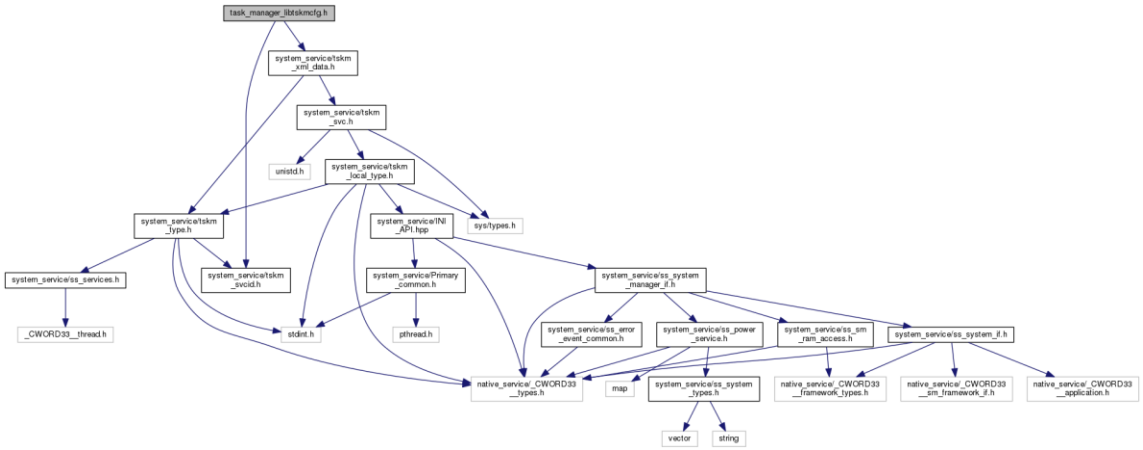
task_manager_libtskmcfg.h File Reference

This file include tskm_svcid.h file and tskm_xml_data.h file.

#include "system_service/tskm_svcid.h"
#include "system_service/tskm_xml_data.h"

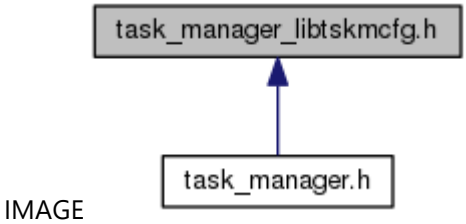
Include dependency graph for task_manager_libtskmcfg.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Detailed Description

This file include tskm_svcid.h file and tskm_xml_data.h file.

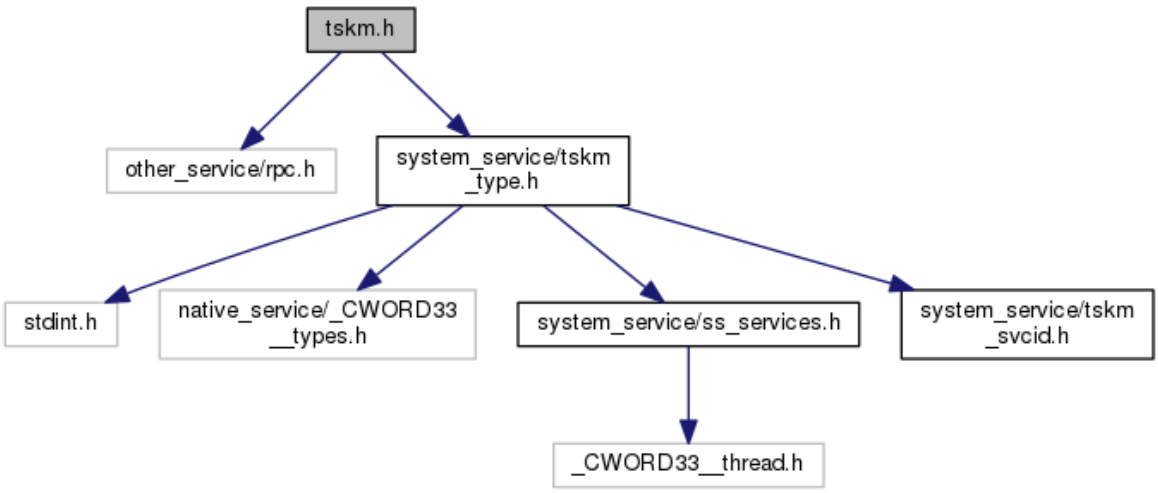
tskm.h File Reference

This file contains primary API of task manager.

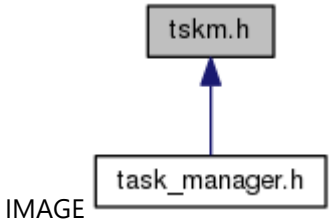
#include <other_service/rpc.h>
#include "system_service/tskm_type.h"

Include dependency graph for tskm.h:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **TSKM_ID** RPC_ID
#define TSKM_Init(p_tskmId) RPC_START_CLIENT(p_tskmId)
#define TSKM_End(tskmId) RPC_end(tskmId)

Functions

TSKM_ERR_t TSKM_SvcCtl (TSKM_SVCID_t svcId, const TSKM_SVC_CTL_t *ctl)
TSKM_ERR_t TSKM_SvcGetInfo (TSKM_SVCID_t svcId, TSKM_SVC_INFO_t *svcInfo)
TSKM_ERR_t TSKM_ErrorReboot (const TSKM_ERROR_REBOOT_t *p_info)
TSKM_ERR_t TSKM_Reboot (const TSKM_RSV_t *p_rsv)
TSKM_ERR_t TSKM_Logging (const TSKM_LOGGING_INFO_t *p_info)
TSKM_ERR_t TSKM_DataInit (HANDLE hApp, const TSKM_DATAINIT_t *p_info)
TSKM_ERR_t TSKM_SetWakeupOrder (const TSKM_WAKEUP_ORDER_t *p_order)

Detailed Description

This file contains primary API of task manager.

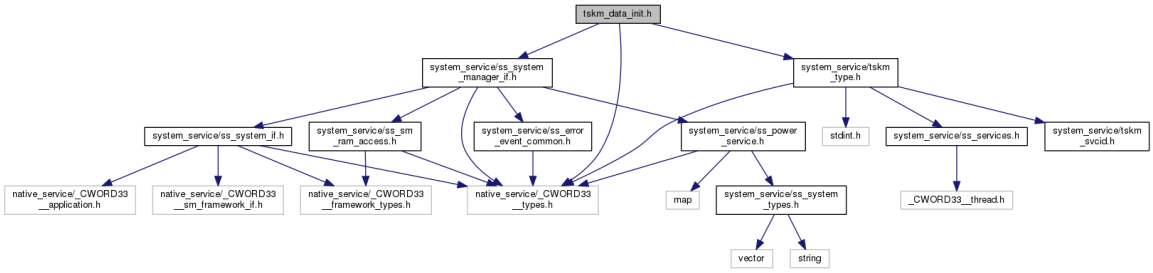
tskm_data_init.h File Reference

This file contains API of initial task manager data.

#include <native_service/_CWORD33__types.h>
#include <system_service/ss_system_manager_if.h>
#include "system_service/tskm_type.h"

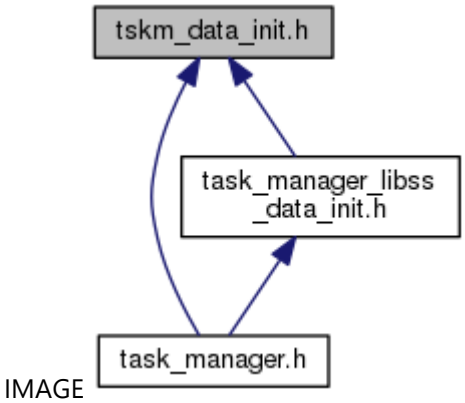
Include dependency graph for tskm_data_init.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Macros

#define TSKM_DATA_INIT_FUNC(ServiceName, ArgName) E_CWORD33_Status tskm_ ## ServiceName ## _data_init(T_SS_SM_START_DataStructType* ArgName)

Functions

TSKM_ERR_t TSKM_GetExtBootInfo (T_SS_SM_START_ExtDataStructType *p_info)

Detailed Description

This file contains API of initial task manager data.

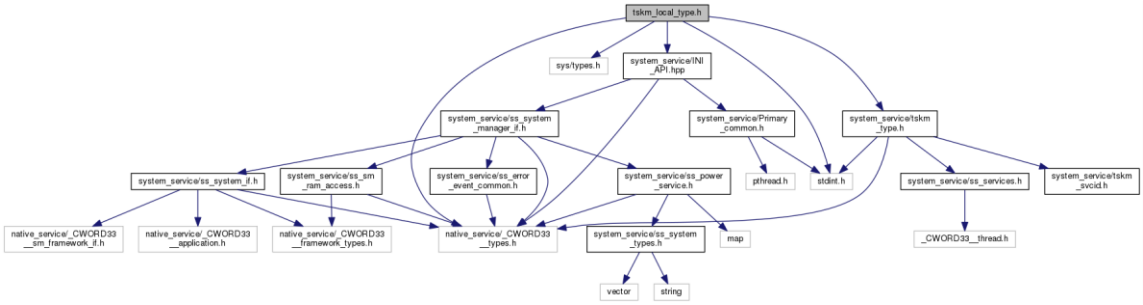
tskm_local_type.h File Reference

localtype of tskm

#include <stdint.h>
#include <sys/types.h>
#include <native_service/_CWORD33__types.h>
#include "system_service/INI_API.hpp"
#include "system_service/tskm_type.h"

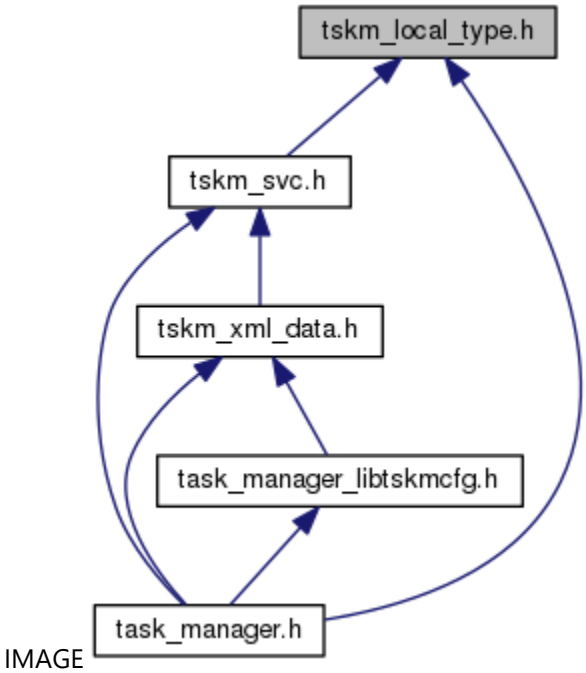
Include dependency graph for tskm_local_type.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct TSKM_GSTEP_REQ_INFO_t
struct TSKM_GSTEP_t

struct TSKM_GSTEP_CTX_t
struct TSKM_NV_HEADER_t
struct TSKM_NV_FOOTER_t
struct TSKM_NV_BODY_t
struct TSKM_NV_INFO_t
struct TSKM_EV_PRI_REQ_WAKEUP_PRM_t
struct TSKM_EV_PRI_REQ_DOWN_PRM_t
struct TSKM_EV_PRI_REP_WAKEUP_COMP_PRM_t
struct TSKM_EV_PRI_REP_DOWN_COMP_PRM_t
struct TSKM_EV_PRI_REP_CONNECT_PRM_t
struct TSKM_EV_PRI_RES_WAKEUP_PRM_t
struct TSKM_EV_PRI_RES_DOWN_PRM_t
struct TSKM_EV_PRI_EX_RES_DEBUGDUMP_PRM_t
struct TSKM_EV_LCL_CHG_SVC_STATE_PRM_t
struct _TSKM_EVENT_INFO_t
union _TSKM_EVENT_INFO_t.prm

Macros

#define **TSKM_CFG_WAIT_SHUTDOWN** 10000
#define **TSKM_CFG_TOUCH_TIMEOUT** 2
#define **TSKM_ST_ACCOFF** 0x01000000U
#define **TSKM_ST_ACCON** 0x02000000U
#define **TSKM_ST_WAKEUP** 0x02010000U
#define **TSKM_ST_RUNNING** 0x02020000U
#define **TSKM_ST_DOWN** 0x02040000U
#define **TSKM_STATE_MASK0** 0xFF000000U
#define **TSKM_STATE_MASK1** 0xFFFF0000U
#define **TSKM_STATE_MASK2** 0xFFFFFF00U
#define **TSKM_STATE_MASK3** 0xFFFFFFFFU
#define **TSKM_GSTEP_NONE** 0x00000000U
#define **TSKM_GSTEP_BUPCHK** 0x00000001U
#define **TSKM_LSTEP_LAST** 0xFFFFFFF0U
#define **TSKM_LSTEP_SHM** 0xFFFFFFF1U
#define **TSKM_LSTEP_BUPCHK** 0xFFFFFFF2U
#define **TSKM_LSTEP_ALL** 0xFFFFFFF3U
#define **TSKM_NV_STRUCT_VERSION** "v000"
#define **TSKM_SVC_RESERVE_MAX** (8)
#define **TSKM_NV_SIZE_ALL** (512)
#define **TSKM_NV_SIZE_HEADER** (32)
#define **TSKM_NV_SIZE_FOOTER** (32)
#define **TSKM_NV_SIZE_BODY** (TSKM_NV_SIZE_ALL - TSKM_NV_SIZE_HEADER - TSKM_NV_SIZE_FOOTER)
#define **TSKM_EV_BOOTINFO_SIZE** sizeof(T_SS_SM_START_DataStructType)
#define **TSKM_EV_EXTBOOTINFO_SIZE** sizeof(T_SS_SM_START_ExtDataStructType)
#define **TSKM_EV_DEBUGDUMP_SIZE** 4096

Typedefs

typedef uint32_t **TSKM_STATE_t**

typedef uint32_t **TSKM_GLOBAL_STEP_t**
typedef uint32_t **TSKM_LOCAL_STEP_t**
typedef struct _TSKM_EVENT_INFO_t **TSKM_EVENT_INFO_t**

Enumerations

enum **TSKM_EVENT_t** { **TSKM_EV_NOP**, **TSKM_EV_PRI_REQ_WAKEUP**,
TSKM_EV_PRI_REQ_DOWN, **TSKM_EV_PRI_REQ_TOUCH**,
TSKM_EV_PRI_REQ_DEBUGDUMP, **TSKM_EV_PRI_REP_LOWMEM**,
TSKM_EV_PRI_REP_WAKEUP_COMP, **TSKM_EV_PRI_REP_DOWN_COMP**,
TSKM_EV_PRI_REP_CONNECT, **TSKM_EV_PRI_REP_DISCONNECT**,
TSKM_EV_PRI_RES_WAKEUP, **TSKM_EV_PRI_RES_DOWN**,
TSKM_EV_PRI_RES_DEBUGDUMP, **TSKM_EV_PRI_REQ_EXIT**, **TSKM_EV_SVC_REP_TERM**,
TSKM_EV_API_REQ_REBOOT, **TSKM_EV_API_REQ_SVC_CTL**,
TSKM_EV_API_REQ_SVC_DISABLE, **TSKM_EV_LCL_REQ_STOP**,
TSKM_EV_LCL_CHG_SVC_STATE, **TSKM_EV_LCL_REQ_SDUMP**,
TSKM_EV_LCL_REP_TIMEOUT, **TSKM_EV_LCL_REP_POLLING**,
TSKM_EV_LCL_REP_LOWMEM, **TSKM_EV_LCL_REQ_TRANS_STEP** }
enum **TSKM_PROTCOL_t** { **TSKM_DATAINIT_REQ** = ***, **TSKM_DATAINIT_RESP**,
TSKM_TRANS_STEP_REQ }

Detailed Description

localtype of tskm

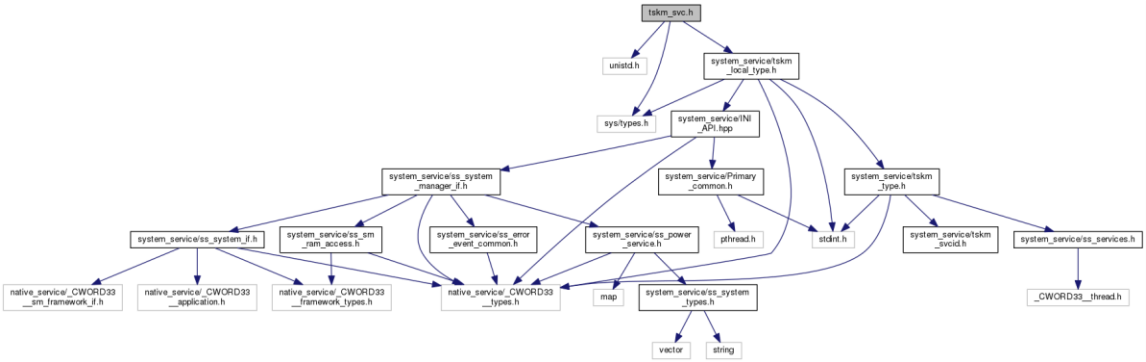
tskm_svc.h File Reference

define of tskm

#include <unistd.h>
#include <sys/types.h>
#include "system_service/tskm_local_type.h"

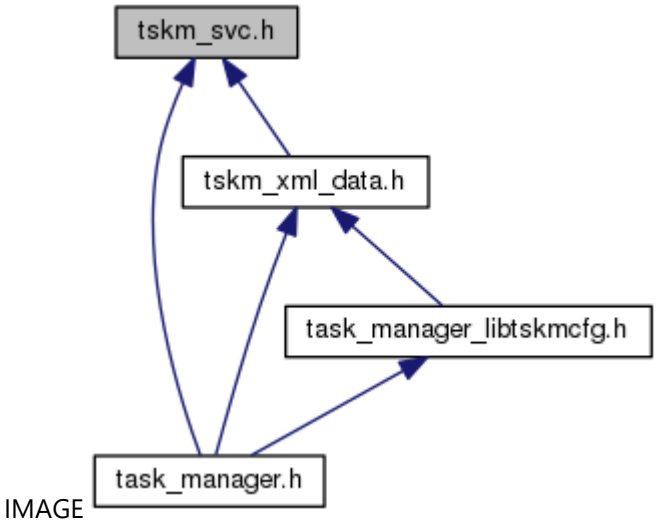
Include dependency graph for tskm_svc.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct TSKM_SVC_ATTR_t
struct TSKM_SVC_CTX_t
struct TSKM_SVCS_CTX_t

Macros

#define **TSKM_SVC_WAIT_REQ_MAX** (8)

Enumerations

enum **TSKM_SVC_TYPE_t** { **TSKM_SVC_TYPE_NATIVE**, **TSKM_SVC_TYPE_UNKNONW** }
enum **TSKM_SVC_POLICY_t** { **TSKM_SVC_POLICY_TSS**, **TSKM_SVC_POLICY_RR**, **TSKM_SVC_POLICY_FIFO** }
enum **TSKM_SVC_LC_t** { **TSKM_SVC_LC_ALWAYS**, **TSKM_SVC_LC_ALWAYS_RECOVERABLE**, **TSKM_SVC_LC_DYNAMIC** }
enum **TSKM_SVC_ASSIGN_CPU_t** { **TSKM_SVC_ASSIGN_CPU_AUTO** = ***, **TSKM_SVC_ASSIGN_CPU_0** = ***, **TSKM_SVC_ASSIGN_CPU_1** = *** }
enum **TSKM_SVC_STATE_t** { **TSKM_SVC_DORMANT**, **TSKM_SVC_WAITCONNECT**, **TSKM_SVC_WAKEUP**, **TSKM_SVC_RUNNING**, **TSKM_SVC_DOWN**, **TSKM_SVC_FINDOWN**, **TSKM_SVC_DISABLE** }
enum **TSKM_SVC_WAIT_STATE_t** { **TSKM_SVC_WAIT_NONE**, **TSKM_SVC_WAIT_TRANSIENT**, **TSKM_SVC_WAIT_BOTH** }

Functions

TSKM_ERR_t tskm_svcsEventHandle (TSKM_SVCS_CTX_t *p_svcs, const TSKM_EVENT_INFO_t *p_inEv, TSKM_EVENT_INFO_t *p_outEv)
TSKM_SVC_CTX_t * tskm_svcsGetSvcBySvcId (TSKM_SVCS_CTX_t *p_svcs, TSKM_SVCID_t svcId)
TSKM_SVC_CTX_t * tskm_svcsGetSvcByPid (TSKM_SVCS_CTX_t *p_svcs, pid_t pid)
TSKM_BOOL_t tskm_svcsIsWaiting (TSKM_SVCS_CTX_t *p_svcs)
TSKM_SVC_WAIT_STATE_t tskm_svcsGetSvcTermWaitState (TSKM_SVCS_CTX_t *p_svcs)
TSKM_ERR_t tskm_svcsSetBootInfo (TSKM_SVCS_CTX_t *p_svcs, T_SS_SM_START_DataStructType *p_info, T_SS_SM_START_ExtDataStructType *p_exInfo)
TSKM_ERR_t tskm_svcsAvtiveSvcTerm (TSKM_SVCS_CTX_t *p_svcs)
TSKM_ERR_t tskm_svcsCallDebugDump (TSKM_SVCS_CTX_t *p_svcs)
TSKM_ERR_t tskm_svcsCallLowMem (TSKM_SVCS_CTX_t *p_svcs)
TSKM_ERR_t tskm_svcEventHandle (TSKM_SVC_CTX_t *p_svc, TSKM_EVENT_INFO_t *p_ev)
TSKM_ERR_t tskm_svcExec (TSKM_SVC_CTX_t *p_svc)
TSKM_ERR_t tskm_svcWakeupRequest (TSKM_SVC_CTX_t *p_svc, TSKM_GSTEP_REQ_INFO_t *p_req)
TSKM_ERR_t tskm_svcDownRequest (TSKM_SVC_CTX_t *p_svc, TSKM_GSTEP_REQ_INFO_t *p_req)
TSKM_ERR_t tskm_svcDisableRequest (TSKM_SVC_CTX_t *p_svc)
TSKM_ERR_t tskm_svcEnableRequest (TSKM_SVC_CTX_t *p_svc)
TSKM_BOOL_t tskm_svcIsCommnicatable (TSKM_SVC_CTX_t *p_svc)

Detailed Description

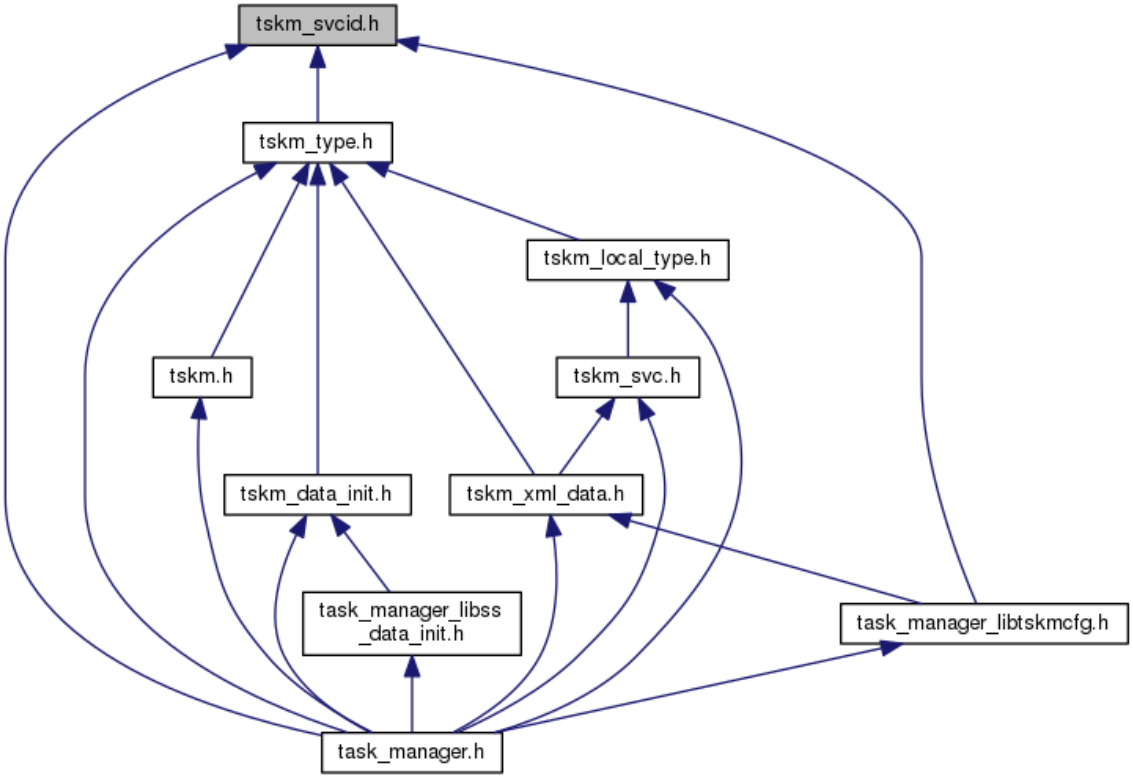
define of tskm

tskm_svcid.h File Reference

This file contains declaration of TSKM_SVC ID.

This graph shows which files directly or indirectly include this file:

IMAGE



Macros

#define **TSKM_SVCID_NONE** 0x00000000U
#define **TSKM_SVCID_TE_RESIDENT** 0x00000001U
#define **TSKM_SVCID_TE_TRANSIENT** 0x00000002U
#define **TSKM_SVCID_ACTIVITYMGR** 0x00000003U
#define **TSKM_SVCID_SYSVUP** 0x00000004U
#define **TSKM_SVCID__CWORD52_VUP** 0x00000005U
#define **TSKM_SVCID_ROOTFSVUP** 0x00000006U
#define **TSKM_SVCID_VUPPROGUI** 0x00000007U
#define **TSKM_SVCID_NORVUP** 0x00000008U
#define **TSKM_SVCID__CWORD58_** 0x00000009U
#define **TSKM_SVCID_SYSUPDATE** 0x0000000AU
#define **TSKM_SVCID_NANDUPDATE** 0x0000000BU
#define **TSKM_SVCID_BTPHONESRV** 0x0000000CU
#define **TSKM_SVCID_BTPBKSRV** 0x0000000DU
#define **TSKM_SVCID_BTMSGSRV** 0x0000000EU
#define **TSKM_SVCID_DTVVUPSRV** 0x0000000FU

#define **TSKM_SVCID_XMVUPSRV** 0x00000020U

Detailed Description

This file contains declaration of TSKM_SVC ID.

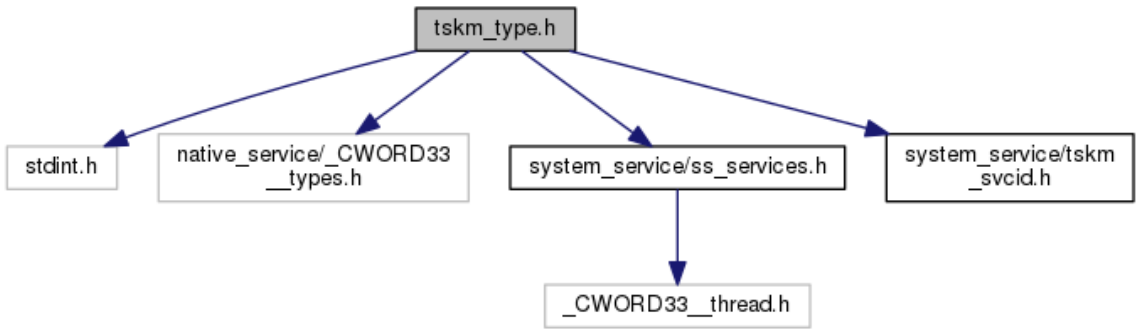
tskm_type.h File Reference

type of tskm

#include <stdint.h>
#include <native_service/_CWORD33__types.h>
#include <system_service/ss_services.h>
#include "system_service/tskm_svcid.h"

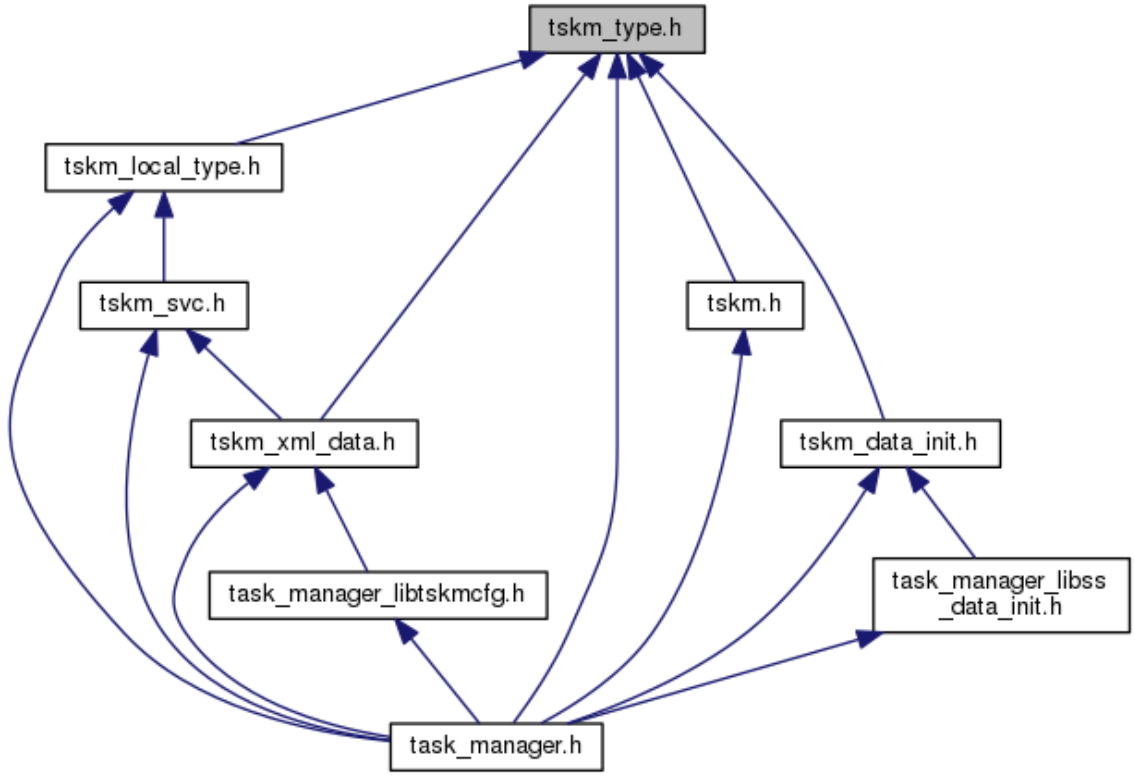
Include dependency graph for tskm_type.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct TSKM_SVC_CTL_t
struct TSKM_SVC_INFO_t
struct TSKM_LOGGING_INFO_t
struct TSKM_ERROR_REBOOT_t
struct TSKM_DATAINIT_t
struct TSKM_WAKEUP_ORDER_t

Macros

#define **TSKM_ERR_t** int32_t
#define **TSKM_E_OK** 0
#define **TSKM_E_PAR** 1
#define **TSKM_E_STATE** 2
#define **TSKM_E_PERMIT** 3
#define **TSKM_E_ALREADY** 4
#define **TSKM_E_RETRY** 5
#define **TSKM_E_NG** 6
#define **TSKM_BOOL_t** int32_t
#define **TSKM_TRUE** 1
#define **TSKM_FALSE** 0
#define **TSKM_RSV_t** int
#define **TSKM_LOGGING_MSG_STR_SIZE** 64
#define **TSKM_ORDER_NAME_MAX** 32

Typedefs

typedef uint32_t **TSKM_SVCID_t**

Enumerations

enum **TSKM_SVC_CMD_t** { **TSKM_SVC_CMD_NOP**, **TSKM_SVC_CMD_EXEC**,
TSKM_SVC_CMD_ENABLE, **TSKM_SVC_CMD_DISABLE**, **TSKM_SVC_CMD_RESERVE** }
enum **TSKM_BOOT_t** { **TSKM_BOOT_NORMAL**, **TSKM_BOOT_FACTRESET**,
TSKM_BOOT_USERRESET, **TSKM_BOOT_VERSIONUP** }
enum **TSKM_LOGGING_TYPE_NORMAL_t** { **TSKM_LOGGING_TYPE_MODULE_LOGS**,
TSKM_LOGGING_TYPE_GRP_RELAUNCH }
enum **TSKM_ERROR_REBOOT_TYPE_t** { **TSKM_ERROR_REBOOT_NORMAL** }
enum **TSKM_DATAINIT_TYPE_t** { **TSKM_DATAINIT_TYPE_USER** }

Detailed Description

type of tskm

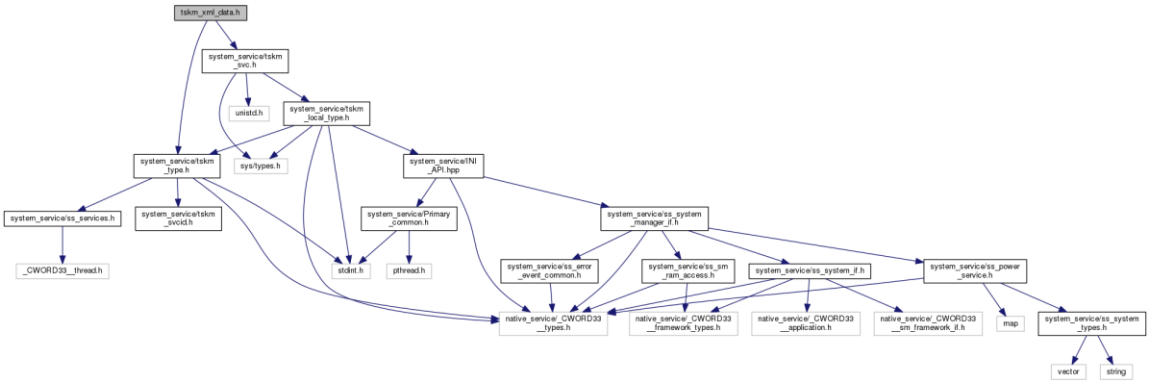
tskm_xml_data.h File Reference

This file provide api to operating task manager with XML data.

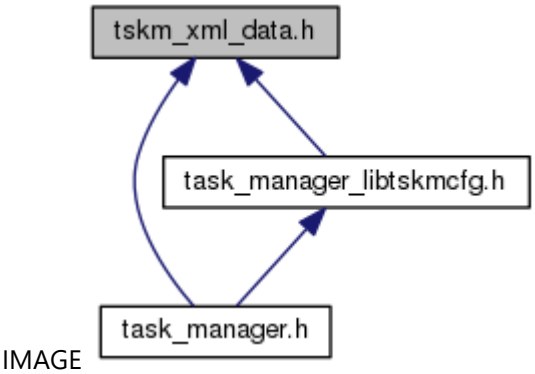
#include "system_service/tskm_type.h"
#include "system_service/tskm_svc.h"

Include dependency graph for tskm_xml_data.h:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Functions

int tskm_initServiceList (TSKM_SVCS_CTX_t *p_svcs, int iFd)
void tskm_initWakeupCtx (TSKM_GSTEP_CTX_t *p_wakeup, BOOL isVupMode)
void tskm_initDownCtx (TSKM_GSTEP_CTX_t *p_down, BOOL isVupMode)

Detailed Description

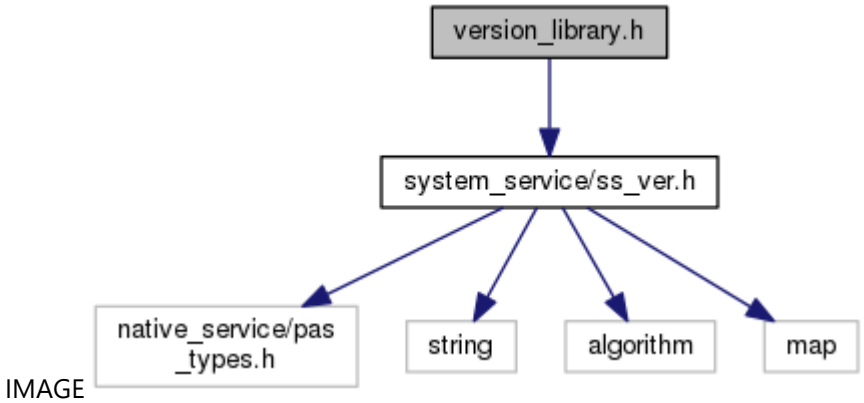
This file provide api to operating task manager with XML data.

version_library.h File Reference

This file include ss_ver.h.

#include "system_service/ss_ver.h"

Include dependency graph for version_library.h:



IMAGE

Detailed Description

This file include ss_ver.h.