Native Service Framework

API specification

Note/
CWORDXX is mask word for confidential information.

2019.10.17
TOYOTA MOTOR CORPRATION

Module Documentation

BaseSystem

Native_service

Detailed Description

Native_service

Framework_unified
_CWORD33_log_library
Backup_manager
Common_library
Notification_persistent_service

Detailed Description

Framework_unified

CWORD77
Framework
Native
Utility
Nsrcs

Namespaces

CWORD33

Classes

class IConfigReader
IConfigReader. class IConfigWriter
Config writer abstract class. class CXMLReader
This file contains declaration of class CXMLReader. class CXMLWriter
This file contains declaration of class CXMLWriter. class CXmlNode
This class represents node of an XML. class CXmlAttr
This class represents the attribute of an xml node. class CXmlParser
This class represents the XML parser. **Macros**
#define _BUILD_VERSION ""

#define WARNING_STRING "\n\!!! WARNING - Build Versions do not match !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n\project build: %s\n\library build: %s\n\environment build: %s\n\!!"
#define BUILDVERSIONLOG(__log_string__, ...) TEXT(__PRETTY_FUNCTION__, __log_string__, __VA_ARGS__)
#define BUILDVERSION_NULL_STATEMENT ((void)0)
#define CHECK_BUILD_VERSION()
#define **__CWORD33__NATIVESERVICES_NSCONFIGPARSER_INC_CONFIG_READER_H__**
#define **__CWORD33__NATIVESERVICES_NSCONFIGPARSER_INC_CONFIG_WRITER_H__**

Typedefs

typedef enum _E_CWORD33_XmlNodeTypes E_CWORD33_XmlNodeTypes

Include Files.

typedef std::list< CXmlNode > **TNodeList**
typedef TNodeList::iterator **TNodeListIterator**
typedef std::list< CXmlAttr > **TAttrList**
typedef std::list< CXmlAttr >::iterator **TAttrListIterator**

Enumerations

enum _E_CWORD33_XmlNodeTypes { _CWORD33__XML_NODE_NONE = ***, _CWORD33__XML_ELEMENT_NODE = ***, _CWORD33__XML_ATTRIBUTE_NODE = ***, _CWORD33__XML_TEXT_NODE = ***, _CWORD33__XML_CDATA_SECTION_NODE = ***, _CWORD33__XML_ENTITY_REF_NODE = ***, _CWORD33__XML_ENTITY_NODE = ***, _CWORD33__XML_PI_NODE = ***, _CWORD33__XML_COMMENT_NODE = ***, _CWORD33__XML_DOCUMENT_NODE = ***, _CWORD33__XML_DOCUMENT_TYPE_NODE = ***, _CWORD33__XML_DOCUMENT_FRAG_NODE = ***, _CWORD33__XML_NOTATION_NODE = ***, _CWORD33__XML_HTML_DOCUMENT_NODE = ***, _CWORD33__XML_DTD_NODE = ***, _CWORD33__XML_ELEMENT_DECL = ***, _CWORD33__XML_ATTRIBUTE_DECL = ***, _CWORD33__XML_ENTITY_DECL = ***, _CWORD33__XML_NAMESPACE_DECL = ***, _CWORD33__XML_XINCLUDE_START = ***, _CWORD33__XML_XINCLUDE_END = ***, _CWORD33__XML_DOCB_DOCUMENT_NODE = *** }*Include Files.*

Functions

CXMLReader * GetCXMLReaderObject (CHAR *f_cFilePath)
CXMLWriter * GetCXMLWriterObject (CHAR *f_cFilePath)
CXMLWriter * GetCXMLWriterObjectNoParam ()

Detailed Description

Macro Definition Documentation

#define _BUILD_VERSION ""

define the macro _BUILD_VERSION

#define BUILDVERSION_NULL_STATEMENT ((void)0)

define the macro BUILDVERSION_NULL_STATEMENT

#define BUILDVERSIONLOG(__log_string__, ...) TEXT(__PRETTY_FUNCTION__, __log_string__, __VA_ARGS__)

define the macro _BUILD_VERSION

#define CHECK_BUILD_VERSION()

Value:(! _CWORD33_::ns::utility::buildVersionsMatch(_BUILD_VERSION)) ? \
BUILDVERSIONLOG(WARNING_STRING, \
CWORD33::ns::utility::utility_private::niceBuildVersion(_BUILD_VERSION), \
CWORD33::ns::utility::utility_private::niceBuildVersion(_CWORD33_::ns::utility::getLibraryBuildVersion()), \
CWORD33::ns::utility::utility_private::niceBuildVersion(_CWORD33_::ns::utility::getEnvironmentBuildVersion())) :\
BUILDVERSION_NULL_STATEMENT

define the macro CHECK_BUILD_VERSION to check the version

#define WARNING_STRING "\n\!!! WARNING - Build Versions do not match !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n\project build: %s\n\library build: %s\n\environment build: %s\n\!!!"

define the macro WARNING_STRING to print the warning

Typedef Documentation

typedef enum _E_CWORD33_XmlNodeTypes E_CWORD33_XmlNodeTypes

Include Files.

type of xml nodes

Enumeration Type Documentation

enum _E_CWORD33_XmlNodeTypes

Include Files.

type of xml nodes

Enumerator

_CWORD33__XML_NODE_NONE not exist node

_CWORD33__XML_ELEMENT_NODE element node

_CWORD33__XML_ATTRIBUTE_NODE attribute node

_CWORD33__XML_TEXT_NODE text node

_CWORD33__XML_CDATA_SECTION_NODE section node

_CWORD33__XML_ENTITY_REF_NODE entity ref node

_CWORD33__XML_ENTITY_NODE entity node

_CWORD33__XML_PI_NODE PI node.

_CWORD33__XML_COMMENT_NODE comment node

_CWORD33__XML_DOCUMENT_NODE document node

_CWORD33__XML_DOCUMENT_TYPE_NODE document type node

_CWORD33__XML_DOCUMENT_FRAG_NODE document frag node

_CWORD33__XML_NOTATION_NODE notation node

_CWORD33__XML_HTML_DOCUMENT_NODE document node

_CWORD33__XML_DTD_NODE dtd node

_CWORD33__XML_ELEMENT_DECL element decl

_CWORD33__XML_ATTRIBUTE_DECL attribute decl

_CWORD33__XML_ENTITY_DECL entity decl

_CWORD33__XML_NAMESPACE_DECL namespace decl

_CWORD33__XML_XINCLUDE_START include start

_CWORD33__XML_XINCLUDE_END include end

_CWORD33__XML_DOCB_DOCUMENT_NODE document node

Function Documentation

CXMLReader* GetCXMLReaderObject (CHAR * *f_cFilePath*)

Brief

Get CXMLReader object.

Parameters:

[IN]	f_cFilePath std::string - path of file to parse

Return values:

CXMLReader	*

Prerequisite

Load the shared library first.

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

None

CXMLWriter* GetCXMLWriterObject (CHAR * *f_cFilePath*)

Brief

Get CXMLWriter object.

Parameters:

[IN]	f_cFilePath std::string - path of file to parse

Return values:

CXMLWriter	*

Prerequisite

Load the shared library first.

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

None

CXMLWriter* GetCXMLWriterObjectNoParam ()

Brief

Get CXMLWriter object.

Parameters:

None	

Return values:

CXMLWriter	*

Prerequisite

Load the shared library first.

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

None

CWORD77

class C_CWORD33__CWORD77_Service
class C_CWORD33__CWORD77_Session
class CCallback< C, M >
class CResCallback< C, M >
struct __CWORD77_Data

CWORD77 Data structure *More...*

class C_CWORD77_DataPool
class C_CWORD77_Data

Macros

#define **MAX_DATA_SIZE** 512;
#define **DELETEPTR**(p) {if (p != NULL) {delete p; p = NULL;}}
#define _CWORD77__DATA_POOL_KEY(ProtocolId) (std::make_pair(ProtocolId, g_c_CWORD77_ServiceName))

Typedefs

typedef E_CWORD33_Status(* **_CWORD77_FuncPtr**) (HANDLE hApp, BOOL bTImerExpiry)
typedef E_CWORD33_Status(* **ResponseServiceTo_CWORD77_**) (HANDLE hApp, E_CWORD77_ResponseType ResponseType, std::string ServiceName)
typedef boost::function< E_CWORD33_Status(HANDLE, std::string, std::string, UI_32, E_CWORD33_Status) > **SessionAckTo_CWORD77_**

Functions

VOID SetReqDataIn_CWORD77_DataPool (const _CWORD77_DataPoolKey &f_DataPoolKey, UI_32 f_uiSize, PVOID f_pData)
E_CWORD33_Status GetRespDataFrom_CWORD77_DataPool (const _CWORD77_DataPoolKey &f_DataPoolKey, UI_32 &f_uiSize, PVOID &f_pData)
VOID SetRespNotfnDataIn_CWORD77_DataPool (const std::string &f_cNotificationName, UI_32 f_uiSize, const PVOID f_pData)
E_CWORD33_Status GetRespNotfnDataFrom_CWORD77_DataPool (const std::string &f_cNotificationName, UI_32 &f_uiSize, PVOID &f_pData)

enum _ETableType { **REQUEST_TABLE** = ***, **RESPONSE_TABLE** }*Enumeration for Type of Table.*

enum _EDataType { **UNKNOWNTYPE** = ***, **SIGNEDINT**, **UNSIGNEDINT**, **FLOAT**, **DOUBLE**, **CHARACTER**, **BOOLEAN**, **STRING** }*Enumeration for Data Type.*

typedef enum _ETableType ETableType

Enumeration for Type of Table.

typedef enum _ETableType * **PETableType**
typedef enum _EDataType EDataType

Enumeration for Data Type.

typedef enum _EDataType * **PEDataType**
typedef struct __CWORD77_Data _CWORD77_Data

CWORD77_Data structure

typedef struct __CWORD77_Data * **P_CWORD77_Data**
typedef std::map< _CWORD77_DataPoolKey, std::vector< CHAR > > **T_CWORD77_DataPool**
C_CWORD77_DataPool **g__CWORD77_DataPool**
UI_32 NoOfElementsinData (std::string Input)
template<typename T > std::string NumberToString (T Number)

Template to Convert Number to String.

template<typename T > T StringToNumber (const std::string &Text)

Template to convert String to Number.

template<class T > std::string ConvertArrayToString (T *Array, int Arraysize)

Template to Convert an Array To String.

template<class T > void ConvertStringToArray (std::string Input, std::vector< T > &Array, UI_32 &Arraysize)

Template to Convert String To Array.

std::string ConvertArrayStringsToString (std::string *strArr, SI_32 Size)
void ConvertStringToArrString (std::string Input, std::string *strArr, UI_32 &ArraySize)

template<class T > void SetRequestArrayData (UI_32 VarName, EDataType DataType, T *Array, UI_32 ArraySize)
template<class T > void SetResponseArrayData (UI_32 VarName, EDataType DataType, T *Array, UI_32 ArraySize)
template<class T > void GetRequestArrayData (UI_32 VarName, T *Array, UI_32 &ArraySize)
template<class T > void GetResponseArrayData (UI_32 VarName, T *Array, UI_32 &ArraySize)
template<class T > void SetRequestData (UI_32 VarName, EDataType DataType, T Array)
template<class T > void SetResponseData (UI_32 VarName, EDataType DataType, T Array)
template<class T > T GetRequestData (UI_32 VarName)
template<class T > T GetResponseData (UI_32 VarName)
void SetRequestArrayStringData (UI_32 VarName, EDataType VarType, std::string DataValue[], UI_32 size)
void SetResponseArrayStringData (UI_32 VarName, EDataType VarType, std::string DataValue[], UI_32 size)
void GetRequestArrayStringData (UI_32 VarName, std::string DataValue[], UI_32 &size)
void GetResponseArrayStringData (UI_32 V arName, std::string DataValue[], UI_32 &size)
void SetRequestStringData (UI_32 VarName, EDataType VarType, std::string DataValue)
void SetResponseStringData (UI_32 VarName, EDataType VarType, std::string DataValue)
std::string GetRequestStringData (UI_32 VarName)
UI_32 GetRequestDataLength (UI_32 Key)
UI_32 GetResponseDataLength (UI_32 Key)
EDataType GetRequestDataType (UI_32 Key)
EDataType GetResponseDataType (UI_32 Key)
#define NTFY_REG__CWORD77__Available "_CWORD77_/Available"

CWORD77 Service Availability Notification

enum **_E_CWORD77_ResponseType** { **RESPONSE** = ***, **NOTIFICATION**, **EVENT** }
typedef enum _E_CWORD77_ResponseType **E_CWORD77_ResponseType**
typedef enum _E_CWORD77_ResponseType * **PE_CWORD77_ResponseType**
typedef std::pair< UI_32, std::string > **_CWORD77_DataPoolKey**

Detailed Description

Class Documentation

struct __CWORD77_Data

CWORD77 Data structure

Class Members:

EDataType	DataType	
string	DataValue	

Macro Definition Documentation

#define _CWORD77__DATA_POOL_KEY(ProtocolId) (std::make_pair(ProtocolId, g_c_CWORD77_ServiceName))

Function Documentation

std::string ConvertArrayStringsToString (std::string * *strArr*, SI_32 *Size*)

ConvertArrayStringsToString Function to convert array of strings to string

Parameters:

in	*strArr*	string * - pointer to array of strings
in	*Size*	SI_32 - Number of strings

Returns:

string

void ConvertStringToArrString (std::string *Input*, std::string * *strArr*, UI_32 & *ArraySize*)

ConvertArrayStringsToString Function to convert string to array of strings

Parameters:

in	*Input*	string - string contains array of strings
out	*strArr*	string * - Pointer to Array of strings
out	*ArraySize*	UI_32 - No of elements in Array

Returns:

None

template<class T > void GetRequestArrayData (UI_32 *VarName*, T * *Array*, UI_32 & *ArraySize*)

GetRequestArrayData Templatized Function to get array data from Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
out	*Array*	T * - Pointer to an array of given type
out	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

Here is the call graph for this function:

IMAGE

void GetRequestArrayStringData (UI_32 *VarName*, std::string *DataValue*[], UI_32 & *size*)

GetRequestArrayStringData API to get array of string data from Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
out	*DataValue*	string [] - array of strings
out	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

template<class T > T GetRequestData (UI_32 *VarName*)

GetRequestData Templatized Function to get data from Resquest *CWORD77* Data base

Parameters:

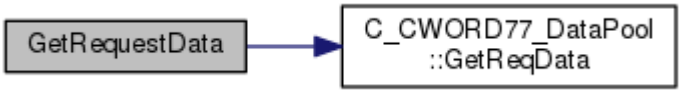
in	*VarName*	UI_32 - key

Returns:

T - value belong to type T

Here is the call graph for this function:

IMAGE



UI_32 GetRequestDataLength (UI_32 *Key*)

GetRequestDataLength API to get No of elements in a key from request data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

UI_32 - No of elements

EDataType GetRequestDataType (UI_32 *Key*)

GetDataTypeFromRequestTable API to get data type of key from request data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

EDataType - Data type of key

std::string GetRequestStringData (UI_32 *VarName*)

GetRequestStringData API to get string data from Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

string - value of data

E_CWORD33_Status GetRespDataFrom_CWORD77_DataPool (const _CWORD77_DataPoolKey & *f_DataPoolKey*, UI_32 & *f_uiSize*, PVOID & *f_pData*)

Brief

API to get data associated with response from *CWORD77* Data Pool

Parameters:

in	*f_DataPoolKey*	_CWORD77_DataPoolKey - Data pool key
out	*f_uiSize*	UI_32 - size of the data
out	*f_pData*	PVOID - void pointer to data

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Error

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

Set Get

See also:

SetRespNotfnDataFrom_CWORD77_DataPool, C_CWORD77_Data::GetRespoData

E_CWORD33_Status GetRespNotfnDataFrom_CWORD77_DataPool (const std::string & *f_cNotificationName*, UI_32 & *f_uiSize*, PVOID & *f_pData*)

Brief

API to set data associated with notification response into *CWORD77* Data Pool

Parameters:

in	*f_cNotificationName*	const std::string& - Name of the notification
out	*f_uiSize*	UI_32& - size of the data

out	*f_pData*	PVOID& - void pointer to data

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Fail

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

Set Get

See also:

SetRespNotfnDataFrom_CWORD77_DataPool, C_CWORD77_Data::GetRespoData

template<class T > void GetResponseArrayData (UI_32 *VarName*, T * *Array*, UI_32 & *ArraySize*)

GetRequestArrayData Templatized Function to get array data from Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
out	*Array*	T * - Pointer to an array of given type
out	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

Here is the call graph for this function:

IMAGE

void GetResponseArrayStringData (UI_32 *VarName*, std::string *DataValue*[], UI_32 & *size*)

GetResponseArrayStringData API to get array of string data from Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
out	*DataValue*	string [] - array of strings
out	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

template<class T > T GetResponseData (UI_32 *VarName*)

GetResponseData Templatized Function to get data from Response *CWORD77* Data base

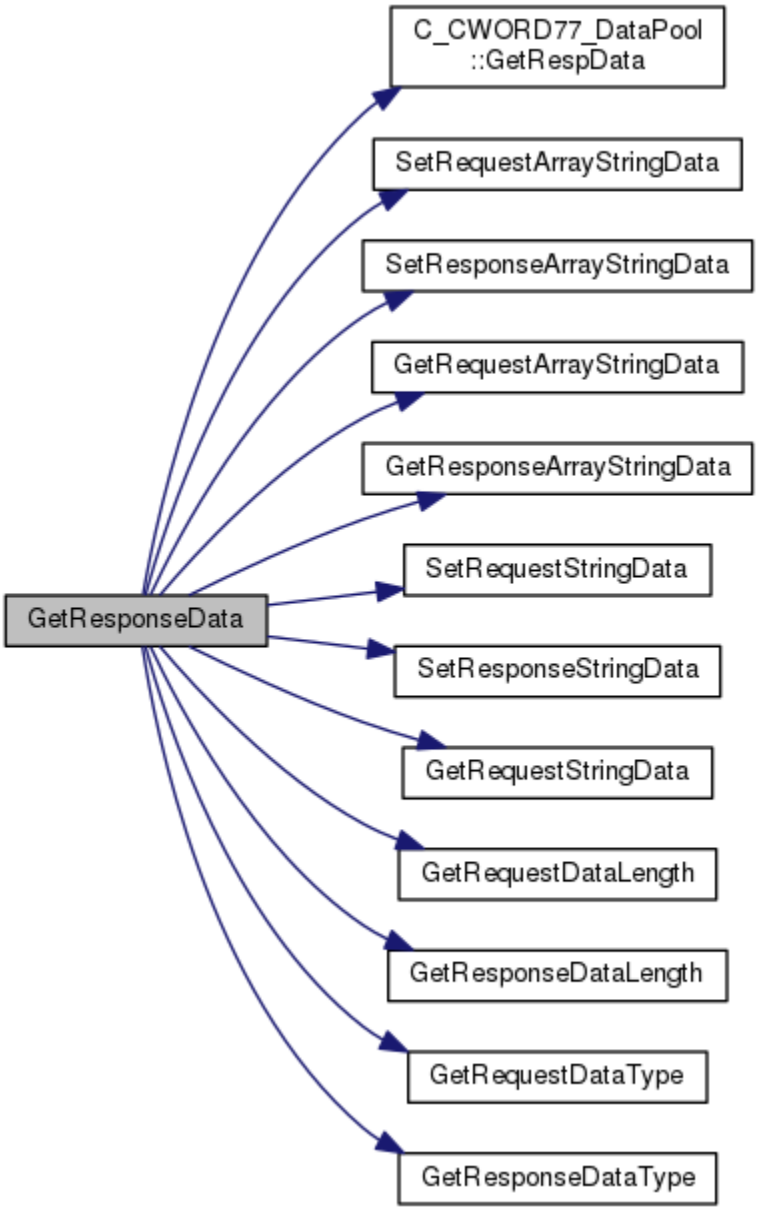
Parameters:

in	*VarName*	UI_32 - key

Returns:

T - value belong to type T

Here is the call graph for this function:



IMAGE

UI_32 GetResponseDataLength (UI_32 *Key*)

GetResponseDataLength API to get No of elements in a key from response data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

UI_32 - No of elements

EDataType GetResponseDataType (UI_32 *Key*)

GetDataTypeFromResponseTable API to get data type of key from response data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

EDataType - Data type of key

UI_32 NoOfElementsinData (std::string *Input*)

NoOfElementsinData Function to get number of elements for given string

Parameters:

in	*Input*	string - string

Returns:

UI_32 - Number of elements

VOID SetReqDataIn_CWORD77_DataPool (const _CWORD77_DataPoolKey & *f_DataPoolKey*, UI_32 *f_uiSize*, PVOID *f_pData*)

Brief

API to set data associated with request into *CWORD77* Data Pool

Parameters:

in	*f_DataPoolKey*	_CWORD77_DataPoolKey - Data pool key
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	PVOID - void pointer to data

Return values:

none	

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

Set Get

See also:

GetRespDataFrom_CWORD77_DataPool, C_CWORD77_Data::SetRespoData

template<class T > void SetRequestArrayData (UI_32 *VarName*, EDataType *DataType*, T * *Array*, UI_32 *ArraySize*)

SetRequestArrayData Templatized Function to set array data into Response *CWORD77* Data base

Parameters:

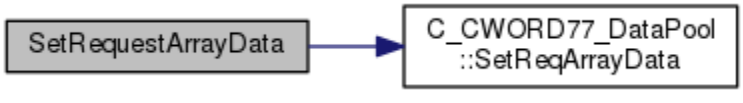
in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*Array*	T * - Pointer to an array of given type
in	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

Here is the call graph for this function:

IMAGE



void SetRequestArrayStringData (UI_32 *VarName*, EDataType *VarType*, std::string *DataValue*[], UI_32 *size*)

SetRequestArrayStringData API to set array of string data into Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*DataValue*	string [] - array of strings
in	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

template<class T > void SetRequestData (UI_32 *VarName*, EDataType *DataType*, T *Array*)

SetRequestData Templatized Function to set data into Request *CWORD77* Data base

Parameters:

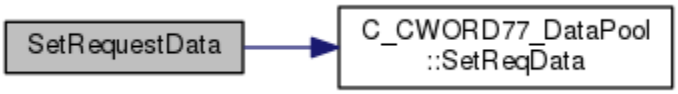
in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*Array*	T - Data type of key

Returns:

None

Here is the call graph for this function:

IMAGE



void SetRequestStringData (UI_32 *VarName*, EDataType *VarType*, std::string *DataValue*)

SetRequestStringData API to set string data into Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*DataValue*	string [] - array of strings

Returns:

None

VOID SetRespNotfnDataIn_CWORD77_DataPool (const std::string & *f_cNotificationName*, UI_32 *f_uiSize*, const PVOID *f_pData*)

Brief

API to set data associated with notification response into *CWORD77* Data Pool

Parameters:

in	*f_cNotificationName*	const std::string& - Name of the notification
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	const PVOID - void pointer to data

Return values:

none	

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

Set Get

See also:

GetRespNotfnDataFrom_CWORD77_DataPool, C_CWORD77_Data::SetRespoData

template<class T > void SetResponseArrayData (UI_32 *VarName*, EDataType *DataType*, T * *Array*, UI_32 *ArraySize*)

SetRequestArrayData Templatized Function to set array data into Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*Array*	T * - Pointer to an array of given type
in	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

Here is the call graph for this function:

IMAGE

void SetResponseArrayStringData (UI_32 *VarName*, EDataType *VarType*, std::string *DataValue*[], UI_32 *size*)

SetRequestArrayStringData API to set array of string data into Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*DataValue*	string [] - array of strings
in	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

template<class T > void SetResponseData (UI_32 *VarName*, EDataType *DataType*, T *Array*)

SetRequestData Templatized Function to set data into Response *CWORD77* Data base

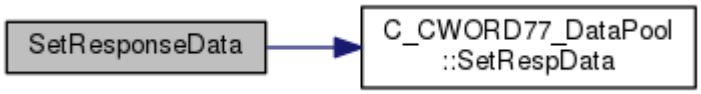
Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*Array*	T - Data type of key

Returns:

None

Here is the call graph for this function:

IMAGE 

void SetResponseStringData (UI_32 *VarName*, EDataType *VarType*, std::string *DataValue*)

SetResponseStringData API to set string data into Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*DataValue*	string [] - array of strings

Returns:

None

Framework

Statemachine

Namespaces

CWORD33

Classes

class C_CWORD33_SyncData

this file has the C_CWORD33_SyncData class definitions struct _CWORD33_DefaultCallbackHandler

struct _CWORD33_ProtocolCallbackHandler
struct _CWORD33_FdProtocolCallbackHandler
struct _CWORD33_NotificationCallbackHandler
struct _CWORD33_NotificationsList
struct _CWORD33_ProtocolEvent
struct _CWORD33_NotificationEvent

struct _ServiceAvailability
struct _ShutdownComplete
struct _CustomCommandLineOptions
struct __CWORD33_ChildThreadAttr
Child Thread Attribute. *More...*
class C_CWORD33_ServiceInterface
struct _OpenSessionAck
struct _CloseSessionReq
struct _CloseSessionAck
struct _SYSTEMTIME

Macros

#define **DEFINE_HANDLE_TYPE**(struct_type, handle_type) typedef struct struct_type* handle_type;
#define **MIN_VAL**(a, b) ((a) < (b)? (a) : (b))
#define **NULL** 0
#define **CONST** const
#define INVALID_HANDLE NULL
Lengths and Max Sizes.
#define **MAX_QUEUE_MSG_SIZE** 4096
#define **MAX_QUEUE_NAME_SIZE** 20
#define **LIMIT_QUEUE_NAME_SIZE** 16
#define **MAX_SYS_INFO_SIZE** 64
#define MAX_NAME_SIZE_APP MAX_QUEUE_NAME_SIZE
Maximum name size of the APP (Client/Service)
#define LIMIT_NAME_SIZE_APP LIMIT_QUEUE_NAME_SIZE
Limit name size of the APP (Client/Service)
#define MAX_NAME_SIZE_NOTIFICATION 24
Maximum name size of the Notification.
#define **MAX_STRING_SIZE_NOTIFICATION** 64
#define **MAX_STRING_SIZE_TAG** 64
#define **MAX_PATH_LENGTH** 255
#define **NS_SHM_ERROR** -1
#define **PFALSE** ((BOOL)0)
#define **PTRUE** ((BOOL)1)
#define **FALSE** PFALSE
#define **TRUE** PTRUE
#define **LOWORD**(l) ((WORD)(l))
#define **HIWORD**(l) ((WORD)(((DWORD)(l) >> 16) & 0xFFFF))
#define **_countof**(array) (sizeof(array)/sizeof(array[0]))
#define **MAJORNO** 0x02
#define **MINORNO** 0x02
#define **REVISION** 0x05
#define **CALLBACK**
#define **INVALID_HANDLE_VALUE** ((void*)(-1))
#define **INFINITE** 0xFFFFFFFF

#define **VOID** void
#define **__readableTo**(extent)
#define **__nullterminated** __readableTo(sentinel(0))
#define **wsprintf** sprintf
#define **DECLARE_HANDLE**(name) struct name##__ { int unused; }; typedef struct name##__ *name

Typedefs

typedef void **VOID**
typedef int8_t **SI_8**
typedef uint8_t **UI_8**
typedef int16_t **SI_16**
typedef uint16_t **UI_16**
typedef int32_t **SI_32**
typedef uint32_t **UI_32**
typedef int64_t **SI_64**
typedef uint64_t **UI_64**
typedef float **F_32**
typedef double **F_64**
typedef BOOL * **PBOOL**
typedef SI_8 * **PSI_8**
typedef UI_8 * **PUI_8**
typedef SI_16 * **PSI_16**
typedef UI_16 * **PUI_16**
typedef UI_32 * **PUI_32**
typedef SI_32 * **PSI_32**
typedef UI_64 * **PUI_64**
typedef SI_64 * **PSI_64**
typedef F_32 * **PF_32**
typedef void * **PVOID**
typedef CONST void * **PCVOID**
typedef char **CHAR**
typedef char * **PCHAR**
typedef CONST char * **PCSTR**
typedef char * **PSTR**
typedef UI_8 **_TUCHAR**
typedef SI_32 **HFILE**
typedef PVOID **HANDLE**
typedef HANDLE * **PHANDLE**
typedef CHAR TSysMsgSystemInfo[MAX_SYS_INFO_SIZE]

Information passed via the framework.

typedef UI_16 MID

Types that have been carried over from Color Radio header file WPFAPI_types.h.

typedef UI_16 **_CWORD33__CID**
typedef UI_8 **RID**
typedef UI_16 **PNO**
typedef UI_8 **BYTE**

typedef enum e_CWORD33_Status E_CWORD33_Status

Status and return types.

typedef enum e_CWORD33_Status * **PE_CWORD33_Status**
typedef enum _E_CWORD33_NotificationType E_CWORD33_NotificationType

Notification types.

typedef enum _EApplicationStates **EApplicationStates**
typedef enum _E_CWORD33_MessagePriorties **E_CWORD33_MessagePriorties**
typedef enum _E_CWORD33_ClearPersistence **E_CWORD33_ClearPersistence**
typedef enum _E_CWORD33_PersistCategory E_CWORD33_PersistCategory

Enum defining category of persistent data.

typedef enum _E_CWORD33_SystemError **E_CWORD33_SystemError**
typedef enum _E_CWORD33_ShutdownType E_CWORD33_ShutdownType

defines the shutdown types

typedef enum _E_CWORD33_ReleaseType E_CWORD33_ReleaseType

defines the Release types

typedef int **INT**
typedef unsigned int **UINT**
typedef unsigned int * **PUINT**
typedef char **CHAR**
typedef short **SHORT**
typedef long **LONG**
typedef SHORT * **PSHORT**
typedef LONG * **PLONG**
typedef unsigned long **ULONG**
typedef ULONG * **PULONG**
typedef unsigned short **USHORT**
typedef USHORT * **PUSHORT**
typedef unsigned char **UCHAR**
typedef UCHAR * **PUCHAR**
typedef unsigned long **DWORD**
typedef unsigned char **BYTE**
typedef unsigned short **WORD**
typedef float **FLOAT**
typedef FLOAT * **PFLOAT**
typedef BOOL * **PBOOL**
typedef BOOL * **LPBOOL**
typedef BYTE * **PBYTE**
typedef BYTE * **LPBYTE**
typedef int * **PINT**
typedef int * **LPINT**
typedef WORD * **PWORD**
typedef WORD * **LPWORD**
typedef long * **LPLONG**
typedef DWORD * **PDWORD**
typedef DWORD * **LPDWORD**
typedef void * **PVOID**

typedef void * **LPVOID**
typedef const void * **LPCVOID**
typedef wchar_t **WCHAR**
typedef size_t **SIZE_T**
typedef DWORD **COLORREF**
typedef DWORD * **LPCOLORREF**
typedef CHAR * **PCHAR**
typedef CHAR * **LPCH**
typedef CHAR * **PCH**
typedef CONST CHAR * **LPCCH**
typedef CONST CHAR * **PCCH**
typedef CHAR * **NPSTR**
typedef CHAR * **LPSTR**
typedef CHAR * **PSTR**
typedef PSTR * **PZPSTR**
typedef CONST PSTR * **PCZPSTR**
typedef CONST CHAR * **LPCSTR**
typedef CONST CHAR * **PCSTR**
typedef PCSTR * **PZPCSTR**
typedef void * **HGDIOBJ**
typedef int **INT_PTR**
typedef int * **PINT_PTR**
typedef unsigned int **UINT_PTR**
typedef unsigned int * **PUINT_PTR**
typedef long **LONG_PTR**
typedef long * **PLONG_PTR**
typedef unsigned long **ULONG_PTR**
typedef unsigned long * **PULONG_PTR**
typedef unsigned long **ULONGLONG**
typedef char **CCHAR**
typedef DWORD **LCID**
typedef PDWORD **PLCID**
typedef WORD **LANGID**
typedef long **LONGLONG**
typedef WORD **ATOM**
typedef pthread_mutex_t **CRITICAL_SECTION**
typedef CRITICAL_SECTION * **LPCRITICAL_SECTION**
typedef __nullterminated WCHAR * **NWPSTR**
typedef __nullterminated WCHAR * **LPWSTR**
typedef __nullterminated WCHAR * **PWSTR**
typedef __nullterminated PWSTR * **PZPWSTR**
typedef __nullterminated CONST PWSTR * **PCZPWSTR**
typedef __nullterminated CONST WCHAR * **LPCWSTR**
typedef __nullterminated CONST WCHAR * **PCWSTR**
typedef __nullterminated PCWSTR * **PZPCWSTR**
typedef LPSTR **PTSTR**
typedef LPSTR **LPTSTR**
typedef LPSTR **PUTSTR**

typedef LPSTR **LPUTSTR**
typedef LPCSTR **PCTSTR**
typedef LPCSTR **LPCTSTR**
typedef LPCSTR **PCUTSTR**
typedef LPCSTR **LPCUTSTR**
typedef WCHAR * **LPWCH**
typedef WCHAR * **PWCHAR**
typedef struct _SYSTEMTIME **SYSTEMTIME**
typedef struct _SYSTEMTIME * **PSYSTEMTIME**
typedef struct _SYSTEMTIME * **LPSYSTEMTIME**

Enumerations

enum e_CWORD33_Status { e_CWORD33_StatusEmptyMediaList = ***,
e_CWORD33_StatusSessionLimitMaxedOut = ***, e_CWORD33_StatusDbRecNotFound = ***,
e_CWORD33_StatusDbResultError = ***, e_CWORD33_StatusDbExecuteFail = ***,
e_CWORD33_StatusSemCloseFail = ***, e_CWORD33_StatusSemUnlockFail = ***,
e_CWORD33_StatusSemLockFail = ***, e_CWORD33_StatusFail = ***,
e_CWORD33_StatusErrOther = ***, e_CWORD78_StatusOK = ***, e_CWORD33_StatusInvldBuf = ***, e_CWORD33_StatusInvldHandle = ***, e_CWORD33_StatusInvldHndlType = ***,
e_CWORD33_StatusInvldQName = ***, e_CWORD33_StatusMsgQFull = ***,
e_CWORD33_StatusInvldNotification = ***, e_CWORD33_StatusInvldParam = ***,
e_CWORD33_StatusInvldBufSize = ***, e_CWORD33_StatusInvldID = ***,
e_CWORD33_StatusCannotRelease = ***, e_CWORD33_StatusBadConnection = ***,
e_CWORD33_StatusExit = ***, e_CWORD33_StatusNotImplemented = ***,
e_CWORD33_StatusThreadBusy = ***, e_CWORD33_StatusThreadSelfJoin = ***,
e_CWORD78_StatusThreadInvalidVal = ***, e_CWORD33_StatusThreadNotExist = ***,
e_CWORD33_StatusFault = ***, e_CWORD33_StatusServNotFound = ***,
e_CWORD33_StatusServerInUse = ***, e_CWORD33_StatusDbIndexing = ***,
e_CWORD33_StatusNullPointer = ***, **e_CWORD33_StatusMsgNotProcessed** = ***,
e_CWORD33_StatusFileLoadSuccess = ***, e_CWORD33_StatusFileLoadError = ***,
e_CWORD33_StatusAccessError = ***, e_CWORD33_StatusDuplicate = ***,
e_CWORD33_StatusMsgQEmpty = ***, **e_CWORD33_StatusThreadAlreadyRunning** = ***,
e_CWORD33_StatusErrNoEBADF = ***, e_CWORD33_StatusErrNoEAGAIN = ***,
e_CWORD33_StatusErrNoEINTR = ***, e_CWORD33_StatusSessionErr = ***,
e_CWORD33_StatusDBCorrupt = ***, e_CWORD33_StatusDBFileNotFound = *** }
enum _E_CWORD33_NotificationType { e_CWORD33_NotificationVar = ***, e_CWORD33_StateVar,
e_CWORD33_PersistedStateVar, e_CWORD33_PersistedStateUserVar,
e_CWORD33_ImmediatePersistedStateVar, e_CWORD33_Unknown }
enum _EApplicationStates { eAppInital = ***, eAppIdle, eAppReady, eAppConnecting,
eAppDisconnecting }
enum **_E_CWORD33_MessagePriorties** { **e_CWORD33_MsgPrioVeryLow** = ***,
e_CWORD33_MsgPrioLow = ***, **e_CWORD33_MsgPrioNormal** = ***,
e_CWORD33_MsgPrioEmergency = *** }
enum _E_CWORD33_ClearPersistence { e_CWORD33_ClearAllData = ***,
e_CWORD33_ClearAllApplicationData, e_CWORD33_ClearAllNotificationData,
e_CWORD33_ClearCurrentUserData, e_CWORD33_ClearCurrentUserApplicationData,
e_CWORD33_ClearCurrentUserNotificationData }

enum _E_CWORD33_PersistCategory { **e_CWORD33_UserData** = ***, **e_CWORD33_FactoryData** = ***, **e_CWORD33_FactoryCustomerData** = ***, **e_CWORD33_DealerData** = *** }*Enum defining category of persistent data.*

enum **_E_CWORD33_SystemError** { **e_CWORD33_SystemErrorNone** = ***, **e_CWORD33_DSPHardwareReset** = *** }

enum _E_CWORD33_ShutdownType { **e_CWORD33_NormalShutdown** = ***, **e_CWORD33_QuickShutdown**, **e_CWORD33_DataResetShutdown** }*defines the shutdown types*

enum _E_CWORD33_ReleaseType { e_CWORD33_NotOnRelease = ***, e_CWORD33_PersistOnShutdown, e_CWORD33_PersistInstantly }*defines the Release types*

E_CWORD33_Status _CWORD33_OnInitialization (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnWakeup (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnShutdown (HANDLE hApp)

<< *deprecated*

E_CWORD33_Status _CWORD33_OnEShutdown (HANDLE hApp)

<< *deprecated*

E_CWORD33_Status _CWORD33_OnStart (HANDLE hApp)

<< *deprecated*

E_CWORD33_Status _CWORD33_OnStop (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnPreStart (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnPreStop (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnBackgroundStart (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnBackgroundStop (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnReinit (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnDestroy (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnDebugDump (HANDLE hApp)

PCSTR _CWORD33_GetServiceAvailabilityNotification (HANDLE hApp)

E_CWORD33_Status _CWORD33_OnDummy (HANDLE hApp)

E_CWORD33_Status _CWORD33_Dispatcher (PCSTR cAppName, const _CWORD33_DefaultCallbackHandler *CbHandler)

E_CWORD33_Status _CWORD33_DispatcherWithArguments (PCSTR cAppName, int argc, char *argv[], const _CWORD33_DefaultCallbackHandler *CbHandler, CustomCommandLineOptions *cmdLineOptions=NULL)

E_CWORD33_Status _CWORD33_SimpleDispatcher (PCSTR cAppName, const _CWORD33_DefaultCallbackHandler *CbHandler, CbFuncPtr CbShutdown, BOOL isChildThread=TRUE)

E_CWORD33_Status _CWORD33_CreateDispatcherWithoutLoop (PCSTR cAppName, HANDLE &hApp, int argc, char *argv[], const _CWORD33_DefaultCallbackHandler *CbHandler, BOOL bIsConnectSM, CustomCommandLineOptions *cmdLineOptions=NULL)

E_CWORD33_Status _CWORD33_DispatchProcessWithoutLoop (HANDLE hApp)

E_CWORD33_Status _CWORD33_DestroyDispatcherWithoutLoop (HANDLE hApp)

E_CWORD33_Status _CWORD33_GetDispatcherFD (HANDLE hApp, SI_32 *efd)

E_CWORD33_Status _CWORD33_GetDefaultCbHandler (_CWORD33_DefaultCallbackHandler *CbHandler)

E_CWORD33_Status _CWORD33_AttachLostSessionCallbackToDispatcher (HANDLE hApp, CbFuncPtr fpLostSession)

E_CWORD33_Status _CWORD33_GetLostSessionData (HANDLE hApp, PSTR pServiceName, PUI_32 puiSessionId)

#define **DEFINE__CWORD33__ERROR**(name)

E_CWORD33_Status _CWORD33_AttachCallbacksToDispatcher (HANDLE hApp, PCSTR pServiceName, const _CWORD33_ProtocolCallbackHandler *pMsgHandler, UI_32 uiHandlerCount, HANDLE hSession=NULL)

E_CWORD33_Status _CWORD33_AttachCallbackToDispatcher (HANDLE hApp, PCSTR pServiceName, UI_32 iCmd, CbFuncPtr fpOnCmd, HANDLE hSession=NULL)

E_CWORD33_Status _CWORD33_AttachCallbacksToDispatcherWithFd (HANDLE hApp, const _CWORD33_FdProtocolCallbackHandler *handlers, UI_32 handlerCount)

E_CWORD33_Status _CWORD33_AttachCallbackToDispatcherWithFd (HANDLE hApp, int fd, CbFuncPtr fpOnCmd)

E_CWORD33_Status _CWORD33_AttachParentCallbacksToDispatcher (HANDLE hChildApp, const _CWORD33_ProtocolCallbackHandler *pMsgHandler, UI_32 uiHandlerCount)

E_CWORD33_Status _CWORD33_DetachCallbacksFromDispatcher (HANDLE hApp, PCSTR pServiceName, const PUI_32 puiCmdArray, UI_32 uiCommandCount, HANDLE hSession=NULL)

E_CWORD33_Status _CWORD33_DetachCallbackFromDispatcher (HANDLE hApp, PCSTR pServiceName, UI_32 iCmd, HANDLE hSession=NULL)

E_CWORD33_Status _CWORD33_DetachCallbacksFromDispatcherWithFd (HANDLE hApp, const int *fdArray, UI_32 uiCommandCount)

E_CWORD33_Status _CWORD33_DetachCallbackFromDispatcherWithFd (HANDLE hApp, int fd)

E_CWORD33_Status _CWORD33_DetachParentCallbacksFromDispatcher (HANDLE hApp, const PUI_32 puiCmdArray, UI_32 uiCommandCount)

E_CWORD33_Status _CWORD33_DetachServiceFromDispatcher (HANDLE hApp, PCSTR pServiceName)

E_CWORD33_Status _CWORD33_SubscribeNotificationsWithCallback (HANDLE hApp, const _CWORD33_NotificationCallbackHandler *pNtfyHandler, UI_32 uiHandlerCount)

E_CWORD33_Status _CWORD33_SubscribeNotificationWithCallback (HANDLE hApp, PCSTR pNotification, CbFuncPtr fpOnCmd)

E_CWORD33_Status _CWORD33_UnsubscribeNotificationsWithCallback (HANDLE hApp, const _CWORD33_NotificationCallbackHandler *pNtfyHandler, UI_32 uiHandlerCount)

E_CWORD33_Status _CWORD33_UnsubscribeNotificationWithCallback (HANDLE hApp, PCSTR pNotification)

PCSTR _CWORD33_GetAppName (HANDLE hApp)

E_CWORD33_Status _CWORD33_SetThreadSpecificData (HANDLE hApp, PVOID data)

PVOID _CWORD33_GetThreadSpecificData (HANDLE hApp)

HANDLE _CWORD33_OpenService (HANDLE hApp, PCSTR pServiceName)

E_CWORD33_Status _CWORD33_CloseService (HANDLE hApp, HANDLE hService)

HANDLE _CWORD33_McOpenSender (HANDLE hApp, PCSTR pName)

E_CWORD33_Status **_CWORD33_McClose** (HANDLE hService)

E_CWORD33_Status **_CWORD33_SendMsg** (HANDLE hService, UI_32 uiCmd, UI_32 uiLength, PCVOID pData)

E_CWORD33_Status _CWORD33_SendPriorityMsg (HANDLE hService, UI_32 uiCmd, UI_32 uiLength, PCVOID pData)

E_CWORD33_Status **_CWORD33_SendSelf** (HANDLE hApp, UI_32 iCmd, UI_32 length, PCVOID data)
E_CWORD33_Status _CWORD33_InvokeSync (HANDLE hService, UI_32 iCmd, UI_32 msgLenght, PCVOID msgData, UI_32 responseLength, PVOID responseData, UI_32 *receivedLength)
E_CWORD33_Status _CWORD33_SetSyncResponseData (HANDLE hApp, PVOID data, UI_32 size)
UI_32 _CWORD33_GetMsgLength (HANDLE hApp)
UI_32 _CWORD33_GetMsgProtocol (HANDLE hApp)
E_CWORD33_Status _CWORD33_GetMsgDataOfSize (HANDLE hApp, PVOID pData, UI_32 uiSize, ESMRetrieveTypes eRetrieveMethod=eSMRRelease)
E_CWORD33_Status _CWORD33_GetDataPointer (HANDLE hApp, void **datap)
PCSTR _CWORD33_GetMsgSrc (HANDLE hApp)
PCSTR _CWORD33_GetLastNotification (HANDLE hApp)
E_CWORD33_Status _CWORD33_ClearMsgData (HANDLE hApp)
E_CWORD33_Status _CWORD33_ForwardMessage (HANDLE hApp, HANDLE hChildQ, UI_32 iCmd)
E_CWORD33_Status _CWORD33_NPRegisterNotification (HANDLE hApp, PCSTR pNotification, const UI_32 max_length, const E_CWORD33_NotificationType persType)
E_CWORD33_Status _CWORD33_NPSetPersistNotfnDefaultValue (HANDLE hApp, PCSTR pNotification, PVOID data, const UI_32 uiLength)
E_CWORD33_Status _CWORD33_NPSetPersistentNotfnType (HANDLE hApp, PCSTR pNotification, E_CWORD33_PersistCategory ePersistCategory)
E_CWORD33_Status _CWORD33_NPRegisterImmediatePersistNotification (HANDLE hApp, PCSTR pNotification, const UI_32 max_length, const UI_32 delay)
E_CWORD33_Status _CWORD33_NPPersistentSync (HANDLE hApp)
E_CWORD33_Status _CWORD33_NPRegisterNotifications (HANDLE hApp, const _CWORD33_NotificationsList *pList, UI_32 uiListLength)
E_CWORD33_Status _CWORD33_NPUnRegisterNotification (HANDLE hApp, PCSTR pNotification)
E_CWORD33_Status _CWORD33_NPUnRegisterNotifications (HANDLE hApp, const _CWORD33_NotificationsList *pList, UI_32 uiListLength)
E_CWORD33_Status _CWORD33_NPPublishNotification (HANDLE hApp, PCSTR pNotification, PCVOID pData, UI_32 iLength)
E_CWORD33_Status _CWORD33_NPReadPersistedData (HANDLE hApp, PCSTR pNotification)
E_CWORD33_Status **_CWORD33_NPRegisterPersistentFile** (HANDLE hApp, PCSTR pTag, BOOL bIsUserFile=FALSE)
E_CWORD33_Status _CWORD33_NPSetFilePersistentType (HANDLE hApp, PCSTR pTag, E_CWORD33_PersistCategory ePersistCategory)
E_CWORD33_Status **_CWORD33_NPLoadPersistentFile** (HANDLE hApp, PCSTR pDstFilePath, PCSTR pTag, HANDLE hUser=NULL)
E_CWORD33_Status _CWORD33_NPReleasePersistentFile (HANDLE hApp, BOOL bIsPersist, PCSTR pTag, PCSTR pFullFilePath, HANDLE hUser=NULL)
E_CWORD33_Status _CWORD33_NPReleasePersistentFile (HANDLE hApp, E_CWORD33_ReleaseType eReleaseType, PCSTR pTag, PCSTR pFullFilePath, HANDLE hUser=NULL)
E_CWORD33_Status _CWORD33_NPRegisterPersistentFolder (HANDLE hApp, PCSTR pTag, BOOL bIsUserFolder=FALSE)
E_CWORD33_Status _CWORD33_NPSetFolderPersistentType (HANDLE hApp, PCSTR pTag, E_CWORD33_PersistCategory ePersistCategory)
E_CWORD33_Status _CWORD33_NPLoadPersistentFolder (HANDLE hApp, PCSTR pDstFolderPath, PCSTR pTag, HANDLE hUser=NULL)

E_CWORD33_Status _CWORD33_NPReleasePersistentFolder (HANDLE hApp, BOOL bIsPersist, PCSTR pTag, PCSTR pFullFolderPath, HANDLE hUser=NULL)
E_CWORD33_Status _CWORD33_NPReleasePersistentFolder (HANDLE hApp, E_CWORD33_ReleaseType e_CWORD33_ReleaseType, PCSTR pTag, PCSTR pFullFolderPath, HANDLE hUser=NULL)
E_CWORD33_Status _CWORD33_OpenSession (HANDLE hService)
E_CWORD33_Status _CWORD33_OpenSessionWithData (HANDLE hService, PVOID pData, UI_32 length)
E_CWORD33_Status _CWORD33_OpenSessionSync (HANDLE hService, OpenSessionAck *ack)
E_CWORD33_Status _CWORD33_OpenSessionWithDataSync (HANDLE hService, PVOID pData, UI_32 length, OpenSessionAck *ack)
HANDLE _CWORD33_GetOpenSessionHandle (HANDLE hApp)
HANDLE _CWORD33_GetOpenSessionSyncHandle (HANDLE hApp, OpenSessionAck *tAck)
HANDLE _CWORD33_GenerateSessionHandle (HANDLE hApp, PCSTR pServiceName)
E_CWORD33_Status _CWORD33_CloseSession (HANDLE hService, HANDLE hSession)
E_CWORD33_Status _CWORD33_CloseSessionSync (HANDLE hService, HANDLE hSession, CloseSessionAck *ack)
UI_32 _CWORD33_GenerateNewSessionId ()
UI_32 _CWORD33_GetSessionId (HANDLE hSession)
UI_32 _CWORD33_GetMsgSessionId (HANDLE hApp)
UI_32 _CWORD33_GetDeferQueueCnt (HANDLE hApp)
BOOL **_CWORD33_IsDeferQueueEmpty** (HANDLE hApp)
E_CWORD33_Status **_CWORD33_DeferMessage** (HANDLE hApp)
E_CWORD33_Status _CWORD33_ClearDeferMessages (HANDLE hApp)
E_CWORD33_Status **_CWORD33_RetrieveDeferMessage** (HANDLE hApp)
E_CWORD33_Status **_CWORD33_RegisterPersistentStorage** (HANDLE hApp, PCSTR pFullFilePath)
E_CWORD33_Status **_CWORD33_ReleaseFileToPersistentStorage** (HANDLE hApp, PCSTR pFullFilePath, BOOL persist)
E_CWORD33_Status **_CWORD33_RegisterEvent** (HANDLE hSession, UI_32 uiEventId)
E_CWORD33_Status **_CWORD33_RegisterEvents** (HANDLE hSession, PVOID puiEventsArray, UI_32 uiListSize)
E_CWORD33_Status **_CWORD33_UnRegisterEvent** (HANDLE hSession, UI_32 uiEventId)
E_CWORD33_Status **_CWORD33_UnRegisterEvents** (HANDLE hSession, PVOID puiEventsArray, UI_32 uiListSize)
E_CWORD33_Status **_CWORD33_DefineStateEvents** (HANDLE hApp, PVOID puiEvents, UI_32 uiListSize)
E_CWORD33_Status **_CWORD33_PublishEvent** (HANDLE hApp, UI_32 uiEventId, PCSTR pClientName, PCVOID pData, UI_32 iLength)
E_CWORD33_Status **_CWORD33_BroadcastEvent** (HANDLE hApp, UI_32 uiEventId, PCVOID pData, UI_32 iLength)
E_CWORD33_Status **_CWORD33_DefinePublicStateEvents** (HANDLE hApp, PUI_32 puiEvents, UI_32 uiListSize)
E_CWORD33_Status **_CWORD33_DefinePrivateStateEvents** (HANDLE hApp, PUI_32 puiEvents, UI_32 uiListSize)
E_CWORD33_Status **_CWORD33_SubscribeToSessionEventWithCallback** (HANDLE hApp, UI_32 uiEventId, CbFuncPtr fpOnCmd, HANDLE hSession)

E_CWORD33_Status **_CWORD33_SubscribeToSessionEventsWithCallbacks** (HANDLE hApp, const _CWORD33_ProtocolCallbackHandler *pEventHandler, UI_32 uiHandlerCount, HANDLE hSession)

E_CWORD33_Status **_CWORD33_UnSubscribeSessionEventWithCallback** (HANDLE hApp, UI_32 uiEventId, HANDLE hSession)

E_CWORD33_Status **_CWORD33_UnSubscribeSessionEventsWithCallbacks** (HANDLE hApp, PUI_32 pEventsArray, UI_32 uiListSize, HANDLE hSession)

E_CWORD33_Status **_CWORD33_PublishPublicEvent** (HANDLE hApp, UI_32 uiEventId, PCVOID pData, UI_32 uiLength)

E_CWORD33_Status **_CWORD33_PublishPrivateEvent** (HANDLE hApp, UI_32 uiEventId, PCVOID pData, UI_32 uiLength, HANDLE hSession)

std::map< std::string, E_CWORD33_ServiceAvailableStatus > _CWORD33_GetServiceAvailabilityTable (HANDLE hApp)

BOOL **_CWORD33_IsStateMachineApp** (HANDLE hApp)

HANDLE _CWORD33_GetXMLConfigHandle (HANDLE hApp)

UI_32 _CWORD33_GetNumberOfSession (HANDLE hApp, PCSTR strServiceName)

E_CWORD33_Status **_CWORD33_SetSessionHandle** (HANDLE hApp, PCSTR strServiceName, HANDLE hSession)

HANDLE **_CWORD33_GetSessionHandle** (HANDLE hApp, PCSTR strServiceName, UI_32 uiSessionId)

HANDLE **_CWORD33_GetCurrentSessionHandle** (HANDLE hApp)

E_CWORD33_Status **_CWORD33_RemoveSessionHandle** (HANDLE hApp, PCSTR strServiceName, UI_32 uiSessionId)

E_CWORD33_Status **_CWORD33_SetMandatoryServiceInfo** (HANDLE hApp, PCSTR pNotification, UI_32 uiEventId)

PCSTR **_CWORD33_GetSessionName** (HANDLE hSession)

BOOL **_CWORD33_IsServiceAvailable** (HANDLE hApp)

E_CWORD33_Status **_CWORD33_PublishServiceAvailability** (HANDLE hApp, BOOL bIsAvailable)

BOOL **_CWORD33_GetSelfAvailability** (HANDLE hApp)

E_CWORD33_Status **_CWORD33_RegisterServiceAvailabilityNotification** (HANDLE hApp, PCSTR pNotification)

E_CWORD33_Status **_CWORD33_UnRegisterServiceAvailabilityNotification** (HANDLE hApp)

HANDLE **_CWORD33_CreateSession** (HANDLE hApp, PCSTR pSessionName)

E_CWORD33_Status **_CWORD33_DestroySession** (HANDLE hApp, HANDLE hSession)

E_CWORD33_Status **_CWORD33_SendResponse** (HANDLE hApp, UI_32 iCmd, UI_32 length, PCVOID data)

E_CWORD33_Status _CWORD33_SendRequest (HANDLE hApp, PCSTR pServerName, UI_32 uiSessionId, UI_32 iCmd, UI_32 length, PCVOID data)

E_CWORD33_Status _CWORD33_NPSetPersonality (HANDLE hApp, PCSTR pUserName)

E_CWORD33_Status _CWORD33_NPChangePersonality (HANDLE hApp, PCSTR pUserName)

E_CWORD33_Status _CWORD33_SendStopToNSNPP (HANDLE hApp, E_CWORD33_ShutdownType eShutdownType, UI_32 uiStopMsgData=0x0)

E_CWORD33_Status **_CWORD33_NPGetReadyStatusOfNPP** (HANDLE hApp)

E_CWORD33_Status **_CWORD33_GetServiceNameOnServiceAvailabilityNotification** (HANDLE hApp, PSTR pServiceName)

HANDLE _CWORD33_GetCurrentUser (HANDLE hApp)

E_CWORD33_Status _CWORD33_SetUser (HANDLE hApp, HANDLE hUser)

E_CWORD33_Status **_CWORD33_SetAppData** (HANDLE hApp, PCSTR pKey, PVOID pData)

PVOID **_CWORD33_GetAppData** (HANDLE hApp, PCSTR pKey)
E_CWORD33_Status _CWORD33_RemoveAppData (HANDLE hApp, PCSTR pKey)
E_CWORD33_Status _CWORD33_GetSystemInfo (HANDLE hApp, PVOID pSystemInfo)
E_CWORD33_Status _CWORD33_SetDeferredSyncResponse (HANDLE hApp)
typedef std::map< std::string, S_CWORD33_SyncDataPacket * > **TSyncDataPacketList**
typedef TSyncDataPacketList::iterator **TSyncDataPacketItr**
E_CWORD33_Status _CWORD33_NPClearPersistedData (HANDLE hApp,
E_CWORD33_ClearPersistence
e_CWORD33_ClearPersistenceScope=e_CWORD33_ClearAllData)
enum **_E_CWORD33_ServiceAvailableStatus** { **e_CWORD33_ServiceAvailable** = ***,
e_CWORD33_ServiceNotAvailable }
enum _ESMRetrieveTypes { eSMRRelease = ***, eSMRRetain = *** }
enum **_E_CWORD33_Shutdowntatus** { **e_CWORD33_ShutdownOk** = ***,
e_CWORD33_ShutdownTimeout, **e_CWORD33_ShutdownFail** }
enum _E_CWORD33_SchedPolicy { **e_CWORD33_SchedPolicyInherit** = ***, **e_CWORD33_SchedPolicyTSS**, **e_CWORD33_SchedPolicyFIFO**, **e_CWORD33_SchedPolicyRR**, **e_CWORD33_SchedPolicyMAX** }*Scheduling Policy.*
typedef boost::function< E_CWORD33_Status(HANDLE) > **TCbFunction**
typedef TCbFunction **CallbackFunctionPtr**
typedef TCbFunction **CbFuncPtr**
typedef E_CWORD33_Status(* **CbArgumentParser**) (SI_32 argument, PCHAR argumentValue)
typedef enum _E_CWORD33_ServiceAvailableStatus **E_CWORD33_ServiceAvailableStatus**
typedef enum _ESMRetrieveTypes **ESMRetrieveTypes**
typedef struct __CWORD33_DefaultCallbackHandler _CWORD33_DefaultCallbackHandler
typedef struct __CWORD33_ProtocolCallbackHandler **_CWORD33_ProtocolCallbackHandler**
typedef struct __CWORD33_FdProtocolCallbackHandler **_CWORD33_FdProtocolCallbackHandler**
typedef struct __CWORD33_NotificationCallbackHandler
_CWORD33_NotificationCallbackHandler
typedef struct __CWORD33_NotificationsList **_CWORD33_NotificationsList**
typedef struct __CWORD33_ProtocolEvent **_CWORD33_ProtocolEvent**
typedef struct __CWORD33_NotificationEvent **_CWORD33_NotificationEvent**
typedef struct _ServiceAvailability **ServiceAvailability**
typedef enum _E_CWORD33_Shutdowntatus **E_CWORD33_ShutdownStatus**
typedef struct _ShutdownComplete **ShutdownComplete**
typedef struct _CustomCommandLineOptions **CustomCommandLineOptions**
typedef enum _E_CWORD33_SchedPolicy E_CWORD33_SchedPolicy
Scheduling Policy.
typedef struct __CWORD33_ChildThreadAttr _CWORD33_ChildThreadAttr
Child Thread Attribute.
#define **MAX_LEN** MAX_QUEUE_NAME_SIZE
#define **MAX_MSGQ_BUFFER** MAX_QUEUE_MSG_SIZE
#define **MAX_SESSION_ID_VAL** 0xFFFF
#define **MAX_FD_MULTIWAITING** 8
#define **_CWORD33__ANY_SOURCE** "NS_ANY_SRC"
#define **_CWORD33__ANY_COMMAND** 0xFFFFFFFF
#define **_CWORD33__NS_NPSERVICE** "NS_NPPService"
#define **_CWORD33__NS_VERSIONUTIL** "nsverutil"

#define **_CWORD33__NS_MSGPROFILERUTIL** "NSDisMsgProfile"
#define **_CWORD33__NS_REMOTECONTROLSERVICE** "NS_RemoteControlService"
HANDLE _CWORD33_CreateChildThread (HANDLE hApp, PCSTR childName, CbFuncPtr CbInitialize, CbFuncPtr CbShutdown)
HANDLE _CWORD33_CreateChildThreadWithPriority (HANDLE hApp, PCSTR childName, CbFuncPtr CbInitialize, CbFuncPtr CbShutdown, SI_32 schedPrio)
E_CWORD33_Status _CWORD33_CreateDispatcherChild (HANDLE &hChildApp, PCSTR childName, PCSTR parentName)
HANDLE _CWORD33_CreateChildThreadWithAttribute (HANDLE hApp, PCSTR childName, CbFuncPtr CbInitialize, CbFuncPtr CbShutdown, const _CWORD33_ChildThreadAttr *attr)
E_CWORD33_Status _CWORD33_CreateChildThreadAttrInit (_CWORD33_ChildThreadAttr *attr)
E_CWORD33_Status _CWORD33_CreateChildThreadAttrSetSched (_CWORD33_ChildThreadAttr *attr, E_CWORD33_SchedPolicy policy, SI_32 priority)
E_CWORD33_Status _CWORD33_StartChildThread (HANDLE hApp, HANDLE hChildQ, UI_32 length, PCVOID data)
E_CWORD33_Status _CWORD33_StopChildThread (HANDLE hApp, HANDLE hChildQ, UI_32 length, PCVOID data)
E_CWORD33_Status _CWORD33_DestroyChildThread (HANDLE hApp, HANDLE hChildQ)
E_CWORD33_Status _CWORD33_SendChild (HANDLE hApp, HANDLE hChildQ, UI_32 iCmd, UI_32 length, PCVOID data)
E_CWORD33_Status _CWORD33_SendParent (HANDLE hChildApp, UI_32 iCmd, UI_32 length, PCVOID data)
E_CWORD33_Status _CWORD33_JoinChild (HANDLE hChildApp)
E_CWORD33_Status _CWORD33_GetChildThreadPriority (HANDLE hChildApp, PSI_32 threadPrio)
#define **INHERIT_PARENT_THREAD_PRIO** -100
enum _EBaseProtocol { PROTOCOL_OPEN_SESSION_REQ = ***, PROTOCOL_OPEN_SESSION_ACK = ***, PROTOCOL_OPEN_SESSION_REQ_SYNC = ***, PROTOCOL_CLOSE_SESSION_REQ = ***, PROTOCOL_CLOSE_SESSION_ACK = ***, PROTOCOL_CLOSE_SESSION_REQ_SYNC = ***, PROTOCOL_REGISTER_EVENTS = ***, PROTOCOL_REGISTER_EVENTS_ACK = ***, PROTOCOL_UNREGISTER_EVENTS = ***, PROTOCOL_THREAD_INITIALIZATION = ***, PROTOCOL_THREAD_WAKEUP = ***, PROTOCOL_THREAD_SHUTDOWN = ***, PROTOCOL_THREAD_DESTROY = ***, PROTOCOL_RSV_0D = ***, PROTOCOL_RSV_0E = ***, PROTOCOL_DIS_MSGPROFILER = ***, PROTOCOL__CWORD33__BASE_CMD = ***, PROTOCOL__CWORD33__MAX_CMD = ***, PROTOCOL_RSV_xF1 = ***, PROTOCOL_RSV_xF2 = ***, PROTOCOL_RSV_xF3 = ***, PROTOCOL_RSV_xF4 = ***, PROTOCOL_RSV_xF5 = ***, PROTOCOL_RSV_xF6 = ***, PROTOCOL_RSV_xF7 = ***, PROTOCOL_RSV_xF8 = ***, PROTOCOL_RSV_xF9 = ***, PROTOCOL_RSV_xFA = ***, PROTOCOL_RSV_xFB = ***, PROTOCOL_RSV_xFC = ***, PROTOCOL_TIMER_DUMMY_CMD = ***, **PROTOCOL__CWORD33__NOTIFICATION** = ***, **PROTOCOL__CWORD33__ANY_COMMAND** = *** }
typedef enum _EBaseProtocol **EBaseProtocol**
typedef struct _OpenSessionAck **OpenSessionAck**
typedef struct _CloseSessionReq **CloseSessionReq**
typedef struct _CloseSessionAck **CloseSessionAck**
#define **NS_INVALID_SESSION** 0xFFFF
#define _CWORD33__PRIORITY_NOT_FOUND -1
if specified thread name not found

HANDLE _CWORD33_AttachTimerCallback (HANDLE hApp, UI_32 initMS, UI_32 repeatMS, CbFuncPtr CbFn)

E_CWORD33_Status _CWORD33_DetachTimerCallback (HANDLE hApp, HANDLE hTimer)

Detailed Description

Class Documentation

struct __CWORD33_DefaultCallbackHandler

Callbacks structure of Application Life cycle

Class Members:

CbFuncPtr	createStateMachine	Callback of created state machine.
CbFuncPtr	onBackgroundStart	Callback of Background start.
CbFuncPtr	onBackgroundStop	Callback of Background stop.
CbFuncPtr	onDebugDump	Callback of debug dump.
CbFuncPtr	onDestroy	Callback of destroy.
CbFuncPtr	onInitilization	Callback of initialization.
CbFuncPtr	onPreStart	Callback of pre start.
CbFuncPtr	onPreStop	Callback of pre stop.
CbFuncPtr	onStart	Callback of startup.
CbFuncPtr	onStop	Callback of stop.
CbFuncPtr	ssFrameworkInterface	Callback to Establish a session with the system_manager.

struct __CWORD33_ProtocolCallbackHandler

Class Members:

CbFuncPtr	callBack	
UI_32	iCmd	

struct __CWORD33_FdProtocolCallbackHandler

Class Members:

CbFuncPtr	callBack	
int	fd	

struct __CWORD33_NotificationCallbackHandler

Class Members:

CbFuncPtr	callBack	
CHAR	cNotification[MAX_STRING_SIZE_NOTIFICATION]	

struct __CWORD33_NotificationsList

Class Members:

CHAR	cNotification[MAX_STRING_SIZE_NOTIFICATION]	
E_CWORD33_NotificationType	persType	
UI_32	uiLengthData	

struct __CWORD33_ProtocolEvent

Class Members:

UI_32	iCmd	
UI_32	iEventId	

struct __CWORD33_NotificationEvent

Class Members:

CHAR	cNotification[MAX_STRING_SIZE_NOTIFICATION]	
UI_32	iEventId	

struct _ServiceAvailability

Class Members:

CHAR	cServiceName[MAX_NAME_SIZE_APP]	
E_CWORD33_ServiceAvailableStatus	eServiceAvailability	

struct _ShutdownComplete

Class Members:

CHAR	cServiceName[MAX_NAME_SIZE_APP]	
E_CWORD33_ShutdownStatus	eShutdownCompleteStatus	

struct _CustomCommandLineOptions

Class Members:

CbArgumentParser	callback	
PCHAR	cLongOptions	
PCSTR	cShortOptions	

struct __CWORD33_ChildThreadAttr

Child Thread Attribute.

Class Members:

E_CWORD33_SchedPolicy	schedPolicy	
SI_32	schedPriority	

struct _OpenSessionAck

Class Members:

CHAR	cSessionName[MAX_QUEUE_NAME_SIZE]	
E_CWORD33_Status	eStatus	
UI_32	sessionId	
UI_32	sessionType	

struct _CloseSessionReq

Class Members:

CHAR	cSessionName[MAX_QUEUE_NAME_SIZE]	
UI_32	sessionId	

struct _CloseSessionAck

Class Members:

E_CWORD33_Status	eStatus	
UI_32	sessionId	

struct _SYSTEMTIME

Class Members:

WORD	wDay	
WORD	wDayOfWeek	
WORD	wHour	
WORD	wMilliseconds	
WORD	wMinute	
WORD	wMonth	
WORD	wSecond	
WORD	wYear	

Macro Definition Documentation

#define DEFINE__CWORD33__ERROR(name)

Value:class name : public _CWORD33_::framework::error::error \
{ \
public: \
  explicit name (E_CWORD33_Status error, PCSTR str) \
    : _CWORD33_::framework::error::error (error, str) \
  {} \
}

Typedef Documentation

typedef struct __CWORD33_DefaultCallbackHandler _CWORD33_DefaultCallbackHandler

Callbacks structure of Application Life cycle

typedef UI_16 MID

Types that have been carried over from Color Radio header file WPFAPI_types.h.
CR carry over types...

Enumeration Type Documentation

enum _E_CWORD33_ClearPersistence

Enumerator

e_CWORD33_ClearAllData (currently only this enum value is supported.)
clears all the data in persistence memory for all users

e_CWORD33_ClearAllApplicationData clears all the data(files, folders) related to all application for all users

e_CWORD33_ClearAllNotificationData clears all the notification data related to all application for all users

e_CWORD33_ClearCurrentUserData clears all the data in persistence memory for current users

e_CWORD33_ClearCurrentUserApplicationData related to all application for current users
clears all the data(files, folders)

e_CWORD33_ClearCurrentUserNotificationData related to all application for current users
clears all the notification data

enum _E_CWORD33_NotificationType

Enumerator

e_CWORD33_NotificationVar Non Persistent. Not stored locally by NPS.

e_CWORD33_StateVar Non Persistent. stored locally by NPS.

e_CWORD33_PersistedStateVar Persistent. Also stored locally by NPS.

e_CWORD33_PersistedStateUserVar User specific Persistent. Also stored locally by NPS.

e_CWORD33_ImmediatePersistedStateVar Not to be used. See _CWORD33_NPRegisterImmediatePersistNotification.

e_CWORD33_Unknown defining any notification.

This is not a type of notification and should not be used while

enum _E_CWORD33_ReleaseType

defines the Release types

Enumerator

e_CWORD33_NotOnRelease 0:not on release

e_CWORD33_PersistOnShutdown 1:persist on shutdown

e_CWORD33_PersistInstantly 2:persist instantly

enum _EApplicationStates

Enumerator

eAppInital Initial: state value.

eAppIdle Idle: application shouldn't be processing anything.

eAppReady Ready: application is ready to do work.

eAppConnecting Connecting: Service is connecting.

eAppDisconnecting Disconnecting: Service is disconnecting.

enum _EBaseProtocol

Enumerator

PROTOCOL_OPEN_SESSION_REQ Open session request (client > service)

PROTOCOL_OPEN_SESSION_ACK Open session ack (service > client)

PROTOCOL_OPEN_SESSION_REQ_SYNC Open session request sync (client > service)

PROTOCOL_CLOSE_SESSION_REQ Close session request (client > service)

PROTOCOL_CLOSE_SESSION_ACK Close session ack (service > client)

PROTOCOL_CLOSE_SESSION_REQ_SYNC Close session request sync (client > service)

PROTOCOL_REGISTER_EVENTS Register for events (client > service)

PROTOCOL_REGISTER_EVENTS_ACK Register for events ack (service > client)

PROTOCOL_UNREGISTER_EVENTS Un-Register for events (client > service)

PROTOCOL_THREAD_INITIALIZATION Initialize thread after creation.

PROTOCOL_THREAD_WAKEUP Wakeup Thread after sleep.

PROTOCOL_THREAD_SHUTDOWN Stop the thread.

PROTOCOL_THREAD_DESTROY Destroy the thread.

PROTOCOL_RSV_0D Reserved.

PROTOCOL_RSV_0E Reserved.

PROTOCOL_DIS_MSGPROFILER Reserved.

PROTOCOL__CWORD33__BASE_CMD Protocols between range PROTOCOL__CWORD33__BASE_CMD to PROTOCOL__CWORD33__MAX_CMD.

PROTOCOL__CWORD33__MAX_CMD are available for *CWORD33* applications to use.

PROTOCOL_RSV_xF1 Reserved.

PROTOCOL_RSV_xF2 Reserved.

PROTOCOL_RSV_xF3 Reserved.

PROTOCOL_RSV_xF4 Reserved.

PROTOCOL_RSV_xF5 Reserved.

PROTOCOL_RSV_xF6 Reserved.

PROTOCOL_RSV_xF7 Reserved.

PROTOCOL_RSV_xF8 Reserved.

PROTOCOL_RSV_xF9 Reserved.

PROTOCOL_RSV_xFA Reserved.

PROTOCOL_RSV_xFB Reserved.

PROTOCOL_RSV_xFC Reserved.

PROTOCOL_TIMER_DUMMY_CMD NS internally use this command in timer.

enum _ESMRetrieveTypes

Enumerator

eSMRRelease data will be released on a read

eSMRRetain data will be retained in the shared memory

enum e_CWORD33_Status

Enumerator

e_CWORD33_StatusEmptyMediaList Empty media list.

e_CWORD33_StatusSessionLimitMaxedOut Maximum session limit reached.

e_CWORD33_StatusDbRecNotFound Database record not found.

e_CWORD33_StatusDbResultError Database result error.

e_CWORD33_StatusDbExecuteFail Database execute fail.

e_CWORD33_StatusSemCloseFail Semaphore close failed.

e_CWORD33_StatusSemUnlockFail Semaphore unlock failed.

e_CWORD33_StatusSemLockFail Semaphore lock failed.

e_CWORD33_StatusFail Failed.

e_CWORD33_StatusErrOther 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_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_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.

Function Documentation

E_CWORD33_Status _CWORD33_AttachCallbacksToDispatcher (HANDLE *hApp*, PCSTR *pServiceName*, const _CWORD33_ProtocolCallbackHandler * *pMsgHandler*, UI_32 *uiHandlerCount*, HANDLE *hSession* = NULL)

Brief

API to register multiple Callback-informations to the Dispatcher.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pServiceName*	PCSTR - Pointer to the service name
in	*pMsgHandler*	_CWORD33_ProtocolCallbackHandler* - Pointer to the Message Handler structure
in	*uiHandlerCount*	UI_32 - Message Handler count(14294967295)
in	*hSession*	HANDLE - Session handle (Default:NULL)

_CWORD33_ProtocolCallbackHandler Structure

```
1 typedef struct __CWORD33_ProtocolCallbackHandler
2 {
3     /* Command ID/request ID/approval ID  of protocol on the service */
4     UI_32 iCmd;
```

```
5       /* Pointer to Callback function that would be called on receiving iCmd from pServiceName. */
6       CbFuncPtr callBack;
7 }_CWORD33_ProtocolCallbackHandler;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_AttachCallbackToDispatcher, _CWORD33_DetachCallbackFromDispatcher,
_CWORD33_DetachCallbacksFromDispatcher,
_CWORD33_AttachParentCallbacksToDispatcher,
_CWORD33_DetachParentCallbacksFromDispatcher,
_CWORD33_DetachServiceFromDispatcher

E_CWORD33_Status _CWORD33_AttachCallbacksToDispatcherWithFd (HANDLE *hApp*, const _CWORD33_FdProtocolCallbackHandler * *handlers*, UI_32 *handlerCount*)

0 (fpOnCmd)NULL [e_CWORD33_StatusInvldParam]
1 (hApp)epoll [e_CWORD33_StatusFail]
2 (epoll_ctl) [e_CWORD33_StatusFail]
3 (epoll_ctl) [e_CWORD33_StatusFail]

4 **Brief**

API to register multiple Callback and file descriptor set to the Dispatcher.

5 **Parameters:**

in	*hApp*	HANDLE - HANDLE for Application
in	*handlers*	_CWORD33_FdProtocolCallbackHandler* - Pointer to the Message Handler structure to be registered
in	*handlerCount*	UI_32 - Message Handler Count(14294967295)

6

_CWORD33_FdProtocolCallbackHandler Structure

```
1 typedef struct __CWORD33_FdProtocolCallbackHandler
2 {
3       int fd;             /* File descriptor for message waiting control */
4       CbFuncPtr callBack;   /*  Pointer to Callback function             */
5 }_CWORD33_FdProtocolCallbackHandler;
```

7 **Return values:**

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred

8 **Prerequisite**

9 Generation/Initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

10 **Change of internal state**

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

E_CWORD33_Status _CWORD33_AttachCallbackToDispatcher (HANDLE *hApp*, PCSTR *pServiceName*, UI_32 *iCmd*, CbFuncPtr *fpOnCmd*, HANDLE *hSession* = NULL)

Brief

API to register a Callback-information to the Dispatcher.

Parameters:

in	*hApp*	HANDLE - Handle for Application
in	*pServiceName*	PCSTR - Pointer to the service name
in	*iCmd*	UI_32 - Command ID/request ID/approval ID of protocol on the service
in	*fpOnCmd*	CbFuncPtr - Pointer to Callback function that would be called on receiving iCmd from pServiceName.
in	*hSession*	HANDLE - Session handle (Default:NULL)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_AttachCallbacksToDispatcher, _CWORD33_DetachCallbackFromDispatcher, _CWORD33_DetachCallbacksFromDispatcher, _CWORD33_AttachParentCallbacksToDispatcher,

_CWORD33_DetachParentCallbacksFromDispatcher,
_CWORD33_DetachServiceFromDispatcher

E_CWORD33_Status _CWORD33_AttachCallbackToDispatcherWithFd (HANDLE *hApp*, int *fd*, CbFuncPtr *fpOnCmd*)

Brief

API to register a callback that link to a single file descriptor to the Dispatcher.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*fd*	int - File descriptor for message waiting control
in	*fpOnCmd*	CbFuncPtr - Pointer to Callback function

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_AttachCallbacksToDispatcherWithFd,
_CWORD33_DetachCallbackFromDispatcherWithFd,
_CWORD33_DetachCallbacksFromDispatcherWithFd

E_CWORD33_Status _CWORD33_AttachLostSessionCallbackToDispatcher (HANDLE *hApp*, CbFuncPtr *fpLostSession*)

Register abnormal session disconnect callback

Parameters:

in	*hApp*	HANDLE - Application framework handle
in	*fpLostSession*	12 Generation/Initialization of Dispatcher for the Application by _CWORD33_CreateDispatcherWithoutLoop has been done.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid Parameter

Preconditions

Generation/Initialization of Dispatcher for the Application by _CWORD33_CreateDispatcherWithoutLoop has been done.

Change of internal status

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

Classification

Public

Type

No match

See also:

_CWORD33_CreateDispatcherWithoutLoop, _CWORD33_DispatcherWithArguments, _CWORD33_SimpleDispatcher

E_CWORD33_Status _CWORD33_AttachParentCallbacksToDispatcher (HANDLE *hChildApp*, const _CWORD33_ProtocolCallbackHandler * *pMsgHandler*, UI_32 *uiHandlerCount*)

Brief

API to register multiple callback information for the parent thread to the Dispatcher.

Parameters:

in	*hChildApp*	HANDLE - HANDLE for Application
in	*pMsgHandler*	_CWORD33_ProtocolCallbackHandler* - Pointer to the Message Handler structure
in	*uiHandlerCount*	UI_32 - Message Handler count(14294967295)

_CWORD33_ProtocolCallbackHandler Structure

```
1 typedef struct __CWORD33_ProtocolCallbackHandler
2 {
3     /* Command ID/request ID/approval ID  of protocol on the service */
4     UI_32 iCmd;
5     /* Pointer to Callback function that would be called on receiving iCmd from pServiceName. */
6     CbFuncPtr callBack;
7 }_CWORD33_ProtocolCallbackHandler;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_AttachCallbackToDispatcher, _CWORD33_AttachCallbackToDispatcher, _CWORD33_DetachCallbackFromDispatcher, _CWORD33_DetachCallbacksFromDispatcher, _CWORD33_AttachParentCallbacksToDispatcher, _CWORD33_DetachParentCallbacksFromDispatcher, _CWORD33_DetachServiceFromDispatcher

HANDLE _CWORD33_AttachTimerCallback (HANDLE *hApp*, UI_32 *initMS*, UI_32 *repeatMS*, CbFuncPtr *CbFn*)

Brief

Attach a callback to hApp dispatcher that will get called initially at initNS nanoseconds, and then repeatedly at repeatNS nanoseconds

Parameters:

in	*hApp*	HANDLE - hApp parent framework HANDLE
in	*initMS*	UI_32 - milliseconds to begin triggering callback(04294967295)
in	*repeatMS*	UI_32 - milliseconds between repeated triggerings of callback(04294967295)
in	*CbFn*	CbFuncPtr - Callback to be triggered

Return values:

Handle	to the timer object or NULL on failure

Preconditons

hApp has already gotten the dispatcher(return NULL)

Change of internal status

Classification

Type

Open Close

See also:

_CWORD33_DetachTimerCallback

E_CWORD33_Status _CWORD33_ClearDeferMessages (HANDLE *hApp*)

_CWORD33_ClearDeferMessages Clear all the defer messages

Parameters:

in	*hApp*	HANDLE - Handle to the App / Thread

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_DeferMessage, _CWORD33_RetrieveDeferMessage

E_CWORD33_Status _CWORD33_ClearMsgData (HANDLE *hApp*)

Type

Sync

See also:

_CWORD33_McOpenSender, _CWORD33_McClose, _CWORD33_SendMsg, _CWORD33_GetMsgLength, _CWORD33_GetMsgSrc, _CWORD33_GetMsgDataOfSize, _CWORD33_GetDataUSID, _CWORD33_ForwardMessage, _CWORD33_GetSystemInfo

E_CWORD33_Status _CWORD33_CloseService (HANDLE *hApp*, HANDLE *hService*)

Brief

API to close service HANDLE.

Parameters:

in	*hApp*	HANDLE - Handle for Application
in	*hService*	HANDLE - Handle to the service(_CWORD33_OpenService HANDLE acquired in)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_OpenService

E_CWORD33_Status _CWORD33_CloseSession (HANDLE *hService*, HANDLE *hSession*)

Brief

API to close the session.

Parameters:

in	*hService*	HANDLE - Handle to the service(_CWORD33_OpenService HANDLE acquired in)
in	*hSession*	HANDLE - Handle to the session(_CWORD33_OpenSession HANDLE acquired in)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusInvldQName	Illegal Message Queue name
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.)

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_OpenSession, _CWORD33_OpenSessionWithData, _CWORD33_GetOpenSessionHandle

E_CWORD33_Status _CWORD33_CloseSessionSync (HANDLE *hService*, HANDLE *hSession*, CloseSessionAck * *ack*)

Brief

API to close the session (synchronous).

Parameters:

in	*hService*	HANDLE - Handle to the service(_CWORD33_OpenService HANDLE acquired in)
in	*hSession*	HANDLE - Handle to the session(_CWORD33_OpenSessionSync HANDLE acquired in)
out	*ack*	CloseSessionAck - Pointer to the response data of the session close

CloseSessionAck Structure

```
1 typedef struct _CloseSessionAck
2 {
3     UI_32 sessionId;                /* Session ID          */
4     E_CWORD33_Status eStatus;               /* Status              */
5 }CloseSessionAck;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusInvldQName	Illegal Message Queue name
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_StatusBadConnection	It can not be a socket connection
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
e_CWORD33_StatusOK	[eStatus in CloseSessionAck]Success in serivice side.
e_CWORD33_StatusFail	[eStatus in CloseSessionAck]Some sort of error occurred in service side.

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_OpenSessionSync, _CWORD33_OpenSessionWithDataSync

HANDLE _CWORD33_CreateChildThread (HANDLE *hApp*, PCSTR *childName*, CbFuncPtr *CbInitialize*, CbFuncPtr *CbShutdown*)

Brief

Create child thread and dispatcher for child thread, and initialize it

Parameters:

in	*hApp*	HANDLE - Handle for Application of parent thread
in	*childName*	PCSTR - Child thread name
in	*CbInitialize*	CbFuncPtr - Pointer to the callback function for initializing child thread
in	*CbShutdown*	CbFuncPtr - Pointer to the callback function for stopping child thread

Return values:

HANDLE	handle for communicate with child thread
NULL	Failure to get HANDLE

Prerequisite

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

Change of internal state

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

Classification

Public

Type

Open Close

See also:

_CWORD33_DestroyChildThread

E_CWORD33_Status _CWORD33_CreateChildThreadAttrInit (_CWORD33_ChildThreadAttr * *attr*)

Brief

Initialize thread attribute object.

Parameters:

out	*attr*	_CWORD33_ChildThreadAttr* - Pointer to thread attribute

_CWORD33_ChildThreadAttr Structure

```
1 typedef struct __CWORD33_ChildThreadAttr
2 {
3   E_CWORD33_SchedPolicy schedPolicy;  // Thread scheduling policy(Default is
e_CWORD33_SchedPolicyInherit)
4   SI_32 schedPriority;        // Thread priority(Default is INHERIT_PARENT_THREAD_PRIO)
5 } _CWORD33_ChildThreadAttr;
```

enum E_CWORD33_SchedPolicy Variable

```
1 typedef enum _E_CWORD33_SchedPolicy {
2   e_CWORD33_SchedPolicyInherit = ***,  // Inherit from parent thread.
3   e_CWORD33_SchedPolicyTSS,         // Time Sharing System scheduling
4   e_CWORD33_SchedPolicyFIFO,        // First in-first out scheduling
5   e_CWORD33_SchedPolicyRR,          // Round-robin scheduling
6   e_CWORD33_SchedPolicyMAX          // E_CWORD33_SchedPolicy Max(Not to be used.)
7 } E_CWORD33_SchedPolicy;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified

Prerequisite

None

Change of internal state

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

Classification

Public

Type

No match

See also:

_CWORD33_CreateChildThreadWithAttribute

E_CWORD33_Status _CWORD33_CreateChildThreadAttrSetSched
(_CWORD33_ChildThreadAttr * *attr*, E_CWORD33_SchedPolicy *policy*, SI_32 *priority*)

Brief

Initialize thread attribute object with specified value.

Parameters:

out	*attr*	_CWORD33_ChildThreadAttr* - Pointer to thread attribute
in	*policy*	E_CWORD33_SchedPolicy - Thread shcheduling policy at

		thread start
in	*priority*	SI_32 - Thread priority at thread start

_CWORD33_ChildThreadAttr Structure

```
1 typedef struct __CWORD33_ChildThreadAttr
2 {
3   E_CWORD33_SchedPolicy schedPolicy;  // Thread scheduling policy(Default is
e_CWORD33_SchedPolicyInherit)
4   SI_32 schedPriority;         // Thread priority(Default is INHERIT_PARENT_THREAD_PRIO)
5 } _CWORD33_ChildThreadAttr;
```

enum E_CWORD33_SchedPolicy Variable

```
1 typedef enum _E_CWORD33_SchedPolicy {
2   e_CWORD33_SchedPolicyInherit = ***,  // Inherit from parent thread.
3   e_CWORD33_SchedPolicyTSS,          // Time Sharing System scheduling
4   e_CWORD33_SchedPolicyFIFO,         // First in-first out scheduling
5   e_CWORD33_SchedPolicyRR,           // Round-robin scheduling
6   e_CWORD33_SchedPolicyMAX           // E_CWORD33_SchedPolicy Max(Not to be used.)
7 } E_CWORD33_SchedPolicy;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldParam	Invalid parameter

Prerequisite

None

Change of internal state

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

Classification

Public

See also:

_CWORD33_CreateChildThreadWithAttribute

HANDLE _CWORD33_CreateChildThreadWithAttribute (HANDLE *hApp*, PCSTR *childName*, CbFuncPtr *CbInitialize*, CbFuncPtr *CbShutdown*, const _CWORD33_ChildThreadAttr * *attr*)

Brief

Create child thread with specified thread attributes and dispatcher for child thread, and initialize it

Parameters:

in	*hApp*	HANDLE - Handle for Application of parent thread
in	*childName*	PCSTR - Child thread name
in	*CbInitialize*	CbFuncPtr - Pointer to the callback function for

		initializing child thread
in	*CbShutdown*	CbFuncPtr - Pointer to the callback function for stopping child thread
out	*attr*	_CWORD33_ChildThreadAttr* - Pointer to thread attributes

_CWORD33_ChildThreadAttr Structure

```
1 typedef struct __CWORD33_ChildThreadAttr
2 {
3   E_CWORD33_SchedPolicy schedPolicy;  // Thread scheduling policy(Default is
e_CWORD33_SchedPolicyInherit)
4   SI_32 schedPriority;          // Thread priority(Default is INHERIT_PARENT_THREAD_PRIO)
5 } _CWORD33_ChildThreadAttr;
```

enum E_CWORD33_SchedPolicy Variable

```
1 typedef enum _E_CWORD33_SchedPolicy {
2   e_CWORD33_SchedPolicyInherit = ***,  // Inherit from parent thread.
3   e_CWORD33_SchedPolicyTSS,          // Time Sharing System scheduling
4   e_CWORD33_SchedPolicyFIFO,         // First in-first out scheduling
5   e_CWORD33_SchedPolicyRR,           // Round-robin scheduling
6   e_CWORD33_SchedPolicyMAX           // E_CWORD33_SchedPolicy Max(Not to be used.)
7 } E_CWORD33_SchedPolicy;
```

Return values:

HANDLE	handle for communicate with child thread
NULL	Failure to get HANDLE

Prerequisite

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

Change of internal state

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

Classification

Public

Type

Open Close

See also:

_CWORD33_DestroyChildThread

HANDLE _CWORD33_CreateChildThreadWithPriority (HANDLE *hApp*, PCSTR *childName*, CbFuncPtr *CbInitialize*, CbFuncPtr *CbShutdown*, SI_32 *schedPrio*)

Brief

Create child thread with specified priority and dispatcher for child thread, and initialize it

Parameters:

in	*hApp*	HANDLE - Handle for Application of parent thread
in	*childName*	PCSTR - Child thread name
in	*CbInitialize*	CbFuncPtr - Pointer to the callback function for initializing child thread
in	*CbShutdown*	CbFuncPtr - Pointer to the callback function for stopping child thread
in	*schedPrio*	SI_32 - priority of child thread (If INHERIT_PARENT_THREAD_PRIO specified, inherit from parent thread priority)

Return values:

HANDLE	handle for communicate with child thread
NULL	Failure to get HANDLE

Prerequisite

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

Change of internal state

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

Classification

Public

Type

Open Close

See also:

_CWORD33_DestroyChildThread

E_CWORD33_Status _CWORD33_CreateDispatcherChild (HANDLE & *hChildApp*, PCSTR *childName*, PCSTR *parentName*)

Brief

Create and initialize an threads dispatcher.

Parameters:

out	*hChildApp*	HANDLE& - Reference of application handle
in	*childName*	PCSTR - create thread name
in	*parentName*	PCSTR - parent thread name

Returns:

status E_CWORD33_Status - e_CWORD33_StatusOK if Success possible errors from call CreateDispatcher

E_CWORD33_Status _CWORD33_CreateDispatcherWithoutLoop (PCSTR *cAppName*, HANDLE & *hApp*, int *argc*, char * *argv*[], const _CWORD33_DefaultCallbackHandler * *CbHandler*, BOOL *bIsConnectSM*, CustomCommandLineOptions * *cmdLineOptions* = NULL)

Brief

This API creates and initializes the dispatcher with command-line options.

Parameters:

in	*cAppName*	PCSTR - Pointer to application thread name
out	*hApp*	HANDLE - Handle for Application
in	*argc*	int - Number of command-line options
in	*argv*	char *[] - Array of pointer to command-line options
in	*CbHandler*	const _CWORD33_DefaultCallbackHandler* - Pointer to default callback functions.
in	*bIsConnectSM*	BOOL - whether or not connect with System Manager(TRUE/FALSE)
in	*cmdLineOptions*	CustomCommandLineOptions* - Parser setting of command-line options(Optional. When don't use, set NULL.)

_CWORD33_DefaultCallbackHandler Structure

```
1   typedef struct __CWORD33_DefaultCallbackHandler
2   {
3       CbFuncPtr onInitilization;          /* Function is called when a Dispatcher is created.
4                   */
5       CbFuncPtr onDestroy;                /* Function is called when the Dispatcher is
6 *     released.           */
7       CbFuncPtr onStart;                 /* Function is called when the Dispatcher is started.
8                   */
9       CbFuncPtr onStop;                  /* Function is called when the Dispatcher is stopped.
10                  */
11      CbFuncPtr onPreStart;               /* Function is called when the Dispatcher is pre started. */
12      CbFuncPtr onPreStop;                /* Function is called when the Dispatcher is pre stoped.
*/
13      CbFuncPtr onBackgroundStart;        /* Function is called when the Dispatcher is
Background started. */
14      CbFuncPtr onBackgroundStop;         /* Function is called when the Dispatcher is
Background stopped. */
15      CbFuncPtr onDebugDump;              /* Function is called when the Dispatcher detects
16 *    abnormal state. */
17      CbFuncPtr createStateMachine;       /* Set dummy function that does nothing.
18                  */
19      CbFuncPtr ssFrameworkInterface;     /* Function to connect to SystemManager
20                  */
21  } _CWORD33_DefaultCallbackHandler;
```

CustomCommandLineOptions Structure

```
1   typedef struct _CustomCommandLineOptions
2   {
3       PCSTR       cShortOptions;      /* Short options list. */
4       PCHAR       cLongOptions;       /* Reserved. Set to NULL. */
```

```
5                   CbArgumentParser callback;          /* Pointer to callback function to parse command-line
6 *                 options. */
7               } CustomCommandLineOptions;
```

About setting of default callback functions(_CWORD33_DefaultCallbackHandler)
Use _CWORD33__MAKE_DEFAULT_CALLBACK when initialize of
_CWORD33_DefaultCallbackHandler structure as argument CbHandler.
Application that run this API need to define functions below.(allow to dummy function that does nothing.)
E_CWORD33_Status _CWORD33_OnInitialization(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnPreStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnPreStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnBackgroundStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnBackgroundStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnDebugDump(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnDestroy(HANDLE hApp)
E_CWORD33_Status _CWORD33_CreateStateMachine(HANDLE hApp)
If application is resident service, link library libSS_SystemIfUnified (This library provides the function _CWORD33_SSFrameworkInterface to connect to SystemManager.).
If application is nonresident service, define function that does nothing below.
E_CWORD33_Status _CWORD33_SSFrameworkInterface(HANDLE hApp)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusDuplicate	Duplication error of entry
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
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

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

Open Close

See also:

_CWORD33_CreateDispatcherWithoutLoop, _CWORD33_DestroyDispatcherWithoutLoop, _CWORD33_GetDispatcherFD

E_CWORD33_Status _CWORD33_DestroyChildThread (HANDLE *hApp*, HANDLE *hChildQ*)

Brief

Terminate child thread.

Parameters:

in	*hApp*	HANDLE - Handle for Application of parent thread
in	*hChildQ*	HANDLE - Handle for communicate with child thread(returned by _CWORD33_CreateChildThread)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldQName	Illegal Message Queue name
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
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_StatusThreadNotExist	Thread is not exist
e_CWORD33_StatusThreadSelfJoin	Self thread specified

Prerequisite

Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.
Generation/Initialization of Dispatcher for the child thread
(_CWORD33_CreateChildThread, etc.) has been done.

Change of internal state

Change of internal state is depend on application implements.

Classification

Public

See also:

_CWORD33_CreateChildThread _CWORD33_CreateChildThreadWithAttribute
_CWORD33_CreateChildThreadWithPriority

E_CWORD33_Status _CWORD33_DestroyDispatcherWithoutLoop (HANDLE *hApp*)

Brief

This API destroy the dispatcher created by _CWORD33_CreateDispatcherWithoutLoop.

Parameters:

in	*hApp*	HANDLE - Handle for Application

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldHandle	Invalid handle

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_CreateDispatcherWithoutLoop has been done.

Change of internal state

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

Classification

Public

Type

Sync

See also:

_CWORD33_CreateDispatcherWithoutLoop

E_CWORD33_Status _CWORD33_DetachCallbackFromDispatcher (HANDLE *hApp*, PCSTR *pServiceName*, UI_32 *iCmd*, HANDLE *hSession* = NULL)

Brief

API to cancel a Callback-information from the Dispatcher.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pServiceName*	PCSTR - Pointer to the service name
in	*iCmd*	UI_32 - Command ID/request ID/approval ID of protocol on the service
in	*hSession*	HANDLE - Session handle (Default:NULL)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_AttachCallbackToDispatcher, _CWORD33_AttachCallbacksToDispatcher,
_CWORD33_DetachCallbacksFromDispatcher,
_CWORD33_AttachParentCallbacksToDispatcher,
_CWORD33_DetachParentCallbacksFromDispatcher,
_CWORD33_DetachServiceFromDispatcher

E_CWORD33_Status _CWORD33_DetachCallbackFromDispatcherWithFd (HANDLE *hApp*, int *fd*)

Brief

API to cancel a Callback and file descriptor from the Dispatcher.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*fd*	int - Cancel the target file descriptor

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_AttachCallbacksToDispatcherWithFd,
_CWORD33_AttachCallbackFromDispatcherWithFd,
_CWORD33_DetachCallbacksFromDispatcherWithFd

E_CWORD33_Status _CWORD33_DetachCallbacksFromDispatcher (HANDLE *hApp*, PCSTR *pServiceName*, const PUI_32 *puiCmdArray*, UI_32 *uiCommandCount*, HANDLE *hSession* = NULL)

Brief

API to cancel multiple Callback-informations from the Dispatcher.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pServiceName*	PCSTR - Pointer to the service name
in	*puiCmdArray*	PUI_32 - Pointer to Command ID/request ID/approval ID of protocol on the service
in	*uiCommandCount*	UI_32 - Cancel handler count
in	*hSession*	HANDLE - Session handle (Default:NULL)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_AttachCallbackToDispatcher, _CWORD33_AttachCallbacksToDispatcher, _CWORD33_DetachCallbackFromDispatcher, _CWORD33_AttachParentCallbacksToDispatcher, _CWORD33_DetachParentCallbacksFromDispatcher, _CWORD33_DetachServiceFromDispatcher

E_CWORD33_Status _CWORD33_DetachCallbacksFromDispatcherWithFd (HANDLE *hApp*, const int * *fdArray*, UI_32 *uiCommandCount*)

Brief

API to cancel multiple Callback and file descriptor set from the Dispatcher.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*fdArray*	int* - Pointer to cancel the target file descriptor array
in	*handlerCount*	UI_32 - Handler count of cancellation

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_AttachCallbacksToDispatcherWithFd, _CWORD33_AttachCallbackFromDispatcherWithFd, _CWORD33_DetachCallbackFromDispatcherWithFd

E_CWORD33_Status _CWORD33_DetachParentCallbacksFromDispatcher (HANDLE *hApp*, const PUI_32 *puiCmdArray*, UI_32 *uiCommandCount*)

Brief

API to cancel the multiple of Callback information for the parent thread from Dispatcher.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*puiCmdArray*	PUI_32 - Pointer to an array of command/request on the service protocol
in	*uiCommandCount*	UI_32 - Handler count of cancellation

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_AttachCallbackToDispatcher, _CWORD33_AttachCallbackToDispatcher, _CWORD33_DetachCallbackFromDispatcher, _CWORD33_DetachCallbacksFromDispatcher, _CWORD33_AttachParentCallbacksToDispatcher, _CWORD33_DetachParentCallbacksFromDispatcher, _CWORD33_DetachServiceFromDispatcher

E_CWORD33_Status _CWORD33_DetachServiceFromDispatcher (HANDLE *hApp*, PCSTR *pServiceName*)

Brief

Disconnect the service from Dispatcher, and API to cancel all the associated callbacks.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pServiceName*	PCSTR - Pointer to the service name

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred

Prerequisite

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

Change of internal state

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

See also:

_CWORD33_AttachCallbacksToDispatcher, _CWORD33_AttachCallbackToDispatcher, _CWORD33_DetachCallbackFromDispatcher, _CWORD33_AttachParentCallbacksToDispatcher

E_CWORD33_Status _CWORD33_DetachTimerCallback (HANDLE *hApp*, HANDLE *hTimer*)

Brief

Detach callback and deallocate a timer previous created with _CWORD33_AttachTimerCallback

Parameters:

in	*hApp*	HANDLE - hApp parent framework HANDLE
in	*hTimer*	HANDLE - timer HANDLE

Return values:

HANDLE	HANDLE used to communicate with child. NULL...failure otherwize...success

e_CWORD33_StatusOK	
e_CWORD33_StatusFail(not	happened
e_CWORD33_StatusInvldParam	

Classification

Type

Open Close

See also:

_CWORD33_AttachTimerCallback

E_CWORD33_Status _CWORD33_Dispatcher (PCSTR *cAppName*, const _CWORD33_DefaultCallbackHandler * *CbHandler*)

Brief

This API creates, initializes the dispatcher and runs mainloop for event handling.

Parameters:

in	*cAppName*	PCSTR - Pointer to application thread name
in	*CbHandler*	const _CWORD33_DefaultCallbackHandler* - Pointer to default callback functions.

_CWORD33_DefaultCallbackHandler Structure

```
typedef struct __CWORD33_DefaultCallbackHandler
{
    CbFuncPtr onInitilization;          /* Function is called when a Dispatcher is created.
            */
    CbFuncPtr onDestroy;                /* Function is called when the Dispatcher is
     released.          */
    CbFuncPtr onStart;                 /* Function is called when the Dispatcher is started.
            */
    CbFuncPtr onStop;                  /* Function is called when the Dispatcher is stopped.
             */
     CbFuncPtr onPreStart;              /* Function is called when the Dispatcher is pre started. */
     CbFuncPtr onPreStop;               /* Function is called when the Dispatcher is pre stoped.
*/
     CbFuncPtr onBackgroundStart;       /* Function is called when the Dispatcher is
Background started. */
     CbFuncPtr onBackgroundStop;        /* Function is called when the Dispatcher is
Background stopped. */
     CbFuncPtr onDebugDump;             /* Function is called when the Dispatcher detects
     abnormal state. */
     CbFuncPtr createStateMachine;      /* Set dummy function that does nothing.
             */
     CbFuncPtr ssFrameworkInterface;    /* Function to connect to SystemManager
             */
} _CWORD33_DefaultCallbackHandler;
```

About setting of default callback functions(_CWORD33_DefaultCallbackHandler)

Use _CWORD33__MAKE_DEFAULT_CALLBACK when initialize of _CWORD33_DefaultCallbackHandler structure as argument CbHandler.

Application that run this API need to define functions below.(allow to dummy function that does nothing.)

E_CWORD33_Status _CWORD33_OnInitialization(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnPreStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnPreStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnBackgroundStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnBackgroundStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnDebugDump(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnDestroy(HANDLE hApp)
E_CWORD33_Status _CWORD33_CreateStateMachine(HANDLE hApp)

If application is resident service, link library libSS_SystemIfUnified (This library provides the function _CWORD33_SSFrameworkInterface to connect to SystemManager.).

If application is nonresident service, define function that does nothing below.

E_CWORD33_Status _CWORD33_SSFrameworkInterface(HANDLE hApp)

Return values:

e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusDuplicate	Duplication error of entry
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
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

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

No match

See also:

_CWORD33_DispatcherWithArguments

E_CWORD33_Status _CWORD33_DispatcherWithArguments (PCSTR *cAppName*, int *argc*, char * *argv*[], const _CWORD33_DefaultCallbackHandler * *CbHandler*, CustomCommandLineOptions * *cmdLineOptions* = NULL)

Brief

This API creates, initializes the dispatcher and runs loop with command-line options.

Parameters:

in	*cAppName*	PCSTR - Pointer to application thread name
in	*argc*	int - Number of command-line options
in	*argv*	char *[] - Array of pointer to command-line options
in	*CbHandler*	const _CWORD33_DefaultCallbackHandler* - Pointer to default callback functions.
in	*cmdLineOptions*	CustomCommandLineOptions* - Parser setting of command-line options(Optional. When don't use, set NULL.)

_CWORD33_DefaultCallbackHandler Structure

```
1   typedef struct __CWORD33_DefaultCallbackHandler
2   {
3       CbFuncPtr onInitilization;          /* Function is called when a Dispatcher is created.
4                   */
5       CbFuncPtr onDestroy;                /* Function is called when the Dispatcher is
6 *     released.           */
7       CbFuncPtr onStart;                 /* Function is called when the Dispatcher is started.
8                   */
9       CbFuncPtr onStop;                  /* Function is called when the Dispatcher is stopped.
10                  */
11      CbFuncPtr onPreStart;               /* Function is called when the Dispatcher is pre started. */
12      CbFuncPtr onPreStop;                /* Function is called when the Dispatcher is pre stoped.
*/
13      CbFuncPtr onBackgroundStart;        /* Function is called when the Dispatcher is
Background started. */
14      CbFuncPtr onBackgroundStop;         /* Function is called when the Dispatcher is
Background stopped. */
15      CbFuncPtr onDebugDump;              /* Function is called when the Dispatcher detects
16 *    abnormal state. */
17      CbFuncPtr createStateMachine;       /* Set dummy function that does nothing.
18                  */
19      CbFuncPtr ssFrameworkInterface;     /* Function to connect to SystemManager
20                  */
21  } _CWORD33_DefaultCallbackHandler;
```

CustomCommandLineOptions Structure

```
1   typedef struct _CustomCommandLineOptions
2   {
3       PCSTR       cShortOptions;      /* Short options list. */
4       PCHAR       cLongOptions;       /* Reserved. Set to NULL. */
5       CbArgumentParser callback;      /* Pointer to callback function to parse command-line
6 *     options. */
7   } CustomCommandLineOptions;
```

About setting of default callback functions(_CWORD33_DefaultCallbackHandler)
Use _CWORD33__MAKE_DEFAULT_CALLBACK when initialize of
_CWORD33_DefaultCallbackHandler structure as argument CbHandler.
Application that run this API need to define functions below.(allow to dummy function that does nothing.)
E_CWORD33_Status _CWORD33_OnInitialization(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnPreStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnPreStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnBackgroundStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnBackgroundStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnDebugDump(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnDestroy(HANDLE hApp)
E_CWORD33_Status _CWORD33_CreateStateMachine(HANDLE hApp)
If application is resident service, link library libSS_SystemIfUnified (This library provides the function _CWORD33_SSFrameworkInterface to connect to SystemManager.).
If application is nonresident service, define function that does nothing below.
E_CWORD33_Status _CWORD33_SSFrameworkInterface(HANDLE hApp)

Return values:

e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusDuplicate	Duplication error of entry
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
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

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

No match

See also:

_CWORD33_Dispatcher, _CWORD33_CreateDispatcher, _CWORD33_DispatchBlock, _CWORD33_DispatchProcess, _CWORD33_CloseDispatcher,

E_CWORD33_Status _CWORD33_DispatchProcessWithoutLoop (HANDLE *hApp*)

Brief

Receive request or notification, and run registered callback to dispatcher.

Parameters:

in	*hApp*	HANDLE - Handle for Application

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_CreateDispatcherWithoutLoop has been done.

Change of internal state

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

Classification

Public

Type

No match

See also:

_CWORD33_CreateDispatcherWithoutLoop, _CWORD33_DestroyDispatcherWithoutLoop, _CWORD33_GetDispatcherFD

E_CWORD33_Status _CWORD33_ForwardMessage (HANDLE *hApp*, HANDLE *hChildQ*, UI_32 *iCmd*)

_CWORD33_ForwardMessage Forward a message to a service or a client.

Parameters:

in	*hApp*	HANDLE - Handle to the service
in	*hChildQ*	HANDLE - Handle to the child thread
in	*iCmd*	UI_32 - .

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_McOpenSender, _CWORD33_McClose, _CWORD33_SendMsg, _CWORD33_GetMsgLength, _CWORD33_GetMsgSrc, _CWORD33_GetMsgDataOfSize, _CWORD33_ClearMsgData, _CWORD33_GetDataUSID, _CWORD33_GetSystemInfo

UI_32 _CWORD33_GenerateNewSessionId ()

_CWORD33_GenerateNewSessionId Used on the Server side to get the next session id

Returns:

sessionID UI_32 - session ID value

See also:

_CWORD33_OpenSession, _CWORD33_GetOpenSessionHandle, _CWORD33_CloseSession, _CWORD33_GetSessionId

HANDLE _CWORD33_GenerateSessionHandle (HANDLE *hApp*, PCSTR *pServiceName*)

_CWORD33_GenerateSessionHandle API will be called by server on receiving the Open session request to Create session

Parameters:

in	*hApp*	HANDLE - Handle to the framework application
in	*pServiceName*	PCSTR - Service Name

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_OpenSession, _CWORD33_CloseSession, _CWORD33_GenerateNewSessionId _CWORD33_GetSessionId

PCSTR _CWORD33_GetAppName (HANDLE *hApp*)

Brief

Get the application thread name that is registered with the dispatcher.

Parameters:

in	*hApp*	HANDLE - Handle for Application

Return values:

PCSTR	Pointer to application thread name
NULL	NULL pointer (Failed to get application name)

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_DispatcherWithArguments, _CWORD33_Dispatcher, _CWORD33_CreateDispatcherWithoutLoop

E_CWORD33_Status _CWORD33_GetChildThreadPriority (HANDLE *hChildApp*, PSI_32 *threadPrio*)

Brief

Get the child thread priority.

Parameters:

in	*hChildApp*	HANDLE - Handle for communicate with child thread(returned by _CWORD33_CreateChildThread)
out	*threadPrio*	PSI_32 - Dispatcher file descriptor

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusThreadNotExist	Thread is not exist
e_CWORD33_StatusFault	Error occured during function

Prerequisite

Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.
Generation/Initialization of Dispatcher for the child thread
(_CWORD33_CreateChildThread, etc.) has been done.

Change of internal state

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

Classification

Public

Type

No match

See also:

_CWORD33_CreateChildThread, _CWORD33_CreateChildThreadWithPriority

HANDLE _CWORD33_GetCurrentUser (HANDLE *hApp*)

_CWORD33_GetCurrentUser Get the current user for the application

Parameters:

in	*hApp*	HANDLE - Application handle

Returns:

userhandle HANDLE - Returns current user handle

E_CWORD33_Status _CWORD33_GetDataPointer (HANDLE *hApp*, void ** *datap*)

Type

Sync

See also:

_CWORD33_McOpenSender, _CWORD33_McClose, _CWORD33_SendMsg, _CWORD33_GetMsgLength, _CWORD33_GetMsgDataOfSize, _CWORD33_ClearMsgData, _CWORD33_GetDataUSID, _CWORD33_ForwardMessage, _CWORD33_GetSystemInfo

E_CWORD33_Status _CWORD33_GetDefaultCbHandler (_CWORD33_DefaultCallbackHandler * *CbHandler*)

Brief

Get default callback handler table

Parameters:

out	*CbHandler*	_CWORD33_GetDefaultCbHandler * - Default callback handler table pointer

_CWORD33_DefaultCallbackHandler Structure

```
1   typedef struct __CWORD33_DefaultCallbackHandler
2   {
3       CbFuncPtr onInitilization;          /* Function is called when a Dispatcher is created.
4                   */
5       CbFuncPtr onDestroy;                /* Function is called when the Dispatcher is
6 *     released.           */
7       CbFuncPtr onStart;                 /* Function is called when the Dispatcher is started.
8                   */
9       CbFuncPtr onStop;                  /* Function is called when the Dispatcher is stopped.
10                  */
11      CbFuncPtr onPreStart;               /* Function is called when the Dispatcher is pre started. */
12      CbFuncPtr onPreStop;                /* Function is called when the Dispatcher is pre stoped.
*/
13      CbFuncPtr onBackgroundStart;        /* Function is called when the Dispatcher is
Background started. */
14      CbFuncPtr onBackgroundStop;         /* Function is called when the Dispatcher is
Background stopped. */
15      CbFuncPtr onDebugDump;              /* Function is called when the Dispatcher detects
16 *    abnormal state. */
17      CbFuncPtr createStateMachine;       /* Set dummy function that does nothing.
```

```
18                      */
19                  CbFuncPtr ssFrameworkInterface;         /* Function to connect to SystemManager
20                      */
21              } _CWORD33_DefaultCallbackHandler;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_CreateDispatcherWithoutLoop has been done.

Change of internal state

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

Classification

Public

Type

No match

See also:

_CWORD33_DispatchProcessWithoutLoop

UI_32 _CWORD33_GetDeferQueueCnt (HANDLE *hApp*)

_CWORD33_GetDeferQueueCnt Get the number of messages in the defer queue

Parameters:

in	*hApp*	HANDLE - Handle to the App / Thread

Returns:

count UI_32 - number of messages on Defered Queue

See also:

_CWORD33_DeferMessage, _CWORD33_RetrieveDeferMessage

E_CWORD33_Status _CWORD33_GetDispatcherFD (HANDLE *hApp*, SI_32 * *efd*)

Brief

Get the Dispatcher file descriptor corresponding to the given application handle.

Parameters:

in	*hApp*	HANDLE - Handle for Application
out	*efd*	int* - Dispatcher file descriptor

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified

e_CWORD33_StatusInvldHandle	Invalid handle

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_CreateDispatcherWithoutLoop has been done.

Change of internal state

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

Classification

Public

Type

No match

See also:

_CWORD33_DispatchProcessWithoutLoop

PCSTR _CWORD33_GetLastNotification (HANDLE *hApp*)

Type

Sync

See also:

_CWORD33_McOpenSender, _CWORD33_McClose, _CWORD33_SendMsg, _CWORD33_GetMsgLength, _CWORD33_GetMsgDataOfSize, _CWORD33_ClearMsgData, _CWORD33_GetDataUSID, _CWORD33_ForwardMessage, _CWORD33_GetSystemInfo

E_CWORD33_Status _CWORD33_GetLostSessionData (HANDLE *hApp*, PSTR *pServiceName*, PUI_32 *puiSessionId*)

Brief

Get last lost session data

Parameters:

in	*hApp*	HANDLE - Application framework handle
out	*pServiceName*	PSTR - Disconnect service name
out	*puiSessionId*	PUI_32 - Disconnect session ID

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid Parameter
e_CWORD33_StatusAccessError	Error when accessing resource

Preconditions

Generation/Initialization of Dispatcher for the Application by _CWORD33_CreateDispatcherWithoutLoop has been done.

Change of internal status

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

Classification

Public

Type

No match

See also:

_CWORD33_CreateDispatcherWithoutLoop, _CWORD33_DispatcherWithArguments, _CWORD33_SimpleDispatcher

E_CWORD33_Status _CWORD33_GetMsgDataOfSize (HANDLE *hApp*, PVOID *pData*, UI_32 *uiSize*, ESMRetrieveTypes *eRetrieveMethod* = eSMRRelease)

Type

Sync

See also:

_CWORD33_McOpenSender, _CWORD33_McClose, _CWORD33_SendMsg, _CWORD33_GetMsgLength, _CWORD33_GetMsgSrc, _CWORD33_ClearMsgData, _CWORD33_GetDataUSID, _CWORD33_ForwardMessage, _CWORD33_GetSystemInfo

UI_32 _CWORD33_GetMsgLength (HANDLE *hApp*)

Brief

API to retrieve the data length of the received message.

Parameters:

in	*hApp*	HANDLE - Handle for Application

Return values:

UI_32	The data length of the received message
0	Error or No data

Prerequisite

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

Change of internal state

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

Classification

Public

Type

Sync

See also:

None

UI_32 _CWORD33_GetMsgProtocol (HANDLE *hApp*)

Brief

API to get the Command ID/Request ID/Approval ID on the protocol.

Parameters:

in	*hApp*	HANDLE - Handle for Application

Return values:

UI_32	Command ID/Request ID/Approval ID
0	Error

Prerequisite

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

Change of internal state

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

Classification

Public

Type

Sync

See also:

None

UI_32 _CWORD33_GetMsgSessionId (HANDLE *hApp*)

Brief

Get session ID from received message.

Parameters:

in	*hApp*	HANDLE - Handle for Application

Return values:

UI_32	session ID value
MAX_SESSION_ID_VAL	Failed to get session ID value

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_GetSessionHandle

PCSTR _CWORD33_GetMsgSrc (HANDLE *hApp*)

Type

Sync

See also:

_CWORD33_McOpenSender, _CWORD33_McClose, _CWORD33_SendMsg, _CWORD33_GetMsgLength, _CWORD33_GetMsgDataOfSize, _CWORD33_ClearMsgData, _CWORD33_GetDataUSID, _CWORD33_ForwardMessage, _CWORD33_GetSystemInfo

UI_32 _CWORD33_GetNumberOfSession (HANDLE *hApp*, PCSTR *strServiceName*)

_CWORD33_GetNumberOfSession API to get the number of sessions created in the system.

Parameters:

in	*hApp*	HANDLE - Application handle
in	*strServiceName*	PCSTR - Service name.

Returns:

status E_CWORD33_Status - success or error

See also:

HANDLE _CWORD33_GetOpenSessionHandle (HANDLE *hApp*)

Brief

Asynchronous API to get the session handler.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application (_CWORD33_CreateDispatcherWithoutLoop() HANDLE acquired in)

Return values:

HANDLE	HANDLE for the session
NULL	Failure to HANDLE acquisition for the session

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_OpenSession, _CWORD33_OpenSessionWithData, _CWORD33_CloseSession, _CWORD33_GenerateNewSessionId, _CWORD33_GetSessionId

HANDLE _CWORD33_GetOpenSessionSyncHandle (HANDLE *hApp*, OpenSessionAck * *tAck*)

Brief

API to get the session ID without using the received data of ProtocolOpenSessionAck.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application (_CWORD33_CreateDispatcherWithoutLoop() HANDLE acquired in)
in	*tAck*	OpenSessionAck - Pointer to the OpenSessionAck structure

OpenSessionAck Structure

```
1 #define MAX_QUEUE_NAME_SIZE 20
2 typedef struct _OpenSessionAck
3 {
4     E_CWORD33_Status eStatus;                  /* Status             */
5     UI_32 sessionId;                    /* Session ID         */
6     CHAR  cSessionName[MAX_QUEUE_NAME_SIZE]; /* Session Name(20byte) */
7     UI_32 sessionType;                  /* Session Type       */
8 }OpenSessionAck;
```

Return values:

HANDLE	HANDLE for the session
NULL	Failure to HANDLE acquisition for the session

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_OpenSessionSync, _CWORD33_OpenSessionWithDataSync, _CWORD33_CloseSessionSync, _CWORD33_GenerateNewSessionId, _CWORD33_GetSessionId

PCSTR _CWORD33_GetServiceAvailabilityNotification (HANDLE *hApp*)

Brief

Base class of application

Parameters:

None	

Return values:

PCSTR	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

std::map<std::string, E_CWORD33_ServiceAvailableStatus>
_CWORD33_GetServiceAvailabilityTable (HANDLE *hApp*)

_CWORD33_GetServiceAvailabilityTable returns the list that stores the available services

Parameters:

in	*hApp*	HANDLE - Handle to the application session

Returns:

ServiceAvailabilityList ServiceAvailabilityTable - ServiceAvailabilityList

See also:

_CWORD33_RegisterEvent

UI_32 _CWORD33_GetSessionId (HANDLE *hSession*)

Brief

Returns the session id corresponding to the given session handle.

Parameters:

in	*hSession*	HANDLE - Handle to the session(_CWORD33_McOpenSender HANDLE acquired in)

Return values:

UI_32	session ID value
MAX_SESSION_ID_VAL	Failed to get session ID value

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_OpenSession, _CWORD33_OpenSessionSync, _CWORD33_OpenSessionWithData, _CWORD33_OpenSessionWithDataSync, _CWORD33_GetOpenSessionHandle, _CWORD33_GetOpenSessionSyncHandle, _CWORD33_GetCurrentSessionHandle, _CWORD33_McOpenSender, _CWORD33_CreateSession

E_CWORD33_Status _CWORD33_GetSystemInfo (HANDLE *hApp*, PVOID *pSystemInfo*)

_CWORD33_GetSystemInfo Gets some internal notification information

Parameters:

in	*hApp*	HANDLE - Handle to a Application Framework
in,out	*pSystemInfo*	PVOID - Buffer pointer to which system info is copied.

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_McOpenSender, _CWORD33_SendMsg, _CWORD33_GetMsgLength, _CWORD33_GetMsgSrc, _CWORD33_GetMsgDataOfSize, _CWORD33_ClearMsgData, _CWORD33_GetDataUSID, _CWORD33_ForwardMessage, _CWORD33_GetSystemInfo

PVOID _CWORD33_GetThreadSpecificData (HANDLE *hApp*)

Brief

Get pointer to the application-specific data from the application HANDLE.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application

Return values:

PVOID	pointer to application-specific data.
NULL	NULL pointer(failed to get data)

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_SetThreadSpecificData

HANDLE _CWORD33_GetXMLConfigHandle (HANDLE *hApp*)

_CWORD33_GetXMLConfigHandle Returns the handle to config file handle

Parameters:

in	*hApp*	HANDLE - Handle to the application session

Returns:

handle HANDLE - Config file handle

See also:

E_CWORD33_Status _CWORD33_InvokeSync (HANDLE *hService*, UI_32 *iCmd*, UI_32 *msgLenght*, PCVOID *msgData*, UI_32 *responseLength*, PVOID *responseData*, UI_32 * *receivedLength*)

Brief

API for synchronous communication of the message.

Parameters:

in	*hService*	HANDLE - Handle for the message queue
in	*iCmd*	UI_32 - Command ID
in	*msgLenght*	UI_32 - Data length of the send message

in	*msgData*	PCVOID - Pointer to send the message
in	*responseLength*	UI_32 - Size of the received message storage buffer
out	*responseData*	PVOID - Pointer to the received message storage buffer
out	*receivedLength*	UI_32 * - Data length of the received message

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusInvldQName	Illegal Message Queue name
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_StatusInvldBuf	Invalid buffer
e_CWORD33_StatusBadConnection	It can not be a socket connection
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_SendMsg

E_CWORD33_Status _CWORD33_JoinChild (HANDLE *hChildApp*)

Brief

Wait for child thread terminates.

Parameters:

in	*hChildApp*	HANDLE - Handle for communicate with child thread(returned by _CWORD33_CreateChildThread)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusThreadNotExist	Thread is not exist
e_CWORD33_StatusInvldParam	Invalid parameter

e_CWORD33_StatusThreadSelfJoin	Self thread specified
e_CWORD33_StatusFail	thread exit or invalid

Prerequisite

Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.
Generation/Initialization of Dispatcher for the child thread
(_CWORD33_CreateChildThread, etc.) has been done.

Change of internal state

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

Classification

Public

See also:

_CWORD33_CreateChildThread, _CWORD33_CreateChildThreadWithPriority, _CWORD33_DestroyChildThread

HANDLE _CWORD33_McOpenSender (HANDLE *hApp*, PCSTR *pName*)

Brief

Generate a handle to the message queue for sending.

Parameters:

in	*hApp*	HANDLE - Handle for Application
in	*pName*	PCSTR - Message queue name of destination service

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_McClose

E_CWORD33_Status _CWORD33_NPChangePersonality (HANDLE *hApp*, PCSTR *pUserName*)

_CWORD33_NPChangePersonality API to send message to Notification Service to set new Personality

Parameters:

in	*hApp*	HANDLE - Application Handle
in	*pUserName*	PCSTR - Name of the new Personality

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_NPRegisterPersistentFile, _CWORD33_NPLoadPersistentFile

E_CWORD33_Status _CWORD33_NPClearPersistedData (HANDLE *hApp*, E_CWORD33_ClearPersistence *e_CWORD33_ClearPersistenceScope* = e_CWORD33_ClearAllData)

Summary

API to delete all the persistent files and notification data from persistent memory related to NS_NPS. Note: This API is only to be used HMI service. This API should not be used by the other application.

Parameters:

in	*hApp*	hApp - Handle to the Framework

hApp HANDLE*

Parameters:

in	*e_CWORD33_ClearPersistenceScope*	e_CWORD33_ClearPersistenceScope - specifies what data to delete from persistent memory

e_CWORD33_ClearPersistenceScope E_CWORD33_ClearPersistence enum

```
1 typedef enum _E_CWORD33_ClearPersistence {
2   e_CWORD33_ClearAllData = ***,        ///< clears all the data in persistence memory for all users
3   ///< (currently only this enum value is supported.)
4   e_CWORD33_ClearAllApplicationData,      ///< clears all the data(files, folders) related to all application for
all
5   users
6   e_CWORD33_ClearAllNotificationData,     ///< clears all the notification data related to all application for
all users
7   e_CWORD33_ClearCurrentUserData,       ///< clears all the data in persistence memory for current users
8   e_CWORD33_ClearCurrentUserApplicationData,  ///< clears all the data(files, folders) related to all
application for
9   current users
10   e_CWORD33_ClearCurrentUserNotificationData  ///< clears all the notification data related to all
application for
11   current users
12 } E_CWORD33_ClearPersistence;
```

Return values:

E_CWORD33_Status	

e_CWORD33_StatusOK e_CWORD33_StatusNullPointer e_CWORD33_StatusInvldParam

Preconditions

Change of the internal state

The internal state is not changed.

Classification

public

Type

sync only

See also:

None

E_CWORD33_Status _CWORD33_NPLoadPersistentFolder (HANDLE *hApp*, PCSTR *pDstFolderPath*, PCSTR *pTag*, HANDLE *hUser* = NULL)

API to send message to Notification Service to copy folder from persistent memory to specified path. The caller recieves an acknowledgement once NPS completes folder copy

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pDstFolderPath*	PCSTR - Complete folder path to which the folder should be copied from persistent area
in	*pTag*	PCSTR - Tag associated with the folder that needs to be copied to pDstFolderPath
in	*hUser*	HANDLE - Handle for the user

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_NPRegisterPersistentFolder, _CWORD33_NPReleasePersistentFolder

E_CWORD33_Status _CWORD33_NPPersistentSync (HANDLE *hApp*)

_CWORD33_NPPersistentSync Processing which synchronizes by NPPService (syncfs)

Parameters:

in	*hApp*	HANDLE - Handle to the Framework

Returns:

status E_CWORD33_Status - success or error

E_CWORD33_Status _CWORD33_NPPublishNotification (HANDLE *hApp*, PCSTR *pNotification*, PCVOID *pData*, UI_32 *iLength*)

Brief

API to send message to Notification Service to notify subscribers

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pNotification*	PCSTR - A pointer to the Notification name
in	*pData*	VOID * - Data buffer pointer
in	*iLength*	UI_32 - Size of data buffer

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 kind of error has occurred
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Prerequisite

NS_NPPService of the process has been started.
Generation/Initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of internal state

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

Classification

Public

See also:

_CWORD33_NPRegisterNotification, _CWORD33_NPUnRegisterNotification
_CWORD33_NPRegisterNotifications, _CWORD33_NPUnRegisterNotifications

E_CWORD33_Status _CWORD33_NPReadPersistedData (HANDLE *hApp*, PCSTR *pNotification*)

Brief

API to requested the persistent data corresponding to the notification if available

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pNotification*	PCSTR - Name of Notification

Returns:

status E_CWORD33_Status - success or error

Classification

Public

Type

Pub-Sub

See also:

none

E_CWORD33_Status _CWORD33_NPRegisterImmediatePersistNotification (HANDLE *hApp*, PCSTR *pNotification*, const UI_32 *max_length*, const UI_32 *delay*)

Brief

API to send message to Notification Service to register a immediate notification

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pNotification*	PCSTR - Name of Notification
in	*max_length*	const UI_32 - Max size of the notification message (should be <= ***Kb)
in	*delay*	const UI_32 - Min time interval between every persist (in sec)

Returns:

status E_CWORD33_Status - success or error

Classification

Public

Type

Pub-Sub

See also:

_CWORD33_NPRegisterNotifications, _CWORD33_NPUnRegisterNotification, _CWORD33_NPUnRegisterNotifications, _CWORD33_NPPublishNotification, _CWORD33_SubscribeNotificationWithCallback, _CWORD33_UnsubscribeNotificationWithCallback, _CWORD33_SubscribeNotificationsWithCallback, _CWORD33_UnsubscribeNotificationsWithCallback

E_CWORD33_Status _CWORD33_NPRegisterNotification (HANDLE *hApp*, PCSTR *pNotification*, const UI_32 *max_length*, const E_CWORD33_NotificationType *persType*)

Brief

To the notification service to register the Notification, API to send a message.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pNotification*	PCSTR - A pointer to the Notification name
in	*max_length*	UI_32 - The maximum length of the Notification message (64byte below)
in	*persType*	

E_CWORD33_NotificationType - Type of Notification enum E_CWORD33_NotificationType Variable
e_CWORD33_NotificationVar : Non Persistent. Not stored locally by NPS.
e_CWORD33_StateVar : Non Persistent. stored locally by NPS.
e_CWORD33_PersistedStateVar : Not to be used. Persistent. Also stored locally by NPS.
e_CWORD33_PersistedStateUserVar : Not to be used. User specific Persistent. Also stored locally by NPS.
e_CWORD33_ImmediatePersistedStateVar : Not to be used.
e_CWORD33_Unknown : Not to be used. This is not a type of 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.)

Prerequisite

NS_NPPService of the process has been started.
Generation/Initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of internal state

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

Classification

Public

Type

Pub-Sub

See also:

_CWORD33_NPRegisterNotifications, _CWORD33_NPUnRegisterNotification, _CWORD33_NPUnRegisterNotifications, _CWORD33_NPPublishNotification, _CWORD33_SubscribeNotificationWithCallback, _CWORD33_UnsubscribeNotificationWithCallback, _CWORD33_SubscribeNotificationsWithCallback, _CWORD33_UnsubscribeNotificationsWithCallback

E_CWORD33_Status _CWORD33_NPRegisterNotifications (HANDLE *hApp*, const _CWORD33_NotificationsList * *pList*, UI_32 *uiListLength*)

Brief

API to send a message to the notification service in order to register multiple Notification of list format.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pList*	_CWORD33_NotificationsList - Pointer to the Notification list to be registered
in	*uiListLength*	UI_32 - List length

_CWORD33_NotificationsList Structure

```
1 #define MAX_STRING_SIZE_NOTIFICATION  64
2 typedef struct __CWORD33_NotificationsList
3 {
4     CHAR cNotification[MAX_STRING_SIZE_NOTIFICATION]; /* Notification Name                */
5     UI_32 uiLengthData;                    /* Notification Name length(64byte below) */
6     E_CWORD33_NotificationType  persType;          /* Notification Type                */
7 }
```

enum E_CWORD33_NotificationType Variable

e_CWORD33_NotificationVar : Non Persistent. Not stored locally by NPS.

e_CWORD33_StateVar : Non Persistent. stored locally by NPS.

e_CWORD33_PersistedStateVar : Not to be used. Persistent. Also stored locally by NPS.

e_CWORD33_PersistedStateUserVar : Not to be used. User specific Persistent. Also stored locally by NPS.

e_CWORD33_ImmediatePersistedStateVar : Not to be used.

e_CWORD33_Unknown : Not to be used. This is not a type of 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.)

Prerequisite

NS_NPPService of the process has been started.
Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of internal state

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

Classification

Public

See also:

_CWORD33_NPRegisterNotification, _CWORD33_NPUnRegisterNotification, _CWORD33_NPUnRegisterNotifications, _CWORD33_NPPublishNotification, _CWORD33_SubscribeNotificationWithCallback, _CWORD33_UnsubscribeNotificationWithCallback, _CWORD33_SubscribeNotificationsWithCallback, _CWORD33_UnsubscribeNotificationsWithCallback

E_CWORD33_Status _CWORD33_NPRegisterPersistentFolder (HANDLE *hApp*, PCSTR *pTag*, BOOL *bIsUserFolder* = FALSE)

API to send message to Notification Service to add a folder path to be persisted

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pTag*	PCSTR - A unique identifier for the folder path specified, this is used in retrieve/Load the folder later
in	*bIsUserFolder*	BOOL - set TRUE if want to register a folder for user.

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_NPLoadPersistentFolder, _CWORD33_NPReleasePersistentFolder

E_CWORD33_Status _CWORD33_NPReleasePersistentFile (HANDLE *hApp*, BOOL *bIsPersist*, PCSTR *pTag*, PCSTR *pFullFilePath*, HANDLE *hUser* = NULL)

Brief

API to send message to Notification Service to notify that the file can be persisted.

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*bIsPersist*	BOOL - If the specified file needs to be persisted or not on release
in	*pTag*	PCSTR - Tag associated with the file released
in	*pFullFilePath*	PCSTR - Full path name of the file to be persisted
in	*hUser*	HANDLE - Handle for the user

Returns:

status E_CWORD33_Status - success or error

Classification

Public

Type

Pub-Sub

See also:

_CWORD33_NPRegisterPersistentFile, _CWORD33_NPLoadPersistentFile

E_CWORD33_Status _CWORD33_NPReleasePersistentFile (HANDLE *hApp*, E_CWORD33_ReleaseType *eReleaseType*, PCSTR *pTag*, PCSTR *pFullFilePath*, HANDLE *hUser* = NULL)

Brief

API to send message to Notification Service to notify that the file can be persisted.

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*eReleaseType*	E_CWORD33_ReleaseType - e_CWORD33_NotOnRelease :not on release e_CWORD33_PersistOnShutdown:persist on shutdown e_CWORD33_PersistInstantly :persist instantly
in	*pTag*	PCSTR - Tag associated with the file released
in	*pFullFilePath*	PCSTR - Full path name of the file to be persisted
in	*hUser*	HANDLE - Handle for the user

Returns:

status E_CWORD33_Status - success or error

Classification

Public

Type

Pub-Sub

See also:

_CWORD33_NPRegisterPersistentFile, _CWORD33_NPLoadPersistentFile

E_CWORD33_Status _CWORD33_NPReleasePersistentFolder (HANDLE *hApp*, BOOL *bIsPersist*, PCSTR *pTag*, PCSTR *pFullFolderPath*, HANDLE *hUser* = NULL)

_CWORD33_NPReleasePersistentFolder API to send message to Notification Service to notify that the folder can be persisted.

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*bIsPersist*	BOOL - If the specified folder needs to be persisted or not on release
in	*pTag*	PCSTR - Tag associated with the folder released
in	*pFullFolderPath*	PCSTR - Full path name of the folder to be persisted
in	*hUser*	HANDLE - Handle for the user

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_NPRegisterPersistentFolder, _CWORD33_NPLoadPersistentFolder

E_CWORD33_Status _CWORD33_NPReleasePersistentFolder (HANDLE *hApp*, E_CWORD33_ReleaseType *e_CWORD33_ReleaseType*, PCSTR *pTag*, PCSTR *pFullFolderPath*, HANDLE *hUser* = NULL)

_CWORD33_NPReleasePersistentFolder API to send message to Notification Service to notify that the folder can be persisted.

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*e_CWORD33_ReleaseType*	E_CWORD33_ReleaseType - e_CWORD33_NotOnRelease :not on release e_CWORD33_PersistOnShutdown:persist on shutdown e_CWORD33_PersistInstantly :persist instantly
in	*pTag*	PCSTR - Tag associated with the folder released
in	*pFullFolderPath*	PCSTR - Full path name of the folder to be persisted
in	*hUser*	HANDLE - Handle for the user

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_NPRegisterPersistentFolder, _CWORD33_NPLoadPersistentFolder

E_CWORD33_Status _CWORD33_NPSetFilePersistentType (HANDLE *hApp*, PCSTR *pTag*, E_CWORD33_PersistCategory *ePersistCategory*)

_CWORD33_NPSetFilePersistentType API to send message to Notification Service to set persistent file category.

Parameters:

in	*hApp*	HANDLE - Handle to the Framework

in	*pTag*	PCSTR - File tag
in	*ePersistCategory*	E_CWORD33_PersistCategory - Persistent category

Returns:

status E_CWORD33_Status - success or error

E_CWORD33_Status _CWORD33_NPSetFolderPersistentType (HANDLE *hApp*, PCSTR *pTag*, E_CWORD33_PersistCategory *ePersistCategory*)

_CWORD33_NPSetFolderPersistentType API to send message to Notification Service to set persistent folder category.

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pTag*	PCSTR - A unique identifier for the folder path specified, this is used in retrieve/Load the folder later
in	*ePersistCategory*	E_CWORD33_PersistCategory - Persistent category

Returns:

status E_CWORD33_Status - success or error

E_CWORD33_Status _CWORD33_NPSetPersistentNotfnType (HANDLE *hApp*, PCSTR *pNotification*, E_CWORD33_PersistCategory *ePersistCategory*)

Brief

API to send message to Notification Service to set persistent notification category.

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pNotification*	PCSTR - Name of Notification
in	*ePersistCategory*	E_CWORD33_PersistCategory - Persistent Category

Returns:

status E_CWORD33_Status - success or error

Classification

Public

Type

Pub-Sub

See also:

none

E_CWORD33_Status _CWORD33_NPSetPersistNotfnDefaultValue (HANDLE *hApp*, PCSTR *pNotification*, PVOID *data*, const UI_32 *uiLength*)

_CWORD33_NPSetPersistNotfnDefaultValue API to send message to Notification Service to set the default value for the notification.

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pNotification*	PCSTR - Name of Notification
in	*data*	PVOID - Set the default value for the notification.
in	*uiLength*	const UI_32 - length of the data

Returns:

status E_CWORD33_Status - success or error

E_CWORD33_Status _CWORD33_NPSetPersonality (HANDLE *hApp*, PCSTR *pUserName*)

_CWORD33_NPSetPersonality API to send message to Notification Service to set Personality

Parameters:

in	*hApp*	HANDLE - Application Handle
in	*pUserName*	PCSTR - Name of the new Personality

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_NPRegisterPersistentFile, _CWORD33_NPLoadPersistentFile

E_CWORD33_Status _CWORD33_NPUnRegisterNotification (HANDLE *hApp*, PCSTR *pNotification*)

Brief

API to send message to Notification Service to remove a notification.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pNotification*	PCSTR - A pointer to the Notification name

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.)

Prerequisite

NS_NPPService of the process has been started.
Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of internal state

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

Classification

Public

See also:

_CWORD33_NPRegisterNotification, _CWORD33_NPRegisterNotifications,
_CWORD33_NPUnRegisterNotifications, _CWORD33_NPPublishNotification,
_CWORD33_SubscribeNotificationWithCallback,
_CWORD33_UnsubscribeNotificationWithCallback,
_CWORD33_SubscribeNotificationsWithCallback,
_CWORD33_UnsubscribeNotificationsWithCallback

E_CWORD33_Status _CWORD33_NPUnRegisterNotifications (HANDLE *hApp*, const _CWORD33_NotificationsList * *pList*, UI_32 *uiListLength*)

Brief

API to send message to Notification Service to delete a notification.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pList*	_CWORD33_NotificationsList - Pointer to the Notification list that you want to delete
in	*uiListLength*	UI_32 - List length

_CWORD33_NotificationsList Structure

```
1 #define MAX_STRING_SIZE_NOTIFICATION  64
2 typedef struct __CWORD33_NotificationsList
3 {
4     CHAR cNotification[MAX_STRING_SIZE_NOTIFICATION]; /* Notification Name                */
5     UI_32 uiLengthData;                       /* Notification Name length(64byte below) */
6     E_CWORD33_NotificationType  persType;             /* Notification Type                */
7 }
```

enum E_CWORD33_NotificationType Variable
e_CWORD33_NotificationVar : Non Persistent. Not stored locally by NPS.
e_CWORD33_StateVar : Non Persistent. stored locally by NPS.
e_CWORD33_PersistedStateVar : Not to be used. Persistent. Also stored locally by NPS.

e_CWORD33_PersistedStateUserVar : Not to be used. User specific Persistent. Also stored locally by NPS.
e_CWORD33_ImmediatePersistedStateVar : Not to be used.
e_CWORD33_Unknown : Not to be used. This is not a type of notification.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusNullPointer	NULL pointer specified
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.)

Prerequisite

NS_NPPService of the process has been started.
Generation/Initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of internal state

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

Classification

Public

See also:

_CWORD33_NPRegisterNotification, _CWORD33_NPRegisterNotifications, _CWORD33_NPUnRegisterNotification, _CWORD33_NPPublishNotification, _CWORD33_SubscribeNotificationWithCallback, _CWORD33_UnsubscribeNotificationWithCallback, _CWORD33_SubscribeNotificationsWithCallback, _CWORD33_UnsubscribeNotificationsWithCallback

E_CWORD33_Status _CWORD33_OnBackgroundStart (HANDLE *hApp*)

Brief

callback of Application Life cycle event

Parameters:

in	*hApp*	HANDLE - Application Handle

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

Depend on the implementation in the inheriting class

Change of internal state

Depend on the implementation in the inheriting class

Classification

Public

Type

Sync Only

include _CWORD33__application.h
Library libNS_FrameworkUnified.so
Set compile option for "-DIMPL__CWORD33__APPLICATION_CALLBACKS_PRE_BACKGROUND" when this API in the each application implements.

See also:

E_CWORD33_Status _CWORD33_OnBackgroundStop (HANDLE *hApp*)

Brief

callback of Application Life cycle event

Parameters:

in	*hApp*	HANDLE - Application Handle

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

Depend on the implementation in the inheriting class

Change of internal state

Depend on the implementation in the inheriting class

Classification

Public

Type

Sync Only

include _CWORD33__application.h
Library libNS_FrameworkUnified.so
Set compile option for "-DIMPL__CWORD33__APPLICATION_CALLBACKS_PRE_BACKGROUND" when this API in the each application implements.

See also:

E_CWORD33_Status _CWORD33_OnDebugDump (HANDLE *hApp*)

Brief

Base class of application

Parameters:

None	

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

E_CWORD33_Status _CWORD33_OnDestroy (HANDLE *hApp*)

Brief

Base class of application

Parameters:

None	

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

E_CWORD33_Status _CWORD33_OnDummy (HANDLE *hApp*)

Brief

dummy callback of Application Life cycle event

Parameters:

in	*hApp*	HANDLE - Application Handle

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

Sync Only

include _CWORD33__application.h
Library libNS_FrameworkUnified.so

See also:

E_CWORD33_Status _CWORD33_OnEShutdown (HANDLE *hApp*)

<< deprecated

Brief

Base class of application

Parameters:

None	

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

E_CWORD33_Status _CWORD33_OnInitialization (HANDLE *hApp*)

Brief

Base class of application

Parameters:

None	

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

E_CWORD33_Status _CWORD33_OnPreStart (HANDLE *hApp*)

Brief

callback of Application Life cycle event

Parameters:

in	*hApp*	HANDLE - Application Handle

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

Depend on the implementation in the inheriting class

Change of internal state

Depend on the implementation in the inheriting class

Classification

Public

Type

Sync Only

include _CWORD33__application.h
Library libNS_FrameworkUnified.so
Set compile option for "-DIMPL__CWORD33__APPLICATION_CALLBACKS_PRE_BACKGROUND" when this API in the each application implements.

See also:

E_CWORD33_Status _CWORD33_OnPreStop (HANDLE *hApp*)

Brief

callback of Application Life cycle event

Parameters:

in	*hApp*	HANDLE - Application Handle

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

Depend on the implementation in the inheriting class

Change of internal state

Depend on the implementation in the inheriting class

Classification

Public

Type

Sync Only

include _CWORD33__application.h
Library libNS_FrameworkUnified.so

Set compile option for "-DIMPL__CWORD33__APPLICATION_CALLBACKS_PRE_BACKGROUND" when this API in the each application implements.

See also:

E_CWORD33_Status _CWORD33_OnReinit (HANDLE *hApp*)

Brief

Base class of application

Parameters:

None	

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

E_CWORD33_Status _CWORD33_OnShutdown (HANDLE *hApp*)

<< deprecated

Brief

Base class of application

Parameters:

None	

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

E_CWORD33_Status _CWORD33_OnStart (HANDLE *hApp*)

<< deprecated

Brief

Base class of application

Parameters:

None	

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

E_CWORD33_Status _CWORD33_OnStop (HANDLE *hApp*)

Brief

Base class of application

Parameters:

None	

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

E_CWORD33_Status _CWORD33_OnWakeup (HANDLE *hApp*)

Brief

Base class of application

Parameters:

None	

Return values:

E_CWORD33_Status	depends on the implement of derived class

Prerequisite

depends on the implement of derived class

Inside state change

depends on the implement of derived class

None

Classification

Public

Type

Sync Only

HANDLE _CWORD33_OpenService (HANDLE *hApp*, PCSTR *pServiceName*)

Brief

API to acquire a service HANDLE.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pServiceName*	PCSTR - Pointer to service name

Return values:

HANDLE	Handle to the service
NULL	NULL pointer (Failed to get service handle)

Prerequisite

Generation/Initialization of Dispatcher for the Application (_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.
Only when will open the session by _CWORD33_OpenSession, you can use the handle obtained by this API.

Change of internal state

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

Classification

Public

See also:

_CWORD33_CloseService, _CWORD33_OpenSession

E_CWORD33_Status _CWORD33_OpenSession (HANDLE *hService*)

Brief

API sends a message to the service requesting a session.

Parameters:

in	*hService*	HANDLE - Handle to the service(_CWORD33_OpenService HANDLE acquired in)

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusInvldQName	Illegal Message Queue name
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.)

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_GetOpenSessionHandle, _CWORD33_OpenSessionWithData, _CWORD33_CloseSession, _CWORD33_GenerateNewSessionId, _CWORD33_GetSessionId

E_CWORD33_Status _CWORD33_OpenSessionSync (HANDLE *hService*, OpenSessionAck * *ack*)

Brief

Synchronous API of sending a message of the session request to the service.

Parameters:

in	*hService*	HANDLE - Handle to the service(_CWORD33_OpenService/_CWORD33_GetOpenSessionHandle/ _CWORD33_GetOpenSessionSyncHandle HANDLE acquired in)
out	*ack*	OpenSessionAck - Pointer to the response data of the session initiation

OpenSessionAck Structure

```
1 #define MAX_QUEUE_NAME_SIZE 20
2 typedef struct _OpenSessionAck
3 {
4     E_CWORD33_Status eStatus;                  /* Status              */
5     UI_32 sessionId;                  /* Session ID          */
6     CHAR  cSessionName[MAX_QUEUE_NAME_SIZE]; /* Session Name(20byte)  */
7     UI_32 sessionType;                 /* Session Type        */
8 }OpenSessionAck;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusInvldQName	Illegal Message Queue name
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_StatusBadConnection	It can not be a socket connection
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared

	memory, etc.)

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_OpenSessionWithDataSync, _CWORD33_GetOpenSessionSyncHandle, _CWORD33_CloseSessionSync, _CWORD33_GenerateNewSessionId, _CWORD33_GetSessionId

E_CWORD33_Status _CWORD33_OpenSessionWithData (HANDLE *hService*, PVOID *pData*, UI_32 *length*)

Brief

API to send to service the message of the session request together with the data.

Parameters:

in	*hService*	HANDLE - Handle to the service(_CWORD33_OpenService HANDLE acquired in)
in	*pData*	PVOID - Pointer to the session data for transmission
in	*length*	UI_32 - Size of the session data for transmission

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldBuf	Invalid buffer
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusInvldQName	Illegal Message Queue name
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.)

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_OpenSession, _CWORD33_GetOpenSessionHandle, _CWORD33_CloseSession, _CWORD33_GenerateNewSessionId, _CWORD33_GetSessionId

E_CWORD33_Status _CWORD33_OpenSessionWithDataSync (HANDLE *hService*, PVOID *pData*, UI_32 *length*, OpenSessionAck * *ack*)

Brief

Synchronous API that transmits to the service a message session request with the data.

Parameters:

in	*hService*	HANDLE - Handle to the service(_CWORD33_OpenService/_CWORD33_GetOpenSessionHandle/ _CWORD33_GetOpenSessionSyncHandle HANDLE acquired in)
in	*pData*	PVOID -
in	*length*	UI_32 -
out	*ack*	OpenSessionAck - Pointer to the response data of the session initiation

OpenSessionAck Structure

```
1 #define MAX_QUEUE_NAME_SIZE 20
2 typedef struct _OpenSessionAck
3 {
4     E_CWORD33_Status eStatus;                  /* Status              */
5     UI_32 sessionId;                  /* Session ID          */
6     CHAR  cSessionName[MAX_QUEUE_NAME_SIZE]; /* Session Name(20byte)  */
7     UI_32 sessionType;                 /* Session Type        */
8 }OpenSessionAck;
```

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldBuf	Invalid Buffer
e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldHndlType	Invalid type of handle
e_CWORD33_StatusInvldQName	Illegal Message Queue name
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_StatusBadConnection	It can not be a socket connection
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_OpenSessionSync, _CWORD33_GetOpenSessionSyncHandle, _CWORD33_CloseSessionSync, _CWORD33_GenerateNewSessionId, _CWORD33_GetSessionId

E_CWORD33_Status _CWORD33_RemoveAppData (HANDLE *hApp*, PCSTR *pKey*)

_CWORD33_RemoveAppData This API removes the data stored against a key in application framework.

Parameters:

in	*hApp*	HANDLE - Application handle
in	*pKey*	PSTR - Data key

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_SetAppData, _CWORD33_GetAppData

E_CWORD33_Status _CWORD33_SendChild (HANDLE *hApp*, HANDLE *hChildQ*, UI_32 *iCmd*, UI_32 *length*, PCVOID *data*)

Brief

Send a message to the child thread.

Parameters:

in	*hApp*	HANDLE - Handle for Application of parent thread
in	*hChildQ*	HANDLE - Handle for communicate with child thread(returned by _CWORD33_CreateChildThread)
in	*iCmd*	UI_32 - Command of message
in	*length*	UI_32 - Length of send message data
in	*data*	PCVOID - Pointer to message data

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldBuf	Invalid buffer
e_CWORD33_StatusInvldQName	Illegal Message Queue name
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
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

Prerequisite

Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.
Generation/Initialization of Dispatcher for the child thread
(_CWORD33_CreateChildThread, etc.) has been done.

Change of internal state

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

Classification

Public

Type

No match

See also:

CWORD33_SendParent

E_CWORD33_Status _CWORD33_SendParent (HANDLE *hChildApp*, UI_32 *iCmd*, UI_32 *length*, PCVOID *data*)

Brief

Send a message to the parent thread.

Parameters:

in	*hChildApp*	HANDLE - Handle for Application of child thread(get as argument of callback function)
in	*iCmd*	UI_32 - Command of message
in	*length*	UI_32 - Length of send message data
in	*data*	PCVOID - Pointer to message data

Return values:

e_CWORD33_StatusOK	Success

e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldBuf	Invalid buffer
e_CWORD33_StatusInvldQName	Illegal Message Queue name
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
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

Prerequisite

Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.
Generation/Initialization of Dispatcher for the child thread
(_CWORD33_CreateChildThread, etc.) has been done.

Change of internal state

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

Classification

Public

Type

No match

See also:

_CWORD33_SendChild

E_CWORD33_Status _CWORD33_SendPriorityMsg (HANDLE *hService*, UI_32 *uiCmd*, UI_32 *uiLength*, PCVOID *pData*)

_CWORD33_SendPriorityMsg Send message to a service. It places your message at the top of the services message queue. **Applications** should not use this unless absolutely required - use cases must be reviewed by **Framework** Team.

Parameters:

in	*hService*	HANDLE - Handle to the service.
in	*uiCmd*	UI_32 - Protocol message id.
in	*uiLength*	UI_32 - Length of the data to be send.
in	*pData*	PVOID - Pointer to the data.

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_McOpenSender, _CWORD33_McClose, _CWORD33_GetMsgLength, _CWORD33_GetMsgSrc, _CWORD33_GetMsgDataOfSize, _CWORD33_ClearMsgData, _CWORD33_GetDataUSID, _CWORD33_ForwardMessage, _CWORD33_GetSystemInfo

E_CWORD33_Status _CWORD33_SendRequest (HANDLE *hApp*, PCSTR *pServerName*, UI_32 *uiSessionId*, UI_32 *iCmd*, UI_32 *length*, PCVOID *data*)

_CWORD33_SendRequest This API retrieves the session handle from the application framework and sends the request to the server with specific service name and session id. This API will work, if application has previously stored associated session handle using _CWORD33_SetSessionHandle.

Parameters:

in	*hApp*	HANDLE - Application handle
in	*pServerName*	PCSTR - Name of the server
in	*uiSessionId*	UI_32 - session id
in	*iCmd*	UI_32 - Protocol command
in	*length*	UI_32 - size of message data
in	*data*	PCVOID - message data

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_SetSessionHandle, _CWORD33_SendResponse

E_CWORD33_Status _CWORD33_SendStopToNSNPP (HANDLE *hApp*, E_CWORD33_ShutdownType *eShutdownType*, UI_32 *uiStopMsgData* = 0x0)

Type

Pub-Sub

E_CWORD33_Status _CWORD33_SetDeferredSyncResponse (HANDLE *hApp*)

_CWORD33_SetDeferredSyncResponse Set sync response deferred flag

Parameters:

in	*hApp*	HANDLE - Handle to the dispatcher to receive message

Returns:

status E_CWORD33_Status - success or error

E_CWORD33_Status _CWORD33_SetSyncResponseData (HANDLE *hApp*, PVOID *data*, UI_32 *size*)

Brief

API to set the response data of the synchronization communication.

Parameters:

in	*hApp*	HANDLE - Handle for Application
in	*data*	PVOID - Pointer to the response data
in	*size*	UI_32 - The response data size

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	NULL pointer specified

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_InvokeSync

E_CWORD33_Status _CWORD33_SetThreadSpecificData (HANDLE *hApp*, PVOID *data*)

Brief

Set pointer to the application-specific data to the application HANDLE.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*data*	PVOID - Pointer to the application-specific data

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle

Prerequisite

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

Change of internal state

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

Classification

Public

See also:

_CWORD33_GetThreadSpecificData

E_CWORD33_Status _CWORD33_SetUser (HANDLE *hApp*, HANDLE *hUser*)

_CWORD33_SetUser Set the current user for the application in application framework

Parameters:

in	*hApp*	HANDLE - Application handle
in	*hUser*	HANDLE - User handle

Returns:

status E_CWORD33_Status - success or error

E_CWORD33_Status _CWORD33_SimpleDispatcher (PCSTR *cAppName*, const _CWORD33_DefaultCallbackHandler * *CbHandler*, CbFuncPtr *CbShutdown*, BOOL *isChildThread* = TRUE)

Brief

Parameters:

in	*cAppName*	PCSTR - Application/ thread name
in	*CbHandler*	const _CWORD33_DefaultCallbackHandler* - Function that primes the dispatch pump. i.e. start function
in	*CbShutdown*	CbFuncPtr * - Shutdown function. Functions gets called if the dispatcher loop exits.
in	*isChildThread*	BOOL - Default value is TRUE. TRUE - child thread dispatcher else main thread dispatcher

Return values:

Never	does. Unless any callback function returns e_CWORD33_StatusExit
e_CWORD33_StatusNullPointer	NULL pointer
e_CWORD33_StatusFail	Invalid Parameter
e_CWORD33_StatusErrOther	Unknown Error

Prerequisite

Change of internal state

Classification

Public

Type

No match

See also:

_CWORD33_DispatcherWithArguments

E_CWORD33_Status _CWORD33_StartChildThread (HANDLE *hApp*, HANDLE *hChildQ*, UI_32 *length*, PCVOID *data*)

Brief

Send initialize request to the child thread.

Parameters:

in	*hApp*	HANDLE - Handle for Application of parent thread
in	*hChildQ*	HANDLE - Handle for communicate with child thread(returned by _CWORD33_CreateChildThread)
in	*length*	UI_32 - Length of send message data
in	*data*	PCVOID - Pointer to message data

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldBuf	Invalid buffer
e_CWORD33_StatusInvldQName	Illegal Message Queue name
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
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

Prerequisite

Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.
Generation/Initialization of Dispatcher for the child thread
(_CWORD33_CreateChildThread, etc.) has been done.

Change of internal state

Change of internal state is depend on application implements.

Classification

Public

Type

Open Close

See also:

_CWORD33_StopChildThread

E_CWORD33_Status _CWORD33_StopChildThread (HANDLE *hApp*, HANDLE *hChildQ*, UI_32 *length*, PCVOID *data*)

Brief

Send shutdown request to the child thread.

Parameters:

in	*hApp*	HANDLE - Handle for Application of parent thread
in	*hChildQ*	HANDLE - Handle for communicate with child thread(returned by _CWORD33_CreateChildThread)
in	*length*	UI_32 - Length of send message data
in	*data*	PCVOID - Pointer to message data

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusInvldBuf	Invalid buffer
e_CWORD33_StatusInvldQName	Illegal Message Queue name
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
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

Prerequisite

Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.
Generation/Initialization of Dispatcher for the child thread
(_CWORD33_CreateChildThread, etc.) has been done.

Change of internal state

Change of internal state is depend on application implements.

Classification

Public

See also:

_CWORD33_StartChildThread _CWORD33_DestroyChildThread

E_CWORD33_Status _CWORD33_SubscribeNotificationsWithCallback (HANDLE *hApp*, const _CWORD33_NotificationCallbackHandler * *pNtfyHandler*, UI_32 *uiHandlerCount*)

Brief

API to send a message to the notification service in order to register multiple Notification of list format.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pNtfyHandler*	_CWORD33_NotificationCallbackHandler - Pointer to a structure of the Notification Handler
in	*uiHandlerCount*	UI_32 - Notification Handler number

_CWORD33_NotificationCallbackHandler Structure

```
1 #define MAX_STRING_SIZE_NOTIFICATION  64
2 typedef struct __CWORD33_NotificationCallbackHandler
3 {
4       CHAR cNotification[MAX_STRING_SIZE_NOTIFICATION]; /* Notification Name         */
5       CbFuncPtr callBack;                          /* Callback function pointer  */
6 }_CWORD33_NotificationCallbackHandler;
```

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.)

Prerequisite

NS_NPPService of the process has been started.
Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of internal state

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

Classification

Public

See also:

_CWORD33_NPRegisterNotification, _CWORD33_NPUnRegisterNotification,
_CWORD33_NPUnRegisterNotifications, _CWORD33_NPPublishNotification,
_CWORD33_SubscribeNotificationWithCallback,
_CWORD33_UnsubscribeNotificationWithCallback,
_CWORD33_UnsubscribeNotificationsWithCallback

E_CWORD33_Status _CWORD33_SubscribeNotificationWithCallback (HANDLE *hApp*, PCSTR *pNotification*, CbFuncPtr *fpOnCmd*)

Brief

API to set the Callback information at the time of Notification received Dispatcher.

Parameters:

in	*hApp*	HANDLE - Application handle
in	*pNotification*	PCSTR - Notification String.
in	*fpOnCmd*	CbFuncPtr - Callback function that will be called on receiving pNotification.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusDuplicate	Duplication error of entry
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.)

Prerequisite

NS_NPPService of the process has been started.
Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of internal state

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

Classification

Public

See also:

_CWORD33_NPRegisterNotification, _CWORD33_NPUnRegisterNotification,
_CWORD33_NPUnRegisterNotifications, _CWORD33_NPPublishNotification,
_CWORD33_UnsubscribeNotificationWithCallback,

_CWORD33_SubscribeNotificationsWithCallback,
_CWORD33_UnsubscribeNotificationsWithCallback

E_CWORD33_Status _CWORD33_UnsubscribeNotificationsWithCallback (HANDLE *hApp*, const _CWORD33_NotificationCallbackHandler * *pNtfyHandler*, UI_32 *uiHandlerCount*)

Brief

API to bulk delete the Callback information at the time of multiple Notification received from the Dispatcher.

Parameters:

in	*hApp*	HANDLE - HANDLE for Application
in	*pNtfyHandler*	_CWORD33_NotificationCallbackHandler - Pointer to a structure of the Notification Handler
in	*uiHandlerCount*	UI_32 - Notification Handler number

_CWORD33_NotificationCallbackHandler Structure

```
1 #define MAX_STRING_SIZE_NOTIFICATION  64
2 typedef struct __CWORD33_NotificationCallbackHandler
3 {
4       CHAR cNotification[MAX_STRING_SIZE_NOTIFICATION]; /* Notification Name         */
5       CbFuncPtr callBack;                          /* Callback function pointer  */
6 }_CWORD33_NotificationCallbackHandler;
```

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.)

Prerequisite

NS_NPPService of the process has been started.
Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of internal state

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

Classification

Public

See also:

_CWORD33_NPRegisterNotification, _CWORD33_NPUnRegisterNotification,
_CWORD33_NPRegisterNotifications, _CWORD33_NPUnRegisterNotifications,
_CWORD33_NPPublishNotification, _CWORD33_SubscribeNotificationWithCallback,
_CWORD33_SubscribeNotificationsWithCallback,
_CWORD33_UnsubscribeNotificationWithCallback

E_CWORD33_Status _CWORD33_UnsubscribeNotificationWithCallback (HANDLE *hApp*, PCSTR *pNotification*)

Brief

API to detach a notification callback from the dispatcher.

Parameters:

in	*hApp*	HANDLE - Application handle
in	*pNotification*	PCSTR - Notification String.

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
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.)

Prerequisite

NS_NPPService of the process has been started.
Generation/Initialization of Dispatcher for the Application
(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of internal state

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

Classification

Public

See also:

_CWORD33_NPRegisterNotification, _CWORD33_NPUnRegisterNotification,
_CWORD33_NPRegisterNotifications, _CWORD33_NPUnRegisterNotifications,
_CWORD33_NPPublishNotification, _CWORD33_SubscribeNotificationWithCallback,
_CWORD33_SubscribeNotificationsWithCallback,
_CWORD33_UnsubscribeNotificationsWithCallback

Statemachine

class C_CWORD33_Action
class C_CWORD33_CompositeState
class C_CWORD33_ConditionConnector
class C_CWORD33_DeepHistoryState
This class implements the additional functionality supported by HSM Shallow History state. class CEventData
class C_CWORD33_EventFactory
This class defines the events used by the statemachine framework. class C_CWORD33_ExternalTransition
class CNullPointerException
Defines the null pointer exception. class C_CWORD33_Guard
class C_CWORD33_HistoryState
this file has the C_CWORD33_HistoryState class definitions class C_CWORD33_HSM
class C_CWORD33_HSMFramework
class C_CWORD33_HSMParentFramework
class C_CWORD33_HSMChildFramework
class C_CWORD33_InternalTransition
This class provides the interface for defining a reaction for an event. class C_CWORD33_LeafState
This class implements the additional functionality supported by HSM Leafstate. class C_CWORD33_LocalTransition
This class provides the interface for local transition over event. class C_CWORD33_OrthogonalState
class C_CWORD33_Reaction
This class provides the interface for defining a reaction for an event. class C_CWORD33_ShallowHistoryState
This class implements the additional functionality supported by HSM Shallow History state. class C_CWORD33_State
class C_CWORD33_Transition
This class provides the interface for reacting local transition and external transition. class CNSSharedMem
this file has the C_CWORD33_Guard class definitions typedef boost::shared_ptr< CEventData > **CEventDataPtr**
typedef std::map< std::string, C_CWORD33_State * > **ChildStateList**
typedef std::map< std::string, C_CWORD33_State * >::iterator **StateIterator**
E_CWORD33_Status _CWORD33_HSMDispatcher (PCSTR cAppName)
E_CWORD33_Status _CWORD33_HSMDispatcherWithArguments (PCSTR cAppName, int argc, char *argv[], const _CWORD33_DefaultCallbackHandler *CbHandler, C_CWORD33_HSMFramework *f_p_CWORD33_HSM=NULL, CustomCommandLineOptions *cmdLineOptions=NULL)
void _CWORD33_SetHSMType (EUserChangeOptions f_eHSMType)
void _CWORD33_EnableAutoPublishServiceAvailable ()
void _CWORD33_DisableAutoPublishServiceAvailable ()
VOID _CWORD33_HSMEnableWaitInStoppingState ()

VOID _CWORD33_HSMDisableWaitInStoppingState ()
E_CWORD33_Status _CWORD33_SetStateMachine (HANDLE hApp, C_CWORD33_HSM *f_p_CWORD33_HSM)
E_CWORD33_Status _CWORD33_AttachHSMEventToDispatcher (HANDLE hApp, PCSTR pServiceName, UI_32 iCmd, UI_32 iEvent, HANDLE hSession=NULL)
E_CWORD33_Status _CWORD33_AttachHSMEventsToDispatcher (HANDLE hApp, PCSTR pServiceName, const _CWORD33_ProtocolEvent *pEventIds, UI_32 uiEventCount, HANDLE hSession=NULL)
E_CWORD33_Status _CWORD33_NPHSMRegisterNotificatsionEvent (HANDLE hApp, PCSTR pNotification, const UI_32 max_length, const E_CWORD33_NotificationType persType)
E_CWORD33_Status _CWORD33_NPHSMRegisterNotificationsEvents (HANDLE hApp, const _CWORD33_NotificationsList *pList, UI_32 uiListLength)
E_CWORD33_Status _CWORD33_NPHSMUnRegisterNotificationEvent (HANDLE hApp, PCSTR pNotification)
E_CWORD33_Status _CWORD33_SubscribeNotificationWithHSMEvent (HANDLE hApp, PCSTR pNotification, UI_32 iEventId)
E_CWORD33_Status _CWORD33_SubscribeNotificationsWithHSMEvent (HANDLE hApp, const _CWORD33_NotificationEvent *pNtfyEvent, UI_32 uiEventCount)
E_CWORD33_Status _CWORD33_UnsubscribeNotificationsWithHSMEvent (HANDLE hApp, const _CWORD33_NotificationEvent *pNtfyHandler, UI_32 uiHandlerCount)
E_CWORD33_Status _CWORD33_UnsubscribeNotificationWithHSMEvent (HANDLE hApp, PCSTR pNotification)
E_CWORD33_Status _CWORD33_AttachParentHSMEventsToDispatcher (HANDLE hChildApp, const _CWORD33_ProtocolEvent *pEventIds, UI_32 uiEventCount)
E_CWORD33_Status _CWORD33_DetachParentHSMEventsFromDispatcher (HANDLE hChildApp, const PUI_32 puiEventArray, UI_32 uiEventCount)
E_CWORD33_Status _CWORD33_DetachHSMEventsFromDispatcher (HANDLE hApp, PCSTR pServiceName, const PUI_32 puiCmdArray, UI_32 uiCmdCount, HANDLE hSession=NULL)
E_CWORD33_Status _CWORD33_DetachHSMEventFromDispatcher (HANDLE hApp, PCSTR pServiceName, UI_32 iCmd, HANDLE hSession=NULL)
C_CWORD33_HSMFramework * _CWORD33_GetStateMachine (HANDLE hApp)
E_CWORD33_Status _CWORD33_SubscribeToSessionEventWithHSMEvent (HANDLE hApp, UI_32 uiEventId, UI_32 uiHSMEventId, HANDLE hSession)
E_CWORD33_Status _CWORD33_SubscribeToSessionEventsWithHSMEvents (HANDLE hApp, const _CWORD33_ProtocolEvent *pEventIds, UI_32 uiEventCount, HANDLE hSession)
E_CWORD33_Status _CWORD33_UnSubscribeSessionEventWithHSMEvent (HANDLE hApp, UI_32 uiEventId, HANDLE hSession)
E_CWORD33_Status _CWORD33_UnSubscribeSessionEventsWithHSMEvents (HANDLE hApp, PUI_32 pEventsArray, UI_32 uiListSize, HANDLE hSession)
E_CWORD33_Status _CWORD33_CreateStateMachine (HANDLE hApp)
E_CWORD33_Status _CWORD33_HSMOnLoadData (HANDLE hApp)
E_CWORD33_Status _CWORD33_HSMOnStopIns_CWORD33_Run (HANDLE hApp)
#define **CREATE_STATE**(class_name) C## class_name *l_p## class_name = new C## class_name(#class_name);
#define **_CWORD33__CONNECT_EVENT**(state, eventid, reaction)
#define **_CWORD33__CONNECT_LOCAL_EVENT**(state, eventid, reaction)
#define _CWORD33__CONNECT_DEFERREDEVENT(state, eventid)

Connect framework event to reaction and add to state as deferred event.

#define **_CWORD33__CONNECT_DEFAULTSTATE**(child) (_CWORD33_GetStateMachine(hApp))->_CWORD33_FrameworkConnect(l_p## child, TRUE);

#define **_CWORD33__CONNECT_STATE**(child) (_CWORD33_GetStateMachine(hApp))->_CWORD33_FrameworkConnect(l_p## child);

#define **CONNECT_DEFAULTSTATE**(parent, child) (_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## parent, l_p## child, TRUE);

#define **CONNECT_STATE**(parent, child) (_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## parent, l_p## child);

#define **CONNECT_ORTHOGONAL_REGION**(orthogonalstate, orthogonalregion) (_CWORD33_GetStateMachine(hApp))->_CWORD33_ConnectOrthogonal(l_p## orthogonalstate, l_p## orthogonalregion);

#define CONNECT_DEFERREDEVENT(state, eventid) _CWORD33_GetStateMachine(hApp)->_CWORD33_Connect(l_p## state, _## eventid, NULL, #eventid, TRUE);

connect the deferred event and reactions and associate them with the state

#define **CONNECT_EVENT**(state, eventid, reaction) (_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## state, _## eventid, l_pTrn## reaction, #eventid);

#define CONNECT_LOCAL_EVENT(state, eventid, reaction) (_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## state, _## eventid, l_pLocalTrn## reaction, #eventid);

connect event to reaction and add to state

#define **PRINTSTATEMACHINE**() (_CWORD33_GetStateMachine(hApp))->_CWORD33_PrintAllStates();

#define CONNECT_CWORD33_EVENT(state, eventid, reaction)

Connect application defined event to application defined reaction and add to framework state.

#define CONNECT_CWORD33_DEFERREDEVENT(state, eventid)

Connect application defined deferred event to framework state.

#define **_CWORD33__CONNECT_APP_EVENT**(state, eventid, reaction)

#define **_CWORD33__CONNECT_APP_DEFERREDEVENT**(state, eventid)

#define **ADD_SHALLOWHISTORYSTATE**(parent)

#define **ADD_DEEPHISTORYSTATE**(parent)

#define CONNECT_DEEPHISTORYEVENT(state, eventid, reaction) (_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## state, _## eventid, l_pTrn##reaction##DEEPHISTORYSTATE, #eventid);

connect deep history event to reaction and add to state

#define **CONNECT_SHALLOWHISTORYEVENT**(state, eventid, reaction) (_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## state, _## eventid, l_pTrn##reaction##SHALLOWHISTORYSTATE, #eventid);

enum _EUserChangeOptions { **eUserchangeIgnore** = ***, **eUserchangeReInit**, **eUserchangeRetPrevState** }*defines options for integrating user change state*

typedef enum _EUserChangeOptions EUserChangeOptions

defines options for integrating user change state

#define CHKNULL(x) if (NULL == x) throw CNullPointerException();

Macro to check pointer for null value and throw null pointer exception.

#define **SHALLOWHISTORYSTATE** "ShallowHistory"

#define **DEEPHISTORYSTATE** "DeepHistory"
#define DEFINE_STATEMACHINE(class_name)

Defines the state machine class.

#define DEFINE_LEAFSTATE(class_name)

Defines the leafstate class that does not override entry and exit of base state.

#define DEFINE_COMPOSITESTATE(class_name)

Defines the Composite state that does not override entry and exit of base state.

#define DEFINE_ORTHOGONALSTATE(class_name)

Defines the Orthogonal state that does not override entry and exit of base state.

#define **DEFINE_LEAFSTATE_OVERRIDE**(class_name)
#define **DEFINE_COMPOSITESTATE_OVERRIDE**(class_name)
#define **DEFINE_ORTHOGONALSTATE_OVERRIDE**(class_name)
#define **DEFINE_INTERNALTRANSITION**(class_name)
#define **DEFINE_EVENT**(event_name, value) static const UI_32 _## event_name = C_CWORD33_EventFactory::ev_CWORD33_EventLimit + value;
#define **DEFINE_ACTION**(class_name)
#define **CREATE_INTERNALTRANSITION**(class_name)
#define **CREATE_EXTERNALTRANSITION**(target_state)
#define CREATE_CONDITIONCONNECTOR(connector_name)

creates the instance of the external transition

#define CREATE_LOCALTRANSITION(target_state)

creates the instance of the local transition

#define **CREATE_ORTHOGONALREGION**(name) C_CWORD33_CompositeState *l_p## name = new C_CWORD33_CompositeState(#name);
#define CREATE_ACTION(class_name)

create action for external transition

#define _**CWORD33__EVENT**(eventid) C_CWORD33_HSMParentFramework::_## eventid
#define **EVENT**(eventid) _## eventid
#define IMPLEMENT_INTERNALTRANSITION(name) C_CWORD33_State* C## name::_CWORD33_Reaction(C_CWORD33_State* f_pSourceState, CEventDataPtr f_pData)

_CWORD33_Reaction function signature for internal transition class

#define **IMPLEMENT_ACTION**(action_name) void C## action_name::_CWORD33_Action(C_CWORD33_State *f_pSourceState, C_CWORD33_State *f_pTargetState, CEventDataPtr f_pEventData)
#define **DEFINE_GUARD**(guard_name)
#define **CREATE_GUARD**(guard_name) C_CWORD33_Guard *l_pGrd## guard_name = new C## guard_name(#guard_name);
#define **CONNECT_GUARD**(transition, guard) l_pTrn## transition->_CWORD33_SetGuard(l_pGrd## guard);
#define **CONNECT_CONDITION**(connector_name, guard, target_state)
#define **IMPLEMENT_GUARD**(guard_name) BOOL C## guard_name::_CWORD33_Evaluate()
#define **CREATE_SHALLOWHISTORYTRANSITION**(parent)
#define CREATE_DEEPHISTORYTRANSITION(parent)

creates the instance of the deep history transition

#define **CONNECT_CONDITION_SHALLOWHISTORYTRANSITION**(connector_name, guard, target_state)
#define **CONNECT_CONDITION_DEEPHISTORYTRANSITION**(connector_name, guard, target_state)
#define **CONNECT_ACTION**(transition, action_name)
#define **_CWORD33__PRINT_HSM**(x)
#define **PRINT_HSM**()
typedef boost::shared_ptr< CEventData > **CEventDataPtr**
HANDLE _CWORD33_CreateHSMChildThread (HANDLE hApp, PCSTR childName, CbFuncPtr CbInitialize, CbFuncPtr CbShutdown, CbFuncPtr CbCreateStateMachine)
HANDLE _CWORD33_CreateHSMChildThreadWithPriority (HANDLE hApp, PCSTR childName, CbFuncPtr CbInitialize, CbFuncPtr CbShutdown, CbFuncPtr CbCreateStateMachine, SI_32 schedPrio)
HANDLE _CWORD33_CreateHSMChildThreadWithAttribute (HANDLE hApp, PCSTR childName, CbFuncPtr CbInitialize, CbFuncPtr CbShutdown, CbFuncPtr CbCreateStateMachine, const _CWORD33_ChildThreadAttr *attr)
typedef std::vector< C_CWORD33_State * > **OrthogonalRegionList**
typedef boost::shared_ptr< CEventData > **CEventDataPtr**
typedef std::map< UI_32, C_CWORD33_Reaction * > EventReactionList
defines the map of event reactions
typedef std::vector< UI_32 > **DeferredEventList**
typedef std::map< UI_32, std::string > EventNameList
defines the map of event names
typedef std::map< UI_32, C_CWORD33_Reaction * >::iterator EventReactionIterator
defines the iterator for event reaction map
typedef std::vector< CEventDataPtr > **EventInfoList**

Detailed Description

Macro Definition Documentation

#define _CWORD33__CONNECT_APP_DEFERREDEVENT(state, eventid)

Value:(_CWORD33_GetStateMachine(hApp))->_CWORD33_FrameworkConnect(C_CWORD33_HSMParentFramework::e## state, _## eventid, \
NULL, #eventid, TRUE);

#define _CWORD33__CONNECT_APP_EVENT(state, eventid, reaction)

Value:(_CWORD33_GetStateMachine(hApp))->_CWORD33_FrameworkConnect(C_CWORD33_HSMParentFramework::e## state, _## eventid, \
l_pTrn## reaction, #eventid);

#define _CWORD33__CONNECT_DEFERREDEVENT(state, eventid)

Value:(_CWORD33_GetStateMachine(hApp))->_CWORD33_FrameworkConnect(C_CWORD33_HSMParentFramework::e## state, \
C_CWORD33_HSMParentFramework::_## eventid, NULL, #eventid, TRUE);

Connect framework event to reaction and add to state as deferred event.

#define _CWORD33__CONNECT_EVENT(state, eventid, reaction)

Value:(_CWORD33_GetStateMachine(hApp))->_CWORD33_FrameworkConnect(C_CWORD33_HSMParentFramework::e## state, \
C_CWORD33_HSMParentFramework::_## eventid, l_pTrn## reaction, #eventid);

#define _CWORD33__CONNECT_LOCAL_EVENT(state, eventid, reaction)

Value:(_CWORD33_GetStateMachine(hApp))->_CWORD33_FrameworkConnect(C_CWORD33_HSMParentFramework::e## state, \
C_CWORD33_HSMParentFramework::_## eventid, l_pLocalTrn## reaction, #eventid);

#define ADD_DEEPHISTORYSTATE(parent)

Value:C_CWORD33_DeepHistoryState *l_p##parent##DEEPHISTORYSTATE = new C_CWORD33_DeepHistoryState(DEEPHISTORYSTATE);\
(_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## parent, l_p##parent##DEEPHISTORYSTATE);\
__CHKNULL(l_p##parent##DEEPHISTORYSTATE);\
l_p##parent##DEEPHISTORYSTATE->SetDefaultHistory();

#define ADD_SHALLOWHISTORYSTATE(parent)

Value:C_CWORD33_ShallowHistoryState *l_p##parent##SHALLOWHISTORYSTATE = new C_CWORD33_ShallowHistoryState(SHALLOWHISTORYSTATE);\
(_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## parent, l_p##parent##SHALLOWHISTORYSTATE);\
__CHKNULL(l_p##parent##SHALLOWHISTORYSTATE);\
l_p##parent##SHALLOWHISTORYSTATE->SetDefaultHistory();

#define CONNECT__CWORD33__DEFERREDEVENT(state, eventid)

Value:(_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## state, C_CWORD33_HSMParentFramework::_## eventid, \
NULL, #eventid, TRUE);

Connect application defined deferred event to framework state.

#define CONNECT__CWORD33__EVENT(state, eventid, reaction)

Value:(_CWORD33_GetStateMachine(hApp))->_CWORD33_Connect(l_p## state, C_CWORD33_HSMParentFramework::_## eventid, \
l_pTrn## reaction, #eventid);

Connect application defined event to application defined reaction and add to framework state.

#define CONNECT_ACTION(transition, action_name)

Value:CHKNULL(l_pTrn## action_name); \
  CHKNULL(l_pTrn## transition); \
  l_pTrn## transition->_CWORD33_AddAction(l_pTrn## action_name); \

#define CONNECT_CONDITION(connector_name, guard, target_state)

Value:CHKNULL(l_pTrn## connector_name)\
  l_pTrn## connector_name->_CWORD33_AddCondition(l_pGrd## guard , l_p## target_state); \

#define CONNECT_CONDITION_DEEPHISTORYTRANSITION(connector_name, guard, target_state)

Value:CHKNULL(l_pTrn## connector_name)\
  l_pTrn## connector_name->_CWORD33_AddCondition(l_pGrd## guard , l_p## target_state##DEEPHISTORYSTATE); \

#define CONNECT_CONDITION_SHALLOWHISTORYTRANSITION(connector_name, guard, target_state)

Value:CHKNULL(l_pTrn## connector_name)\
  l_pTrn## connector_name->_CWORD33_AddCondition(l_pGrd## guard , l_p## target_state##SHALLOWHISTORYSTATE); \

#define CREATE_ACTION(class_name)

Value:C## class_name *l_pTrn## class_name = new C## class_name(#class_name); \
  CHKNULL(l_pTrn## class_name);

create action for external transition

#define CREATE_CONDITIONCONNECTOR(connector_name)

Value:C_CWORD33_ConditionConnector *l_pTrn## connector_name = new C_CWORD33_ConditionConnector(#connector_name); \
  CHKNULL(l_pTrn## connector_name);

creates the instance of the external transition

#define CREATE_DEEPHISTORYTRANSITION(parent)

Value:C_CWORD33_ExternalTransition *l_pTrn##parent##DEEPHISTORYSTATE = \
  new C_CWORD33_ExternalTransition(l_p##parent##DEEPHISTORYSTATE); \
  CHKNULL(l_pTrn##parent##DEEPHISTORYSTATE);

creates the instance of the deep history transition

#define CREATE_EXTERNALTRANSITION(target_state)

Value:C_CWORD33_ExternalTransition *l_pTrn## target_state = new C_CWORD33_ExternalTransition(l_p## target_state); \
  CHKNULL(l_pTrn## target_state);

#define CREATE_INTERNALTRANSITION(class_name)

Value:C## class_name *l_pTrn## class_name = new C## class_name(); \
__CHKNULL(l_pTrn## class_name);

#define CREATE_LOCALTRANSITION(target_state)

Value:C_CWORD33_LocalTransition *l_pLocalTrn## target_state = new C_CWORD33_LocalTransition(l_p## target_state); \
__CHKNULL(l_pLocalTrn## target_state);

creates the instance of the local transition

#define CREATE_SHALLOWHISTORYTRANSITION(parent)

Value:C_CWORD33_ExternalTransition *l_pTrn##parent##SHALLOWHISTORYSTATE = \
new C_CWORD33_ExternalTransition(l_p##parent##SHALLOWHISTORYSTATE); \
__CHKNULL(l_pTrn##parent##SHALLOWHISTORYSTATE);

#define DEFINE_ACTION(class_name)

Value:class C## class_name: public C_CWORD33_Action{ \
public: \
C## class_name(std::string f_strName):C_CWORD33_Action(f_strName) {} \
VOID _CWORD33_Action(C_CWORD33_State *f_pSourceState, C_CWORD33_State *f_pTargetState, CEventDataPtr f_pEventData); \
};

#define DEFINE_COMPOSITESTATE(class_name)

Value:class C## class_name : public C_CWORD33_CompositeState { \
public: \
C## class_name(std::string f_pName):C_CWORD33_CompositeState(f_pName) {} };

Defines the Composite state that does not override entry and exit of base state.

#define DEFINE_COMPOSITESTATE_OVERRIDE(class_name)

Value:class C## class_name : public C_CWORD33_CompositeState{ \
public: \
C## class_name(std::string f_pName); \
virtual ~C## class_name(); \
__E_CWORD33_Status _CWORD33_OnEntry(CEventDataPtr f_pEventData); \
__E_CWORD33_Status _CWORD33_OnExit(CEventDataPtr f_pEventData); \
};

#define DEFINE_GUARD(guard_name)

Value:class C## guard_name : public C_CWORD33_Guard{ \
public: \
C## guard_name(std::string f_pName):C_CWORD33_Guard(f_pName) {} \
virtual BOOL _CWORD33_Evaluate();\
};

#define DEFINE_INTERNALTRANSITION(class_name)

Value:class C## class_name : public C_CWORD33_InternalTransition{ \
public: \
virtual C_CWORD33_State* _CWORD33_Reaction(C_CWORD33_State* f_pSourceState, CEventDataPtr f_pData); \
};

#define DEFINE_LEAFSTATE(class_name)

Value:class C## class_name : public C_CWORD33_LeafState { \
public: \
C## class_name(std::string f_pName):C_CWORD33_LeafState(f_pName) {} };

Defines the leafstate class that does not override entry and exit of base state.

#define DEFINE_LEAFSTATE_OVERRIDE(class_name)

Value:class C## class_name : public C_CWORD33_LeafState{ \
public: \
C## class_name(std::string f_pName); \
virtual ~C## class_name(); \
E_CWORD33_Status _CWORD33_OnEntry(CEventDataPtr f_pEventData); \
E_CWORD33_Status _CWORD33_OnExit(CEventDataPtr f_pEventData); \
};

#define DEFINE_ORTHOGONALSTATE(class_name)

Value:class C## class_name : public C_CWORD33_OrthogonalState { \
public: \
C## class_name(std::string f_pName):C_CWORD33_OrthogonalState(f_pName) {} };

Defines the Orthogonal state that does not override entry and exit of base state.

#define DEFINE_ORTHOGONALSTATE_OVERRIDE(class_name)

Value:class C## class_name : public C_CWORD33_OrthogonalState{ \
public: \
C## class_name(std::string f_pName); \
virtual ~C## class_name(); \
E_CWORD33_Status _CWORD33_OnEntry(CEventDataPtr f_pEventData); \
E_CWORD33_Status _CWORD33_OnExit(CEventDataPtr f_pEventData); \
};

#define DEFINE_STATEMACHINE(class_name)

Value:class C## class_name : public C_CWORD33_HSM{ \
public: \
C## class_name(); \
virtual ~C## class_name(); \
E_CWORD33_Status _CWORD33_Create(PVOID f_pEventData = NULL); \
};

Defines the state machine class.

Function Documentation

E_CWORD33_Status _CWORD33_AttachHSMEventsToDispatcher (HANDLE *hApp*, PCSTR *pServiceName*, const _CWORD33_ProtocolEvent * *pEventIds*, UI_32 *uiEventCount*, HANDLE *hSession* = NULL)

Brief

Register events in dispatcher.

Parameters:

in	*hApp*	HANDLE hApp - Application framework handle
in	*pServiceName*	PCSTR pServiceName - Service Name
in	*pEventIds*	const _CWORD33_ProtocolEvent *pEventIds - structure mapping event ids with the messages
in	*uiEventCount*	UI_32 uiEventCount - no of events
in	*hSession*	HANDLE hSession - Session Handle.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_HSMDispatcherWithArguments has been done.

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_AttachHSMEventToDispatcher

_CWORD33_AttachHSMEventsToDispatcher Registers a multiple event with the dispatcher for a given service.

Parameters:

in	*hApp*	HANDLE - Application framework handle
in	*pServiceName*	PCSTR - Service Name
in	*pEventIds*	const _CWORD33_ProtocolEvent* - structure mapping event ids with the messages
in	*uiEventCount*	UI_32 - no of events
in	*hSession*	HANDLE - Session Handle.

Returns:

status E_CWORD33_Status - success or error

See also:

E_CWORD33_Status _CWORD33_AttachHSMEventToDispatcher (HANDLE *hApp*, PCSTR *pServiceName*, UI_32 *iCmd*, UI_32 *iEvent*, HANDLE *hSession* = NULL)

Brief

Register an event in dispatcher

Parameters:

in	*hApp*	HANDLE hApp - Application framework handle
in	*pServiceName*	PCSTR pServiceName - Service Name
in	*iCmd*	UI_32 iCmd - service protocol command/request/acknowledgment ID
in	*iEvent*	UI_32 iEvent - event to be posted to active state on receiving iCmd from pServiceName.
in	*hSession*	HANDLE hSession - Session Handle.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_HSMDispatcherWithArguments has been done.

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_AttachHSMEventsToDispatcher

_CWORD33_AttachHSMEventToDispatcher Registers a single event with the dispatcher for a given service.

Parameters:

in	*hApp*	HANDLE - Application framework handle
in	*pServiceName*	PCSTR - Service Name
in	*iCmd*	UI_32 - service protocol command/request/acknowledgment ID
in	*iEvent*	UI_32 - event to be posted to active state on receiving iCmd from pServiceName.
in	*hSession*	HANDLE - Session Handle.

Returns:

status E_CWORD33_Status - success or error

See also:

E_CWORD33_Status _CWORD33_AttachParentHSMEventsToDispatcher (HANDLE *hChildApp*, const _CWORD33_ProtocolEvent * *pEventIds*, UI_32 *uiEventCount*)

Brief

Registers a multiple HSM event with the dispatcher for a given service.

Parameters:

in	*hApp*	HANDLE hApp - Child Thread handle
in	*pEventIds*	const _CWORD33_ProtocolEvent *pEventIds - structure mapping event ids with the messages
in	*uiEventCount*	UI_32 uiEventCount - no of events

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_AttachParentHSMEventsToDispatcher Registers a multiple HSM event with the dispatcher for a given service.

Parameters:

in	*hChildApp*	HANDLE - Application framework handle
in	*pEventIds*	const _CWORD33_ProtocolEvent* - structure mapping event ids with the messages
in	*uiEventCount*	UI_32 - no of events

Returns:

status E_CWORD33_Status - success or error

See also:

HANDLE _CWORD33_CreateHSMChildThread (HANDLE *hApp*, PCSTR *childName*, CbFuncPtr *CbInitialize*, CbFuncPtr *CbShutdown*, CbFuncPtr *CbCreateStateMachine*)

Brief

Start a subordinate dispatcher in a separate thread

Parameters:

in	*hApp*	HANDLE - parent framework HANDLE
in	*childName*	PCSTR - Name to give to the child object
in	*CbInitialize*	CbFuncPtr - Initialization callback - use this to initialize and configure the child's dispatcher.
in	*CbShutdown*	CbFuncPtr - Shutdown callback - use this to shutdown and close the child The correct way for child to exit dispatch loop is to return e_CWORD33_StatusExit directly from this callback, or from some other interaction triggered by this callback
in	*CbCreateStateMachine*	CbFuncPtr - StateMachine callback - use this to add new states, events and reactions in the Thread statemachine.

Return values:

HANDLE	create thread with Priority success
NULL	create thread with Priority failed

Preconditions

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

Change of the internal state

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

Classification

Public

Type

sync only

See also:

HANDLE _CWORD33_CreateHSMChildThreadWithAttribute (HANDLE *hApp*, PCSTR *childName*, CbFuncPtr *CbInitialize*, CbFuncPtr *CbShutdown*, CbFuncPtr *CbCreateStateMachine*, const _CWORD33_ChildThreadAttr * *attr*)

Summary

Start a subordinate dispatcher in a separate thread with given attribute

Parameters:

in	*hApp*	HANDLE - parent framework HANDLE
in	*childName*	PCSTR - Name to give to the child object
in	*CbInitialize*	CbFuncPtr - Initialization callback - use this to initialize and configure the child's dispatcher.
in	*CbShutdown*	CbFuncPtr - Shutdown callback - use this to shutdown and close the child The correct way for child to exit dispatch loop is to return e_CWORD33_StatusExit directly from this callback, or from some other interaction triggered by this callback
in	*CbCreateStateMachine*	CbFuncPtr - StateMachine callback - use this to add new states, events and reactions in the Thread statemachine.
in	*attr*	const _CWORD33_ChildThreadAttr * - given attribute to create thread

Return values:

HANDLE	create thread with Priority success
NULL	create thread with Priority failed

Preconditions - Generation/Initialization of Dispatcher for the Application

(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

HANDLE _CWORD33_CreateHSMChildThreadWithPriority (HANDLE *hApp*, PCSTR *childName*, CbFuncPtr *CbInitialize*, CbFuncPtr *CbShutdown*, CbFuncPtr *CbCreateStateMachine*, SI_32 *schedPrio*)

Summary

Start a subordinate dispatcher in a separate thread with given Priority

Parameters:

in	*hApp*	HANDLE - parent framework HANDLE
in	*childName*	PCSTR - Name to give to the child object
in	*CbInitialize*	CbFuncPtr - Initialization callback - use this to initialize and configure the child's dispatcher.
in	*CbShutdown*	CbFuncPtr - Shutdown callback - use this to shutdown and close the child The correct way for child to exit

		dispatch loop is to return e_CWORD33_StatusExit directly from this callback, or from some other interaction triggered by this callback
in	*CbCreateStateMachine*	CbFuncPtr - StateMachine callback - use this to add new states, events and reactions in the Thread statemachine.
in	*schedPrio*	SI_32 - given Priority for create thread

Return values:

HANDLE	create thread with Priority success
NULL	create thread with Priority failed

Preconditions - Generation/Initialization of Dispatcher for the Application

(_CWORD33_CreateDispatcherWithoutLoop, etc.) has been done.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

E_CWORD33_Status _CWORD33_CreateStateMachine (HANDLE *hApp*)

_CWORD33_CreateStateMachine The application state machine is created in this interface. User defined events can also be connected to the framework states in this interface.

Brief

It's a callback function to create state machine used by application.

Parameters:

in	*hApp*	HANDLE - handle to application

Return values:

e_CWORD33_StatusOK	-successed
Except	e_CWORD33_StatusOK - failure

Prerequisite

None

Inside state change

None

Classification

Public

See also:

_CWORD33_DispatcherWithArguments, _CWORD33_Dispatcher, _CWORD33_CreateDispatcherWithoutLoop

E_CWORD33_Status _CWORD33_DetachHSMEventFromDispatcher (HANDLE *hApp*, PCSTR *pServiceName*, UI_32 *iCmd*, HANDLE *hSession* = NULL)

Brief

detach an event from the dispatcher

Parameters:

in	*hApp*	HANDLE hApp - Application framework handle
in	*pServiceName*	PCSTR pServiceName - Service Name
in	*iCmd*	UI_32 iCmd - service protocol command/request/acknowledgment ID
in	*hSession*	HANDLE hSession - Session Handle.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle
e_CWORD33_StatusFail	process failed

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_HSMDispatcherWithArguments has been done.

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_DetachHSMEventsFromDispatcher

_CWORD33_DetachHSMEventFromDispatcher API to detach a notification HSM event from the dispatcher.

Parameters:

in	*hApp*	HANDLE - Application framework handle
in	*pServiceName*	PCSTR - Service Name
in	*iCmd*	UI_32 - service protocol command/request/acknowledgment ID

in	*hSession*	HANDLE - Session Handle.

Returns:

status E_CWORD33_Status - success or error

See also:

E_CWORD33_Status _CWORD33_DetachHSMEventsFromDispatcher (HANDLE *hApp*, PCSTR *pServiceName*, const PUI_32 *puiCmdArray*, UI_32 *uiCmdCount*, HANDLE *hSession* = NULL)

Brief

detach multiple events from the dispatcher

Parameters:

in	*hApp*	HANDLE hApp - Application framework handle
in	*pServiceName*	PCSTR pServiceName - Service Name
in	*puiCmdArray*	const PUI_32 puiCmdArray - Array of commands
in	*uiCmdCount*	UI_32 uiCmdCount - Number of Commands
in	*hSession*	HANDLE hSession - Session Handle.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle
e_CWORD33_StatusFail	process failed
e_CWORD33_StatusInvldParam	invalid parameter

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_HSMDispatcherWithArguments has been done.

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_DetachHSMEventFromDispatcher

_CWORD33_DetachHSMEventsFromDispatcher API to detach a notification HSM event from the dispatcher.

Parameters:

in	*hApp*	HANDLE - Application framework handle
in	*pServiceName*	PCSTR - Service Name
in	*puiCmdArray*	const PUI_32 - Array of commands

in	*uiCmdCount*	UI_32 - Number of Commands
in	*hSession*	HANDLE - Session Handle.

Returns:

status E_CWORD33_Status - success or error

See also:

E_CWORD33_Status _CWORD33_DetachParentHSMEventsFromDispatcher (HANDLE *hChildApp*, const PUI_32 *puiEventArray*, UI_32 *uiEventCount*)

Brief

detach notification HSM events from the dispatcher.

Parameters:

in	*hApp*	HANDLE hApp - Child Thread handle
in	*puiEventArray*	const PUI_32 puiEventArray - Array of events
in	*uiEventCount*	UI_32 uiEventCount - Number of events

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_DetachParentHSMEventsFromDispatcher API to detach a notification HSM event from the dispatcher.

Parameters:

in	*hChildApp*	HANDLE - Child Thread handle
in	*puiEventArray*	const PUI_32 puiEventArray - Array of events
in	*uiEventCount*	UI_32 - Number of events

Returns:

status E_CWORD33_Status - success or error

See also:

void _CWORD33_DisableAutoPublishServiceAvailable ()

Brief

Disable to the AutoPublishServiceAvailability Feature.

Return values:

None	

Prerequisite

Prerequisite is nothing.

Change of internal state

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

Classification

Forbidden

See also:

_CWORD33_HSMDispatcherWithArguments _CWORD33_PublishServiceAvailability

_CWORD33_DisableAutoPublishServiceAvailable Disables the AutoPublishServiceAvailability Feature that publishes the service availability on entry of s_CWORD33_Ready state and publishes the service unavailability on exit of s_CWORD33_Ready state. Applications are responsible for publishing the service availability.

void _CWORD33_EnableAutoPublishServiceAvailable ()

Brief

Enables the AutoPublishServiceAvailability Feature

Parameters:

none	

Return values:

E_CWORD33_Status	

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

none

See also:

C_CWORD33_HSMFramework* _CWORD33_GetStateMachine (HANDLE *hApp*)

Brief

Get pointer to the statemachine object.

Parameters:

in	*hApp*	HANDLE - Handle to the framework application

Return values:

*C_CWORD33_HSMParentFramework**	pointer to the statemachine object
NULL	Failed to get pointer to the statemachine object

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_HSMDispatcherWithArguments has been done.

Change of internal state

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

Classification

Public

Type

sync only

See also:

_CWORD33_HSMDispatcherWithArguments, _CWORD33_SetStateMachine

_CWORD33_GetStateMachine returns the pointer to the statemachine object

Parameters:

in	*hApp*	HANDLE - Handle to the application session

Returns:

pointer to the StateMachine object C_CWORD33_HSMParentFramework* - pointer to the StateMachine object

See also:

VOID _CWORD33_HSMDisableWaitInStoppingState ()

Brief

Chanage state to not wait in sStoppingState.

Return values:

None	

Prerequisite

Prerequisite is nothing.

Change of internal state

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

Classification

Public

Type

none

See also:

_CWORD33_HSMEnableWaitInStoppingState

E_CWORD33_Status _CWORD33_HSMDispatcher (PCSTR *cAppName*)

Brief

Creates, initializes and runs the HSM dispatcher.

Parameters:

in	*cAppName*	PCSTR - pointer of Application/ thread name

Return values:

E_CWORD33_Status	

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

none

See also:

E_CWORD33_Status _CWORD33_HSMDispatcherWithArguments (PCSTR *cAppName*, int *argc*, char * *argv*[], const _CWORD33_DefaultCallbackHandler * *CbHandler*, C_CWORD33_HSMFramework * *f_p_CWORD33_HSM* = NULL, CustomCommandLineOptions * *cmdLineOptions* = NULL)

Brief

This API creates, initializes the dispatcher and runs loop with command-line options.

Parameters:

in	*cAppName*	PCSTR - Pointer to application thread name
in	*argc*	int - Number of command-line options
in	*argv*	char *[] - Array of pointer to command-line options
in	*CbHandler*	const _CWORD33_DefaultCallbackHandler* - Pointer to default callback functions.
in	*f_p_CWORD33_HSM*	C_CWORD33_HSMFramework* - state machine object pointer(default NULL)
in	*cmdLineOptions*	CustomCommandLineOptions* - Parser setting of command-line options(Optional. When don't use, set NULL.)

_CWORD33_DefaultCallbackHandler Structure

```
1 typedef struct __CWORD33_DefaultCallbackHandler
2 {
3     CbFuncPtr onInitilization;     /* Function is called when a Dispatcher is created.     */
4     CbFuncPtr onDestroy;           /* Function is called when the Dispatcher is released.  */
5     CbFuncPtr onStart;             /* Function is called when the Dispatcher is started.   */
6     CbFuncPtr onStop;              /* Function is called when the Dispatcher is stopped.   */
7     CbFuncPtr onPreStart;          /* Function is called when the Dispatcher is pre started. */
8     CbFuncPtr onPreStop;           /* Function is called when the Dispatcher is pre stoped. */
9     CbFuncPtr onBackgroundStart;   /* Function is called when the Dispatcher is Background started. */
10    CbFuncPtr onBackgroundStop;    /* Function is called when the Dispatcher is Background stopped. */
11    CbFuncPtr onDebugDump;  /* Function is called when the Dispatcher detects abnormal state. */
12    CbFuncPtr createStateMachine;     /* Set dummy function that does nothing.              */
13    CbFuncPtr ssFrameworkInterface;   /* Function to connect to SystemManager               */
14 } _CWORD33_DefaultCallbackHandler;
```

CustomCommandLineOptions Structure

```
1 typedef struct _CustomCommandLineOptions
2 {
3   PCSTR          cShortOptions;        /* Short options list. */
4   PCHAR          cLongOptions;         /* Reserved. Set to NULL. */
5   CbArgumentParser callback;  /* Pointer to callback function to parse command-line options. */
6 } CustomCommandLineOptions;
```

About setting of default callback functions(_CWORD33_DefaultCallbackHandler)
Use _CWORD33__MAKE_DEFAULT_CALLBACK when initialize of
_CWORD33_DefaultCallbackHandler structure as argument CbHandler.

Application that run this API need to define functions below.(allow to dummy function that does nothing.)

E_CWORD33_Status _CWORD33_OnInitialization(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnPreStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnPreStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnBackgroundStart(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnBackgroundStop(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnDebugDump(HANDLE hApp)
E_CWORD33_Status _CWORD33_OnDestroy(HANDLE hApp)
E_CWORD33_Status _CWORD33_CreateStateMachine(HANDLE hApp)

If application is resident service, link library libSS_SystemIfUnified (This library provides the function _CWORD33_SSFrameworkInterface to connect to SystemManager.).

If application is nonresident service, define function that does nothing below.

E_CWORD33_Status _CWORD33_SSFrameworkInterface(HANDLE hApp)

Return values:

e_CWORD33_StatusNullPointer	NULL pointer specified
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusFail	Some sort of error occurred
e_CWORD33_StatusDuplicate	Duplication error of entry
e_CWORD33_StatusInvldHandle	Invalid handle
e_CWORD33_StatusErrOther	Other error has occurred(Cannot access shared memory, etc.)
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

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

No match

See also:

none

VOID _CWORD33_HSMEnableWaitInStoppingState ()

Brief

Change state application when call _CWORD33_OnStop().

Return values:

None	

Prerequisite

Prerequisite is nothing.

Change of internal state

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

Classification

Public

Type

none

See also:

_CWORD33_HSMDisableWaitInStoppingState

E_CWORD33_Status _CWORD33_HSMOnLoadData (HANDLE *hApp*)

All the persistent data of the applications should be loaded in this function.

Brief

This function defines loading persistent data of HSM application.

Parameters:

in	*hApp*	HANDLE - handle to application

Return values:

e_CWORD33_StatusOK	-successed
Except	e_CWORD33_StatusOK - failure

Prerequisite

None

Inside state change

None

Classification

Private

Type

None

See also:

None

E_CWORD33_Status _CWORD33_HSMOnStopIns_CWORD33_Run (HANDLE *hApp*)

Send stop request to all the child threads.

Brief

This function defines sending stop request to the child threads of HSM application.

Parameters:

in	*hApp*	HANDLE - handle to application

Return values:

e_CWORD33_StatusOK	-successed
Except	e_CWORD33_StatusOK - failure

Prerequisite

None

Inside state change

None

Classification

Private

Type

None

See also:

None

E_CWORD33_Status _CWORD33_NPHSMRegisterNotificationsEvents (HANDLE *hApp*, const _CWORD33_NotificationsList * *pList*, UI_32 *uiListLength*)

Brief

register many notifications in service

Parameters:

in	*hApp*	HANDLE hApp - Handle to the Framework
in	*pList*	const _CWORD33_NotificationsList *pList - List of the notifications that need to be registered
in	*uiListLength*	UI_32 uiListLength - Length of the list

Return values:

e_CWORD33_StatusOK	success

e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_NPHSMRegisterNotificationsEvents API to send message to Notification Service to register a notification

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pList*	_CWORD33_NotificationsList - List of the notifications that need to be registered
in	*uiListLength*	const UI_32 - Length of the list

Returns:

status E_CWORD33_Status - success or error

E_CWORD33_Status _CWORD33_NPHSMRegisterNotificatsionEvent (HANDLE *hApp*, PCSTR *pNotification*, const UI_32 *max_length*, const E_CWORD33_NotificationType *persType*)

Brief

register a notification in service

Parameters:

in	*hApp*	HANDLE hApp - Handle to the Framework
in	*pNotification*	PCSTR pNotification - Name of Notification
in	*max_length*	const UI_32 max_length - Max size of the notification message
in	*persType*	const E_CWORD33_NotificationType persType - Type of persistent

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_NPHSMRegisterNotificatsionEvent API to send message to Notification Service to register a notification

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pNotification*	PCSTR - Name of Notification
in	*max_length*	const UI_32 - Max size of the notification message
in	*persType*	const E_CWORD33_NotificationType - Type of persistent

Returns:

status E_CWORD33_Status - success or error

E_CWORD33_Status _CWORD33_NPHSMUnRegisterNotificationEvent (HANDLE *hApp*, PCSTR *pNotification*)

Brief

remove a notification in service

Parameters:

in	*hApp*	HANDLE hApp - Handle to the Framework
in	*pNotification*	cPCSTR pNotification - Name of Notification

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_NPHSMUnRegisterNotificationEvent API to send message to Notification Service to remove a notification

Parameters:

in	*hApp*	HANDLE - Handle to the Framework
in	*pNotification*	PCSTR - Name of Notification

Returns:

status E_CWORD33_Status - success or error

void _CWORD33_SetHSMType (EUserChangeOptions *f_eHSMType*)

Brief

Set the Application Statemachine Type

Parameters:

in	*cAppName*	PCSTR - pointer of Application/ thread name 1 typedef enum _EUserChangeOptions { 2 eUserchangeIgnore = ***, 3 eUserchangeReInit, 4 eUserchangeRetPrevState 5 } EUserChangeOptions;

Return values:

E_CWORD33_Status	

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

none

See also:

E_CWORD33_Status _CWORD33_SetStateMachine (HANDLE *hApp*, C_CWORD33_HSM * *f_p_CWORD33_HSM*)

Brief

Get pointer to the statemachine object.

Parameters:

in	*hApp*	HANDLE - Handle to the framework application

Return values:

*C_CWORD33_HSMParentFramework**	pointer to the statemachine object
NULL	Failed to get pointer to the statemachine object

Prerequisite

Generation/Initialization of Dispatcher for the Application by _CWORD33_HSMDispatcherWithArguments has been done.

Change of internal state

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

Classification

public

type

sync only

See also:

_CWORD33_HSMDispatcherWithArguments, _CWORD33_GetStateMachine

_CWORD33_SetStateMachine sets the statemachine object

Parameters:

in	*hApp*	HANDLE - Handle to the application session
in	*f_p_CWORD33_HSM*	C_CWORD33_HSM - Pointer to statemachine object

Returns:

pointer to the StateMachine object C_CWORD33_HSMParentFramework* - pointer to the StateMachine object

See also:

E_CWORD33_Status _CWORD33_SubscribeNotificationsWithHSMEvent (HANDLE *hApp*, const _CWORD33_NotificationEvent * *pNtfyEvent*, UI_32 *uiEventCount*)

Brief

attach many events to eath dispatcher on receiving a specific notification.

Parameters:

in	*hApp*	HANDLE hApp - Handle to the Framework
in	*pNtfyEvent*	const _CWORD33_NotificationEvent *pNtfyEvent - Notification event structure.
in	*uiEventCount*	UI_32 uiEventCount - Number of Notification event

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_AttachNotificationEventsToDispatcher API to attach a event to the dispatcher on receiving a specific notification.

Parameters:

in	*hApp*	HANDLE - Application framework handle
in	*pNtfyEvent*	const _CWORD33_NotificationEvent* - Notification event structure.
in	*uiEventCount*	UI_32 - Number of Notification event

Returns:

status E_CWORD33_Status - success or error

See also:

E_CWORD33_Status _CWORD33_SubscribeNotificationWithHSMEvent (HANDLE *hApp*, PCSTR *pNotification*, UI_32 *iEventId*)

Brief

attach an event to the dispatcher on receiving a specific notification.

Parameters:

in	*hApp*	HANDLE hApp - Handle to the Framework
in	*pNotification*	cPCSTR pNotification - Name of Notification
in	*iEventId*	UI_32 iEventId - event ID

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_AttachNotificationEventToDispatcher API to attach a event to the dispatcher on receiving a specific notification.

Parameters:

in	*hApp*	HANDLE - Application framework handle
in	*pNotification*	PCSTR - Notification String.
in	*iEventId*	UI_32 - event to be posted to active state on receiving iCmd from pServiceName.

Returns:

status E_CWORD33_Status - success or error

See also:

E_CWORD33_Status _CWORD33_SubscribeToSessionEventsWithHSMEvents (HANDLE *hApp*, const _CWORD33_ProtocolEvent * *pEventIds*, UI_32 *uiEventCount*, HANDLE *hSession*)

Brief

subscribing to multiple events of a service. attaches the session events with HSM events.

Parameters:

in	*hApp*	HANDLE hApp - Application handle
in	*pEventIds*	const _CWORD33_ProtocolEvent *pEventIds - structure mapping session event id's with the hsm event id's
in	*uiEventCount*	UI_32 uiEventCount - count of events
in	*hSession*	HANDLE hSession - Session Handle.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle
e_CWORD33_StatusFail	process failed

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_SubscribeToSessionEventsWithHSMEvents This API is used for subscribing to multiple events of a service. This API also attaches the session events with hsm events.

Parameters:

in	*hApp*	HANDLE - Application handle
in	*pEventIds*	_CWORD33_ProtocolEvent - structure mapping session event id's with the hsm event id's
in	*uiEventCount*	UI_32 - count of events
in	*hSession*	HANDLE - Session handle

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_SubscribeToSessionEventWithHSMEvent

E_CWORD33_Status _CWORD33_SubscribeToSessionEventWithHSMEvent (HANDLE *hApp*, UI_32 *uiEventId*, UI_32 *uiHSMEventId*, HANDLE *hSession*)

Brief

subscribing to single event of a service. attaches the session event with HSM events.

Parameters:

in	*hApp*	HANDLE hApp - Application handle
in	*uiEventId*	UI_32 uiEventId - Session Event id
in	*uiHSMEventId*	UI_32 uiHSMEventId- Statemachine Event id
in	*hSession*	HANDLE hSession - Session Handle.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle
e_CWORD33_StatusFail	process failed

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_SubscribeToSessionEventWithHSMEvent This API is used for subscribing to single event of a service. This API also attaches the session event with HSM events.

Parameters:

in	*hApp*	HANDLE - Application handle
in	*uiEventId*	UI_32 - Session Event id
in	*uiHSMEventId*	UI_32 - Statemachine Event id
in	*hSession*	HANDLE - Session handle

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_SubscribeToSessionEventsWithHSMEvents

E_CWORD33_Status _CWORD33_UnsubscribeNotificationsWithHSMEvent (HANDLE *hApp*, const _CWORD33_NotificationEvent * *pNtfyHandler*, UI_32 *uiHandlerCount*)

Brief

detach events from the dispatcher.

Parameters:

in	*hApp*	HANDLE hApp - Handle to the Framework
in	*pNtfyHandler*	const _CWORD33_NotificationEvent *pNtfyHandler - Notification handler structure.
in	*uiHandlerCount*	UI_32 uiHandlerCount - Notification handler structure.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_UnsubscribeNotificationsWithHSMEvent API to detach a notification event from the dispatcher.

Parameters:

in	*hApp*	HANDLE - Application framework handle
in	*pNtfyHandler*	const _CWORD33_NotificationEvent* - Notification handler structure.
in	*uiHandlerCount*	UI_32 - Number of notification handlers

Returns:

status E_CWORD33_Status - success or error

See also:

E_CWORD33_Status _CWORD33_UnsubscribeNotificationWithHSMEvent (HANDLE *hApp*, PCSTR *pNotification*)

Brief

detach a notification event from the dispatcher.

Parameters:

in	*hApp*	HANDLE hApp - Handle to the Framework
in	*pNotification*	PCSTR pNotification - Notification String.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_DetachNotificationEventsFromDispatcher API to detach a notification event from the dispatcher.

Parameters:

in	*hApp*	HANDLE - Application framework handle

in	*pNotification*	PCSTR - Notification String.

Returns:

status E_CWORD33_Status - success or error

See also:

E_CWORD33_Status _CWORD33_UnSubscribeSessionEventsWithHSMEvents (HANDLE *hApp*, PUI_32 *pEventsArray*, UI_32 *uiListSize*, HANDLE *hSession*)

Brief

unsubscribe from multiple events of a service. detaches HSM events from session events.

Parameters:

in	*hApp*	HANDLE hApp - Application handle
in	*pEventsArray*	PUI_32 pEventsArray - Array of events id's.
in	*uiListSize*	UI_32 uiListSize - count of elements in array.
in	*hSession*	HANDLE hSession - Session Handle.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle
e_CWORD33_StatusFail	process failed

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_UnSubscribeSessionEventsWithHSMEvents API to unsubscribe from multiple events of a service. Also detaches HSM events from session events.

Parameters:

in	*hApp*	HANDLE - Application Handle
in	*pEventsArray*	PUI_32 - Array of events id's.
in	*uiListSize*	UI_32 - count of elements in array.
in	*hSession*	HANDLE - Session Handle

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_UnSubscribeSessionEventWithHSMEvent

E_CWORD33_Status _CWORD33_UnSubscribeSessionEventWithHSMEvent (HANDLE *hApp*, UI_32 *uiEventId*, HANDLE *hSession*)

Brief

unsubscribe from event of a service. detaches HSM event from session event.

Parameters:

in	*hApp*	HANDLE hApp - Application handle
in	*uiEventId*	UI_32 uiEventId - Event id
in	*hSession*	HANDLE hSession - Session Handle.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	invalid state handle
e_CWORD33_StatusFail	process failed

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_UnSubscribeSessionEventWithHSMEvent API to unsubscribe from event of a service. Also detaches HSM event from session event.

Parameters:

in	*hApp*	HANDLE - Application Handle
in	*uiEventId*	UI_32 - Event id
in	*hSession*	HANDLE - Session Handle

Returns:

status E_CWORD33_Status - success or error

See also:

_CWORD33_UnSubscribeSessionEventsWithHSMEvents

_CWORD33_log_library

#define **_CWORD33_LOG_SHARED_MEM_NAME** "_CWORD33_logconfig.cfg"
#define
_CWORD33_LOG_RAMDISC_NAME "/ramd/log/_CWORD33_log/_CWORD33__debug.log"
#define **_CWORD33_LOG_RAMDISC_PATH** "/ramd/log/_CWORD33_log"
#define **_CWORD33_LOG_FLAG_MODE_DEBUG** 1
#define **_CWORD33_LOG_FLAG_MODE_RELEASE** 0
#define **_CWORD33_LOG_REALTIMELOG_DISABLE_MASK** 0x80
#define **_CWORD33_LOG_REALTIMELOG_MODE_FREEZE** 0xFF
#define **_CWORD33_LOG_REALTIMELOG_MODE_OFF** 0
#define **_CWORD33_LOG_REALTIMELOG_MODE_UART** 1
#define **_CWORD33_LOG_REALTIMELOG_MODE_USB** 2
#define
_CWORD33_LOG_REALTIMELOG_MODE_USB_DISABLE (_CWORD33_LOG_REALTIMELOG_MODE_USB | _CWORD33_LOG_REALTIMELOG_DISABLE_MASK)
#define **_CWORD33_LOG_REALTIMELOG_MODE_ETHER** 3
#define **_CWORD33_LOG_REALTIMELOG_MODE_MAX** 4

Functions

int _CWORD33_LOG_Freeze (void)

Detailed Description

Function Documentation

int _CWORD33_LOG_Freeze (void)

Brief

Freeze _CWORD33_log API.

Return values:

0	success
-1	error

Prerequisite

none

Classification

Public

Type

Sync Only

See also:

none

Backup_manager

#define NTFY_BackupMgr_Availability "NS_BackupMgr/Availability"
#define BKUP_RET_NORMAL 0
#define BKUP_RET_ERROR -1
#define BKUP_RET_ERRPARAM -2
#define BKUP_RET_ERRINIT -3
#define BKUP_RET_ERRTERM -4
#define BKUP_RET_ERRNOENT -5
#define BKUP_RET_ERRSIZE -6

Functions

int32_t Backup_DataRd (PCSTR tag_id, uint32_t ui_offset, void *pv_buf, uint32_t ui_size)
int32_t Backup_DataWt (PCSTR tag_id, void *pv_buf, uint32_t ui_offset, uint32_t ui_size)
int32_t Backup_DataFil (PCSTR tag_id, uint32_t ui_offset, uint8_t uc_pat, uint32_t ui_size)
int32_t Backup_DataSz (PCSTR tag_id, uint32_t *pui_size)
int32_t Backup_DataRdByNumID (uint32_t num_id, uint32_t ui_offset, void *pv_buf, uint32_t ui_size)
int32_t Backup_DataSzByNumID (uint32_t num_id, uint32_t *pui_size)
int32_t Backup_DataChk (PCSTR tag_id)
int32_t Backup_DataDel (PCSTR tag_id)

Detailed Description

Macro Definition Documentation

#define BKUP_RET_ERRINIT -3

Return value: initializing

#define BKUP_RET_ERRNOENT -5

Return value: data does not exist

#define BKUP_RET_ERROR -1

Return value: abnormal termination(content not specified)

#define BKUP_RET_ERRPARAM -2

Return value: parameter error

#define BKUP_RET_ERRSIZE -6

Return value: data size error

#define BKUP_RET_ERRTERM -4

Return value: terminating

#define BKUP_RET_NORMAL 0

Return value: terminated normally

#define NTFY_BackupMgr_Availability "NS_BackupMgr/Availability"

Service availability notify

Function Documentation

int32_t Backup_DataChk (PCSTR *tag_id*)

Brief

Check backup data.

Parameters:

in	*tagID*	PCSTR - Area ID(character string of less than 64 bytes)

Return values:

BKUP_RET_NORMAL	Terminated normally
BKUP_RET_ERRPARAM	Parameter error(Sync message transmission or

	reception(McInvokeSync) error[e_CWORD33_StatusInvldParam])
BKUP_RET_ERRINIT	Initializing(Sync message send or receive(McInvokeSync) error [e_CWORD33_StatusErrOther])
BKUP_RET_ERRTERM	Terminating(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusExit])
BKUP_RET_ERRNOENT	Data dose not exist(Sync message transmission or reception(McInvokeSync) fails [e_CWORD33_StatusFileLoadError])
BKUP_RET_ERRSIZE	Data size error(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusAccessError])
BKUP_RET_ERROR	Abnormal termination(McOpenSender failed, or sync message transmission or reception(McInvokeSync) error other than the above)

Prerequisite

None.

Change of internal state

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

Classification

Public

Type

Sync

See also:

none

int32_t Backup_DataDel (PCSTR *tag_id*)

Brief

Delete backup data.

Parameters:

in	*tag_id*	PCSTR - Area ID(character string of less than 64 bytes)

Return values:

BKUP_RET_NORMAL	Terminated normally
BKUP_RET_ERRPARAM	Parameter error(Sync message transmission or reception(McInvokeSync) error[e_CWORD33_StatusInvldParam])
BKUP_RET_ERRINIT	Initializing(Sync message send or receive(McInvokeSync) error [e_CWORD33_StatusErrOther])
BKUP_RET_ERRTERM	Terminating(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusExit])
BKUP_RET_ERRNOENT	Data dose not exist(Sync message transmission or reception(McInvokeSync) fails [e_CWORD33_StatusFileLoadError])
BKUP_RET_ERRSIZE	Data size error(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusAccessError])

BKUP_RET_ERROR	Abnormal termination(McOpenSender failed, or sync message transmission or reception(McInvokeSync) error other than the above)

Prerequisite

None.

Change of internal state

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

Classification

Public

Type

Sync

See also:

none

int32_t Backup_DataFil (PCSTR *tag_id*, uint32_t *ui_offset*, uint8_t *uc_pat*, uint32_t *ui_size*)

Brief

Write data to backup area with the specified pattern

Parameters:

in	*tagID*	PCSTR - Area ID(character string of less than 64 bytes)
in	*uiOffset*	uint32_t - Offset from the beginning of the area(0~size of backup area identified by area ID)
in	*ucPat*	uint8_t - Write pattern data(0x00~0xff)
in	*uiSize*	uint32_t - Write data size(0~size of backup area identified by area ID)

Return values:

BKUP_RET_NORMAL	Terminated normally
BKUP_RET_ERRPARAM	Parameter error(Sync message transmission or reception(McInvokeSync) error[e_CWORD33_StatusInvldParam])
BKUP_RET_ERRINIT	Initializing(Sync message send or receive(McInvokeSync) error [e_CWORD33_StatusErrOther])
BKUP_RET_ERRTERM	Terminating(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusExit])
BKUP_RET_ERRNOENT	Data dose not exist(Sync message transmission or reception(McInvokeSync) fails [e_CWORD33_StatusFileLoadError])
BKUP_RET_ERRSIZE	Data size error(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusAccessError])
BKUP_RET_ERROR	Abnormal termination(McOpenSender failed, or sync message transmission or reception(McInvokeSync) error other than the above)

Prerequisite

None.

Change of internal state

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

Classification

Public

Type

Sync

See also:

none

int32_t Backup_DataRd (PCSTR *tag_id*, uint32_t *ui_offset*, void * *pv_buf*, uint32_t *ui_size*)

Brief

Read data from backup area

Parameters:

in	*tagID*	PCSTR - Area ID(character string of less than 64 bytes)
in	*uiOffset*	uint32_t - Offset from the beginning of the area(0~size of backup area identified by area ID)
out	*pvBuf*	void * - A pointer which point to buffer used to storage read data
in	*uiSize*	uint32_t - Read data size(0~size of backup area identified by area ID)

Return values:

BKUP_RET_NORMAL	Terminated normally
BKUP_RET_ERRPARAM	Parameter error(Sync message transmission or reception(McInvokeSync) error[e_CWORD33_StatusInvldParam])
BKUP_RET_ERRINIT	Initializing(Sync message send or receive(McInvokeSync) error [e_CWORD33_StatusErrOther])
BKUP_RET_ERRTERM	Terminating(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusExit])
BKUP_RET_ERRNOENT	Data dose not exist(Sync message transmission or reception(McInvokeSync) fails [e_CWORD33_StatusFileLoadError])
BKUP_RET_ERRSIZE	Data size error(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusAccessError])
BKUP_RET_ERROR	Abnormal termination(McOpenSender failed, or sync message transmission or reception(McInvokeSync) error other than the above)

Prerequisite

None.

Change of internal state

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

Classification

Public

Type

Sync

See also:

Backup_DataWt

int32_t Backup_DataRdByNumID (uint32_t *num_id*, uint32_t *ui_offset*, void * *pv_buf*, uint32_t *ui_size*)

Brief

Read data from backup area

Parameters:

in	*numID*	uint32_t - Area ID specified by number
in	*uiOffset*	uint32_t - Offset from the beginning of the area(0~size of backup area identified by area ID)
out	*pvBuf*	void * - A pointer which point to buffer used to storage read data
in	*uiSize*	uint32_t - Read data size(0~size of backup area identified by area ID)

Return values:

BKUP_RET_NORMAL	Terminated normally
BKUP_RET_ERRPARAM	Parameter error(Sync message transmission or reception(McInvokeSync) error[e_CWORD33_StatusInvldParam])
BKUP_RET_ERRINIT	Initializing(Sync message send or receive(McInvokeSync) error [e_CWORD33_StatusErrOther])
BKUP_RET_ERRTERM	Terminating(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusExit])
BKUP_RET_ERRNOENT	Data dose not exist(Sync message transmission or reception(McInvokeSync) fails [e_CWORD33_StatusFileLoadError])
BKUP_RET_ERRSIZE	Data size error(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusAccessError])
BKUP_RET_ERROR	Abnormal termination(McOpenSender failed, or sync message transmission or reception(McInvokeSync) error other than the above)

Prerequisite

None.

Change of internal state

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

Classification

Public

Type

Sync

See also:

Backup_DataRd

int32_t Backup_DataSz (PCSTR *tag_id*, uint32_t * *pui_size*)

Brief

Get the size of the backup area.

Parameters:

in	*tagID*	PCSTR - Area ID(character string of less than 64 bytes)
out	*puiSize*	uint32_t * - A pointer which point to storage size

Return values:

BKUP_RET_NORMAL	Terminated normally
BKUP_RET_ERRPARAM	Parameter error(Sync message transmission or reception(McInvokeSync) error[e_CWORD33_StatusInvldParam])
BKUP_RET_ERRINIT	Initializing(Sync message send or receive(McInvokeSync) error [e_CWORD33_StatusErrOther])
BKUP_RET_ERRTERM	Terminating(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusExit])
BKUP_RET_ERRNOENT	Data dose not exist(Sync message transmission or reception(McInvokeSync) fails [e_CWORD33_StatusFileLoadError])
BKUP_RET_ERRSIZE	Data size error(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusAccessError])
BKUP_RET_ERROR	Abnormal termination(McOpenSender failed, or sync message transmission or reception(McInvokeSync) error other than the above)

Prerequisite

None.

Change of internal state

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

Classification

Public

Type

Sync

See also:

none

int32_t Backup_DataSzByNumID (uint32_t *num_id*, uint32_t * *pui_size*)

Brief

Get the size of the backup area.

Parameters:

in	*numID*	uint32_t - Area ID specified by number
out	*puiSize*	uint32_t * - A pointer which point to storage size

Return values:

BKUP_RET_NORMAL	Terminated normally
BKUP_RET_ERRPARAM	Parameter error(Sync message transmission or reception(McInvokeSync) error[e_CWORD33_StatusInvldParam])
BKUP_RET_ERRINIT	Initializing(Sync message send or receive(McInvokeSync) error [e_CWORD33_StatusErrOther])
BKUP_RET_ERRTERM	Terminating(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusExit])
BKUP_RET_ERRNOENT	Data dose not exist(Sync message transmission or reception(McInvokeSync) fails [e_CWORD33_StatusFileLoadError])
BKUP_RET_ERRSIZE	Data size error(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusAccessError])
BKUP_RET_ERROR	Abnormal termination(McOpenSender failed, or sync message transmission or reception(McInvokeSync) error other than the above)

Prerequisite

None.

Change of internal state

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

Classification

Public

Type

Sync

See also:

Backup_DataSz

int32_t Backup_DataWt (PCSTR *tag_id*, void * *pv_buf*, uint32_t *ui_offset*, uint32_t *ui_size*)

Brief

Write data to backup area

Parameters:

in	*tagID*	PCSTR - Area ID(character string of less than 64 bytes)

in	*pvBuf*	void * - A pointer which point to buffer used to storage write data
in	*uiOffset*	uint32_t - Offset from the beginning of the area(0~size of backup area identified by area ID)
in	*uiSize*	uint32_t - Write data size(0~size of backup area identified by area ID)

Return values:

BKUP_RET_NORMAL	Terminated normally
BKUP_RET_ERRPARAM	Parameter error(Sync message transmission or reception(McInvokeSync) error[e_CWORD33_StatusInvldParam])
BKUP_RET_ERRINIT	Initializing(Sync message send or receive(McInvokeSync) error [e_CWORD33_StatusErrOther])
BKUP_RET_ERRTERM	Terminating(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusExit])
BKUP_RET_ERRNOENT	Data dose not exist(Sync message transmission or reception(McInvokeSync) fails [e_CWORD33_StatusFileLoadError])
BKUP_RET_ERRSIZE	Data size error(Sync message transmission or reception(McInvokeSync) error [e_CWORD33_StatusAccessError])
BKUP_RET_ERROR	Abnormal termination(McOpenSender failed, or sync message transmission or reception(McInvokeSync) error other than the above)

Prerequisite

None.

Change of internal state

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

Classification

Public

Type

Sync

See also:

Backup_DataRd

Common_library

#define CL_RegionAlloc(region, type, length) cl_region_alloc(region, sizeof(type) * length, CL_ALIGNOF(type))

#define CL_RegionCleanupAdd(region, type, length) cl_region_cleanup_add(region, sizeof(type) * length, CL_ALIGNOF(type))

Enumerations

enum **ENUM_LOCK_ID** { **LOCK_ANA_LOG_SEM** = ***, **LOCK_POS_MTX_1**, **LOCK_POS_MTX_2**, **LOCK_POS_MTX_3**, **LOCK_POS_MTX_4**, **LOCK_POS_MTX_5**, **LOCK_POS_MTX_6**,

LOCK_POS_MTX_7, **LOCK_POS_MTX_8**, **LOCK_POS_MTX_9**, **LOCK_POS_MTX_10**, **LOCK_POS_MTX_11**, **LOCK_POS_MTX_12**, **LOCK_POS_MTX_13**, **LOCK_POS_MTX_14**, **LOCK_POS_MTX_15**, **LOCK_POS_MTX_16**, **LOCK_POS_MTX_17**, **LOCK_POS_MTX_18**, **LOCK_POS_MTX_19**, **LOCK_POS_MTX_20**, **LOCK_POS_MTX_21**, **LOCK_POS_MTX_22**, **LOCK_POS_MTX_23**, **LOCK_POS_MTX_24**, **LOCK_POS_MTX_25**, **LOCK_POS_MTX_26**, **LOCK_POS_MTX_27**, **LOCK_POS_MTX_28**, **LOCK_POS_MTX_29**, **LOCK_POS_MTX_30**, **LOCK_POS_MTX_31**, **LOCK_POS_MTX_32**, **LOCK_CLOCK_SEM_TZ**, **LOCK_SEC_DATA_PROTECT**, **LOCK_OS_SEM_ID__CWORD68_**, **LOCK_ROM_ACCESS_IF**, **LOCK_BOOT_ACCESS_IF**, **LOCK_RAM_ACCESS_IF**, **LOCK_HRDS_1**, **LOCK_HRDS_2**, **LOCK_HRDS_3**, **LOCK_HRDS_4**, **LOCK_HRDS_5**, **LOCK_HRDS_6**, **LOCK_HRDS_7**, **LOCK_HRDS_8**, **LOCK_LOGG_ACCESS_IF**, **LOCK__CWORD76__HMI**, **LOCK__CWORD76___CWORD58_**, **LOCK_DIAGCODE_MEM**, **LOCK_MOUNTER_MOUNT**, **LOCK_NOR_ERCNT**, **LOCK_RS_WNG_HIS_LIST_DATA**, **LOCK_RS_WNG_SCREEN_DATA**, **LOCK_RS_SCREEN_DATA**, **LOCK_NSLOG_ACCES_IF_1**, **LOCK_NSLOG_ACCES_IF_2**, **LOCK_NSLOG_ACCES_IF_3**, **LOCK_NSLOG_ACCES_IF_4**, **LOCK_NSLOG_ACCES_IF_5**, **LOCK_NSLOG_ACCES_IF_6**, **LOCK_NSLOG_ACCES_IF_7**, **LOCK_NSLOG_ACCES_IF_8**, **LOCK_NSLOG_ACCES_IF_9**, **LOCK_NSLOG_ACCES_IF_10**, **LOCK_NSLOG_ACCES_IF_11**, **LOCK_NSLOG_ACCES_IF_12**, **LOCK_NSLOG_ACCES_IF_13**, **LOCK_NSLOG_ACCES_IF_14**, **LOCK_NSLOG_ACCES_IF_15**, **LOCK_NSLOG_ACCES_IF_16**, **LOCK_NSLOG_ACCES_IF_17**, **LOCK_NSLOG_ACCES_IF_18**, **LOCK_NSLOG_ACCES_IF_19**, **LOCK_NSLOG_ACCES_IF_20**, **LOCK_NSLOG_ACCES_IF_21**, **LOCK_NSLOG_ACCES_IF_22**, **LOCK_NSLOG_ACCES_IF_23**, **LOCK_NSLOG_ACCES_IF_24**, **LOCK_NSLOG_ACCES_IF_25**, **LOCK_NSLOG_ACCES_IF_26**, **LOCK_NSLOG_ACCES_IF_27**, **LOCK_NSLOG_ACCES_IF_28**, **LOCK_NSLOG_ACCES_IF_29**, **LOCK_NSLOG_ACCES_IF_30**, **LOCK_NSLOG_ACCES_IF_31**, **LOCK_NSLOG_ACCES_IF_32**, **LOCK_NSLOG_ACCES_IF_33**, **LOCK_NSLOG_ACCES_IF_34**, **LOCK_NSLOG_ACCES_IF_35**, **LOCK_NSLOG_ACCES_IF_36**, **LOCK_NSLOG_ACCES_IF_37**, **LOCK_NSLOG_ACCES_IF_38**, **LOCK_NSLOG_ACCES_IF_39**, **LOCK_NSLOG_ACCES_IF_40**, **LOCK_NSLOG_ACCES_IF_41**, **LOCK_NSLOG_ACCES_IF_42**, **LOCK_NSLOG_ACCES_IF_43**, **LOCK_INFOSETTING_REV**, **LOCK_SUBMENU_SELECT**, **LOCK_DIAG_SEM1**, **LOCK_DIAG_SEM2**, **LID_NUM** }

Functions

int32_t CL_LockSystemInit (void)
int32_t CL_LockProcessInit (void)
void * CL_LockMap (int32_t lid)
int32_t CL_LockUnmap (void *addr)
int32_t CL_LockGet (void *addr)
int32_t CL_LockNowait (void *addr)
int CL_LockRelease (void *addr)
int CL_MonitorInit (CL_MonitorInit_t init_type)
int CL_MonitorSetEntry (CL_MonitorType_t type, uint32_t id, CL_MonitorState_t state, uint32_t timeout, uint32_t user_data)
int CL_MonitorGetEntry (CL_MonitorType_t type, uint32_t id, CL_MonitorEntry_t *entry)
int CL_MonitorSearchInit (CL_MonitorSearch_t *serch)
int CL_MonitorSearchDestroy (CL_MonitorSearch_t *serch)
int CL_MonitorSearchTimeout (CL_MonitorSearch_t *search)
int CL_ProcessInit (void)
int CL_ProcessCreate (const char *file, char *const argv[], char *const envp[], const CL_ProcessAttr_t *attr)

int CL_ProcessCreateAttrInit (CL_ProcessAttr_t *attr)
int CL_ProcessCreateAttrSetName (CL_ProcessAttr_t *attr, const char *name)
int CL_ProcessCreateAttrSetUid (CL_ProcessAttr_t *attr, uid_t uid)
int CL_ProcessCreateAttrSetGid (CL_ProcessAttr_t *attr, gid_t gid)
int CL_ProcessCreateAttrSetSchedule (CL_ProcessAttr_t *attr, CL_ProcessSchedPolicy_t policy, int priority)
int CL_ProcessCreateAttrSetGroup (CL_ProcessAttr_t *attr, int create)
int **CL_ProcessCreateAttrSetCpuAssign** (CL_ProcessAttr_t *attr, int cpu_assign)
int CL_ProcessCreateAttrSetStackSize (CL_ProcessAttr_t *attr, int stack_size)
int CL_ProcessCreateAttrSetHoldFds (CL_ProcessAttr_t *attr, int hold_fds[])
int CL_ProcessCreateAttrSetDisableCloseFds (CL_ProcessAttr_t *attr)
int CL_ProcessCreateAttrSetCgroup (CL_ProcessAttr_t *attr, const char *cgroup_name)
int CL_ProcessTerminate (pid_t pid)
int CL_ProcessTerminateGroup (pid_t pid)
int **CL_ProcessAbort** (pid_t pid)
int **CL_ProcessAbortGroup** (pid_t pid)
int **CL_ProcessEuthanizeGroup** (pid_t pid)
int CL_ProcessCleanup (int sigchld_fd, CL_ProcessCleanupInfo_t *cleanup_info)
int **CL_ThreadCreate** (pthread_t *thread, pthread_attr_t *attr, CL_ThreadAttr_t *cl_attr, void *(*start_routine)(void *), void *arg)
int **CL_ThreadCreateAttrInit** (CL_ThreadAttr_t *attr)
int **CL_ThreadCreateAttrSetName** (CL_ThreadAttr_t *attr, const char *name)
int CL_ProcessCreateCgroupCreate (const char *cgroup_name, CL_ProcessCreateCgroupAttr_t *attr)
int CL_ProcessCreateCgroupAttrInit (CL_ProcessCreateCgroupAttr_t *attr)
int CL_ProcessCreateCgroupAttrSetRtThrottling (CL_ProcessCreateCgroupAttr_t *attr, int runtime_us)
int CL_ProcessCreateCgroupAttrSetCpuShares (CL_ProcessCreateCgroupAttr_t *attr, int cpu_shares)
int CL_ProcessCreateCgroupAttrSetCfsBandwidthControl (CL_ProcessCreateCgroupAttr_t *attr, int cfs_quota_us)
int CL_ProcessCreateCgroupAttrSetMemoryLimit (CL_ProcessCreateCgroupAttr_t *attr, int memory_limit)
int CL_ProcessCreateCgroupAttrSetMemoryUsageNotification (CL_ProcessCreateCgroupAttr_t *attr, int usage_in_bytes, int event_fd)
int CL_ProcessCreateCgroupDelete (const char *cgroup_name)
int CL_ProcessCreateCgroupClassify (const char *cgroup_name, pid_t pid)
cl_region_t * CL_RegionCreate (size_t size)
void CL_RegionDestroy (cl_region_t *region)
void * **cl_region_alloc** (cl_region_t *region, size_t size, size_t align_size)
bool CL_RegionFree (cl_region_t *region, void *p)
cl_region_cleanup_t * **cl_region_cleanup_add** (cl_region_t *region, size_t size, size_t align_size)
EXT_C int CL_SemWait (sem_t *semid, unsigned int timeout)
void CL_TlsInit (int pno)
CL_TlsData_t * CL_TlsGet (void)
int CL_TlsGetRcvFD (void)
void CL_TlsSetRcvFD (int fd)
int CL_TlsThreadSeqID (void)

Detailed Description

Macro Definition Documentation

#define CL_RegionAlloc(region, type, length) cl_region_alloc(region, sizeof(type) * length, CL_ALIGNOF(type))

Brief

Region allocation

Parameters:

in	*region*	cl_region_t * - the region to allocate
in	*size*	size_t * - the size to allocate
in	*align_size*	size_t * - the size to align

Return values:

the	pointer to region allocated

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

#define CL_RegionCleanupAdd(region, type, length) cl_region_cleanup_add(region, sizeof(type) * length, CL_ALIGNOF(type))

Brief

add region cleanup

Parameters:

in	*region*	cl_region_t * - the region to allocate
in	*size*	size_t * - the size to add
in	*align_size*	size_t * - the size to align

Return values:

the	pointer to region cleanup added

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

Function Documentation

int32_t CL_LockGet (void * *addr*)

Brief

Get the Lock. Block until get the Lock.

Parameters:

in	*addr*	void* - Lock information address.

Return values:

0	Success
EINVAL	Invalid parameter
EDEADLK	Mutex already locked

Prerequisite

Should call CL_LockMap to get the Lock information address.

Change of internal state

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

Classification

Public

Type

Sync only

See also:

CL_LockRelease

void* CL_LockMap (int32_t *lid*)

Brief

Mapping the Lock information address.

Parameters:

in	*lid*	int32_t - LockID of the Lock used.(0~LID_NUM)

Return values:

addr	Lock address
MAP_FAILED	Error (errno)

Prerequisite

Should call CL_LockSystemInit(), CL_LockProcessInit() to open the Lock file.

Change of internal state

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

Classification

Public

Type

Sync only

See also:

CL_LockUnmap

int32_t CL_LockNowait (void * *addr*)

Brief

Try to get Lock, shall immediately return.

Parameters:

in	*addr*	void* - Lock information address.

Return values:

0	Success
EINVAL	Invalid parameter
EBUSY	Busy

Prerequisite

Should call CL_LockMap to get the Lock information address.

Change of internal state

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

Classification

Public

Type

Sync only

See also:

CL_LockRelease

int32_t CL_LockProcessInit (void)

Brief

Initialize the process Lock file.

Return values:

0	Success
-1	Error

Prerequisite

Should call CL_LockSystemInit() to generate Lock file.

Change of internal state

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

Classification

Public

Type

Sync only

See also:

none

int CL_LockRelease (void * *addr*)

Brief

Release the Lock.

Parameters:

in	*addr*	addr - Lock information address.

Return values:

0	Success
EINVAL	Invalid parameter
EPERM	The current thread does not own the mutex

Prerequisite

Should call CL_LockMap to get the Lock information address.

Change of internal state

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

Classification

Public

Type

Sync only

See also:

CL_LockGet, CL_LockNowait

int32_t CL_LockSystemInit (void)

Brief

Initialize the system lock.

Return values:

0	Success
-1	Error

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

Sync only

See also:

none

int32_t CL_LockUnmap (void * *addr*)

Brief

Unmapping the Lock information.

Parameters:

in	*addr*	void* - Lock information address.

Return values:

0	Success
-1	Error

Prerequisite

Should call CL_LockSystemInit(), CL_LockProcessInit() before this function is called

Change of internal state

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

Classification

Public

Type

Sync only

See also:

CL_LockMap

int CL_MonitorGetEntry (CL_MonitorType_t *type*, uint32_t *id*, CL_MonitorEntry_t * *entry*)

Brief

Get Monitor Entry

Parameters:

in	*type*	CL_MonitorType_t - the type of monitor
in	*id*	uint32_t - the monitor id
out	*entry*	CL_MonitorEntry_t* - the pointer to monitor entry

Return values:

0	Success
-1	Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

int CL_MonitorInit (CL_MonitorInit_t *init_type*)

Brief

Monitor initialize

Parameters:

in	*init_type*	CL_MonitorInit_t - the type of initialzation

Return values:

0	Success
-1	Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

int CL_MonitorSearchDestroy (CL_MonitorSearch_t * *serch*)

Brief

Monitor searth destroy

Parameters:

in	*serch*	CL_MonitorSearch_t* - the pointor to monitory search

Return values:

0	Success
-1	Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

int CL_MonitorSearchInit (CL_MonitorSearch_t * *serch*)

Brief

Monitor searth intialize

Parameters:

in	*serch*	CL_MonitorSearch_t* - the pointor to monitory search

Return values:

0	Success

-1	Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

int CL_MonitorSearchTimeout (CL_MonitorSearch_t * *search*)

Brief

Monitor searth timeout

Parameters:

in	*serch*	CL_MonitorSearch_t* - the pointor to monitory search

Return values:

0	Success
-1	Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

int CL_MonitorSetEntry (CL_MonitorType_t *type*, uint32_t *id*, CL_MonitorState_t *state*, uint32_t *timeout*, uint32_t *user_data*)

Brief

Set Monitor Entry

Parameters:

in	*type*	CL_MonitorType_t - the type of monitor

in	*state*	CL_MonitorState_t - the state of monitor
in	*timeout*	uint32_t - the vlaue of timeout
in	*user_data*	uint32_t - the user data

Return values:

0	Success
-1	Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

int CL_ProcessCleanup (int *sigchld_fd*, CL_ProcessCleanupInfo_t * *cleanup_info*)

Brief

Parameters:

in	*sigchld_fd*	int - CL_ProcessInit() function return to fd
out	*cleanup_info*	CL_ProcessCleanupInfo_t* - Pointer to chiled process info

CL_ProcessCleanupInfo_t struct

```
1 typedef struct {
2   pid_t pid;          /* The process ID of the child. */
3   int code;           /* signal code */
4   int status;         /* the exit status of the child or the signal */
5 } CL_ProcessCleanupInfo_t;
```

Return values:

0	Success
1	Success (If WNOHANG was specified and no child(ren) specified by id has yet changed state)
-1	Error (setting to errno)

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreate (const char * *file*, char *const *argv*[], char *const *envp*[], const CL_ProcessAttr_t * *attr*)

Brief

Parameters:

in	*file*	const char* -
in	*argv[]*	char* const -
in	*envp[]*	char* const
in	*attr*	const CL_ProcessAttr_t* -

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

Return values:

int	Success(PID)
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

Open Close

See also:

CL_ProcessTerminate, CL_ProcessTerminateGroup CL_ProcessCreateAttrInit, CL_ProcessCreateAttrSetCgroup, CL_ProcessCreateAttrSetDisableCloseFds, CL_ProcessCreateAttrSetGid, CL_ProcessCreateAttrSetGroup, CL_ProcessCreateAttrSetHoldFds, CL_ProcessCreateAttrSetName,

CL_ProcessCreateAttrSetSchedule, CL_ProcessCreateAttrSetStackSize, CL_ProcessCreateAttrSetUid

int CL_ProcessCreateAttrInit (CL_ProcessAttr_t * *attr*)

Brief

Parameters:

out	*attr*	const CL_ProcessAttr_t* -

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

Method(Async) / Fire and Forget / Broadcast / Sync / Set Get / Open Close / Request Notify / No match

See also:

none

int CL_ProcessCreateAttrSetCgroup (CL_ProcessAttr_t * *attr*, const char * *cgroup_name*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessAttr_t* -
in	*cgroup_name*	const char* - Cgroup name

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateAttrSetDisableCloseFds (CL_ProcessAttr_t * *attr*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessAttr_t* -

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateAttrSetGid (CL_ProcessAttr_t * *attr*, gid_t *gid*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessAttr_t* -
in	*gid*	gid_t - Group ID

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateAttrSetGroup (CL_ProcessAttr_t * *attr*, int *create*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessAttr_t* -

in	*create*	int - (0 or 1)

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateAttrSetHoldFds (CL_ProcessAttr_t * *attr*, int *hold_fds*[])

Brief

Parameters:

in,out	*attr*	const CL_ProcessAttr_t* -
in	*hold_fds[]*	int -

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateAttrSetName (CL_ProcessAttr_t * *attr*, const char * *name*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessAttr_t* -
in	*name*	const char* - process name (max length 16byte)

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateAttrSetSchedule (CL_ProcessAttr_t * *attr*, CL_ProcessSchedPolicy_t *policy*, int *priority*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessAttr_t* -
in	*policy*	CL_ProcessSchedPolicy_t - schedule policy
in	*priority*	int - priority

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

enum CL_ProcessSchedPolicy_t Variable

```
1 typedef enum {
2   CL_PROCESS_SCHED_POLICY_OTHER = ***,  // TSS
3   CL_PROCESS_SCHED_POLICY_RR,         // Round-robin
4   CL_PROCESS_SCHED_POLICY_FIFO,       // FIFO
5 } CL_ProcessSchedPolicy_t;
```

target to priority
-2019 : CL_PROCESS_SCHED_POLICY_OTHER
199 : CL_PROCESS_SCHED_POLICY_RR or CL_PROCESS_SCHED_POLICY_FIFO

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateAttrSetStackSize (CL_ProcessAttr_t * *attr*, int *stack_size*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessAttr_t* -
in	*stack_size*	int - stack size(byte)

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
3 } CL_ProcessAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateAttrSetUid (CL_ProcessAttr_t * *attr*, uid_t *uid*)

Type

No match

Brief

Parameters:

in,out	*attr*	const CL_ProcessAttr_t* -
in	*uid*	uid_t - user ID

CL_ProcessAttr_t struct

```
1 typedef struct {
2   char body[148];
```

```
3 } CL_ProcessAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateCgroupAttrInit (CL_ProcessCreateCgroupAttr_t * *attr*)

Brief

Parameters:

out	*attr*	const CL_ProcessCreateCgroupAttr_t* -

CL_ProcessCreateCgroupAttr_t struct

```
1 typedef struct {
2   char body[24];
3 } CL_ProcessCreateCgroupAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateCgroupAttrSetCfsBandwidthControl (CL_ProcessCreateCgroupAttr_t * *attr*, int *cfs_quota_us*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessCreateCgroupAttr_t* -
in	*cfs_quota_us*	int -

CL_ProcessCreateCgroupAttr_t struct

```
1 typedef struct {
2   char body[24];
3 } CL_ProcessCreateCgroupAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateCgroupAttrSetCpuShares (CL_ProcessCreateCgroupAttr_t * *attr*, int *cpu_shares*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessCreateCgroupAttr_t* -

in	*cpu_shares*	int -

CL_ProcessCreateCgroupAttr_t struct

```
1 typedef struct {
2   char body[24];
3 } CL_ProcessCreateCgroupAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateCgroupAttrSetMemoryLimit (CL_ProcessCreateCgroupAttr_t * *attr*, int *memory_limit*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessCreateCgroupAttr_t* -
in	*memory_limit*	int - memory limit (byte)

CL_ProcessCreateCgroupAttr_t struct

```
1 typedef struct {
2   char body[24];
3 } CL_ProcessCreateCgroupAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateCgroupAttrSetMemoryUsageNotification
(CL_ProcessCreateCgroupAttr_t * *attr*, int *usage_in_bytes*, int *event_fd*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessCreateCgroupAttr_t* -
in	*usage_in_bytes*	int - usage in bytes(byte)
in	*event_fd*	int -

CL_ProcessCreateCgroupAttr_t struct

```
1 typedef struct {
2   char body[24];
3 } CL_ProcessCreateCgroupAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateCgroupAttrSetRtThrottling (CL_ProcessCreateCgroupAttr_t * *attr*, int *runtime_us*)

Brief

Parameters:

in,out	*attr*	const CL_ProcessCreateCgroupAttr_t* -
in	*runtime_us*	int -

CL_ProcessCreateCgroupAttr_t struct

```
1 typedef struct {
2   char body[24];
3 } CL_ProcessCreateCgroupAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateCgroupClassify (const char * *cgroup_name*, pid_t *pid*)

Brief

Parameters:

in,out	*cgroup_name*	const char* -
in	*pid*	pid_t - process ID

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessCreateCgroupCreate (const char * *cgroup_name*, CL_ProcessCreateCgroupAttr_t * *attr*)

Brief

Parameters:

in	*cgroup_name*	const char* - cgroup name
in	*attr*	const CL_ProcessCreateCgroupAttr_t* -

CL_ProcessCreateCgroupAttr_t struct

```
1 typedef struct {
2   char body[24];
3 } CL_ProcessCreateCgroupAttr_t;
```

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

Open Close

See also:

CL_ProcessCreateCgroupDelete, CL_ProcessCreateCgroupAttrInit,
CL_ProcessCreateCgroupAttrSetCfsBandwidthControl,
CL_ProcessCreateCgroupAttrSetCpuShares, CL_ProcessCreateCgroupAttrSetMemoryLimit,
CL_ProcessCreateCgroupAttrSetMemoryUsageNotification,
CL_ProcessCreateCgroupAttrSetRtThrottling,

int CL_ProcessCreateCgroupDelete (const char * *cgroup_name*)

Brief

Parameters:

in	*cgroup_name*	const char* - cgroup name

Return values:

0	Success
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

Open Close

See also:

CL_ProcessCreateCgroupCreate

int CL_ProcessInit (void)

Brief

Return values:

int	file descriptor
-1	Error

Prerequisite

Prerequisites are nothing.

Change of internal state

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

Classification

Public

Type

No match

See also:

none

int CL_ProcessTerminate (pid_t *pid*)

Brief

Parameters:

in	*pid*	pid_t - PID

Return values:

int	file descriptor
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

Open Close

See also:

CL_ProcessCreate

int CL_ProcessTerminateGroup (pid_t *pid*)

Brief

Parameters:

in	*pid*	pid_t - process group ID

Return values:

int	file descriptor
-1	Error

Prerequisite

Change of internal state

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

Classification

Public

Type

Open Close

See also:

CL_ProcessCreate

cl_region_t* CL_RegionCreate (size_t *size*)

Brief

Region create

Parameters:

in	*size*	size_t - the size to create

Return values:

0	Success
-1	Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

void CL_RegionDestroy (cl_region_t * *region*)

Brief

Region destroy

Parameters:

in	*region*	cl_region_t * - the region to destroy

Return values:

0	Success
-1	Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

bool CL_RegionFree (cl_region_t * *region*, void * *p*)

Brief

Region free

Parameters:

in	*region*	cl_region_t * - the region to free
in	*p*	void * - the pointer to region

Return values:

true	Success
false	Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

EXT_C int CL_SemWait (sem_t * *semid*, unsigned int *timeout*)

Brief

Sem wait

Parameters:

in	*semid*	sem_t * - the pointer to the union sem_t
in	*timeout*	unsigned int - the count of timeout

Return values:

0	Success
not	0 Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

CL_TlsData_t* CL_TlsGet (void)

Brief

Get CL_TlsData_t

Return values:

the	pointer to CL_TlsData_t

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

int CL_TlsGetRcvFD (void)

Brief

Get receive fd of CL_TlsData_t

Return values:

receive	fd of CL_TlsData_t

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

void CL_TlsInit (int *pno*)

Brief

Tls initialize

Parameters:

in	*pno*	int - the pno of CL_TlsData_t

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

void CL_TlsSetRcvFD (int *fd*)

Brief

Set receive fd of CL_TlsData_t

Parameters:

in	*fd*	int - the receive fd of CL_TlsData_t

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

int CL_TlsThreadSeqID (void)

Brief

Get id of CL_TlsData_t

Return values:

the	id of CL_TlsData_t

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync

See also:

None

Notification_persistent_service

struct _NC_register_notif_msg
struct _NC_register_multiple_notif_msg
struct _NC_register_immediate_notif_msg
struct _NC_register_multiple_immediate_notif_msg
struct _NC_unregister_notif_msg
struct _NC_unregister_multiple_notif_msg
struct _NC_savepersdata_ack
struct _NC_subscribe_multiple_notif_msg
struct _NC_unsubscribe_multiple_notif_msg
struct __NC_get_persdata_failed_ack
struct _NC_User
struct _NC_RegisterPersistentFileMsg
struct _NC_LoadPersistedFileMsg

struct _NC_ReleasePersistentFileMsg
struct _NC_RegisterPersistentFolderMsg
struct _NC_LoadPersistedFolderMsg
struct _NC_ReleasePersistentFolderMsg
struct _NC_LoadPersistedAck
struct _NC_NorPersistentData
struct _NC_ClearPersistedDataReq
struct _NC_ClearPersisteDatadAck
struct _NC_StopMsgData
struct _NC_SetPersistType
struct _NC_ImmediateWriteAck
struct _NotificationInfo
struct _ImmediateNotificationInfo
struct _SubscribeInfo

Macros

#define **NTFY_NPPService_UserChange** "NPPService/UserChange"

Typedefs

typedef struct _NC_register_notif_msg NC_register_notif_msg
typedef struct _NC_register_multiple_notif_msg NC_register_multiple_notif_msg
typedef struct _NC_register_immediate_notif_msg NC_register_immediate_notif_msg
typedef struct _NC_register_multiple_immediate_notif_msg
NC_register_multiple_immediate_notif_msg
typedef struct _NC_unregister_notif_msg NC_unregister_notif_msg
typedef struct _NC_unregister_notif_msg **NC_subscribe_msg**
typedef struct _NC_unregister_notif_msg **NC_unsubscribe_frm_notif_msg**
typedef struct _NC_unregister_notif_msg **NC_get_pers_data_msg**
typedef struct _NC_unregister_multiple_notif_msg NC_unregister_multiple_notif_msg
typedef struct _NC_savepersdata_ack NC_savepersdata_ack
typedef struct _NC_subscribe_multiple_notif_msg NC_subscribe_multiple_notif_msg
typedef struct _NC_unsubscribe_multiple_notif_msg NC_unsubscribe_multiple_notif_msg
typedef struct __NC_get_persdata_failed_ack NC_get_persdata_failed_ack
typedef struct _NC_User NC_User
typedef struct _NC_RegisterPersistentFileMsg NC_RegisterPersistentFileMsg
typedef struct _NC_LoadPersistedFileMsg NC_LoadPersistedFileMsg
typedef struct _NC_ReleasePersistentFileMsg NC_ReleasePersistentFileMsg
typedef struct _NC_RegisterPersistentFolderMsg NC_RegisterPersistentFolderMsg
typedef struct _NC_LoadPersistedFolderMsg NC_LoadPersistedFolderMsg
typedef struct _NC_ReleasePersistentFolderMsg NC_ReleasePersistentFolderMsg
typedef struct _NC_LoadPersistedAck NC_LoadPersistedAck
typedef struct _NC_LoadPersistedAck **NC_ReleasePersistedAck**
typedef struct _NC_NorPersistentData NC_NorPersistentData
typedef struct _NC_ClearPersistedDataReq NC_ClearPersistedDataReq
typedef struct _NC_ClearPersisteDatadAck NC_ClearPersisteDatadAck
typedef struct _NC_StopMsgData NC_StopMsgData
typedef struct _NC_SetPersistType NC_SetFilePersistType
typedef struct _NC_SetPersistType **NC_SetFolderPersistType**

typedef struct _NC_ImmediateWriteAck NC_ImmediateWriteAck
typedef struct _NotificationInfo NotificationInfo
typedef struct _ImmediateNotificationInfo ImmediateNotificationInfo
typedef struct _SubscribeInfo SubscribeInfo
typedef enum _NS_NPServiceProtocol NS_NPServiceProtocol
typedef enum _NS_NPServiceEvent NS_NPServiceEvent

Enumerations

enum _NS_NPServiceProtocol { **NPS_REGISTER_EV_REQ** = PROTOCOL__CWORD33__BASE_CMD + 0, NPS_UNREGISTER_EV_REQ, NPS_PUBLISH_EV_REQ, NPS_NOTIFY_EV_REQ, NPS_SUBSCRIBE_TO_EV_REQ, NPS_UNSUBSCRIBE_FROM_EV_REQ, NPS_SET_PERSIST_FILE_PATH_REQ, NPS_GET_PERS_DATA_REQ, NPS_GET_PERS_DATA_ACK, NPS_GET_PERS_FILE_REQ, NPS_GET_PERS_FILE_ACK, NPS_SAVE_PERS_DATA_REQ, NPS_SAVE_PERS_DATA_ACK, NPS_RELEASE_PERS_FILE_REQ, NPS_NPP_STOP_REQ, NPS_TST_WAKEUP, NPS_BATCH_SUBSCRIBE_TO_EV_REQ, NPS_GET_PERS_DATA_FAILED_ACK, NPS_SET_PERSONALITY_REQ, NPS_CHANGE_PERSONALITY_REQ, NPS_USER_CHANGE_REQ, NPS_SET_PERSIST_FOLDER_PATH_REQ, NPS_GET_PERS_FOLDER_REQ, NPS_RELEASE_PERS_FOLDER_REQ, NPS_GET_PERS_FOLDER_ACK, NPS_NPP_STOP_ACK, NPS_BATCH_UNSUBSCRIBE_FROM_EV_REQ, NPS_GET_READYSTATUS_REQ, NPS_GET_READYSTATUS_ACK, NPS_REGISTER_NOR_EV_REQ, NPS_DELETE_PERSISTED_DATA_REQ, NPS_DELETE_PERSISTED_DATA_ACK, NPS_SET_DEFAULT_PERS_DATA, NPS_SET_NOTFN_PERSISTENT_TYPE, NPS_SET_FILE_PERSISTENT_TYPE, NPS_SET_FOLDER_PERSISTENT_TYPE, NPS_SYNCHRONOUS_WRITE_NOTIFY_REQ, NPS_IMMEDIATE_WRITE_ACK, NPS_NPP_SYNC_REQ }
enum _NS_NPServiceEvent { **NPS_NPP_READY_EVENT** = *** }

Functions

E_CWORD33_Status NPRegisterNotifications (HANDLE hNPMsgQ, PCSTR pPublisherName, UI_32 numNotifications, NotificationInfo *pNotificationArray)
E_CWORD33_Status NPRegisterImmediateNotifications (HANDLE hNPMsgQ, PCSTR pPublisherName, UI_32 numNotifications, ImmediateNotificationInfo *pNotificationArray)
E_CWORD33_Status NPRegisterNotification (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR notif_name, const UI_32 max_length, const E_CWORD33_NotificationType perstype)
E_CWORD33_Status NPUnRegisterNotification (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pNotification)
E_CWORD33_Status NPSetPersistentNotfnType (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pNotification, E_CWORD33_PersistCategory ePersistCategory)
E_CWORD33_Status NPSetPersistNotfnDefaultValue (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pNotification, PCVOID pData, const UI_32 iLength)
E_CWORD33_Status NPUnRegisterNotifications (HANDLE hNPMsgQ, PCSTR pPublisherName, UI_32 numNotifications, NotificationInfo *pNotificationArray)
E_CWORD33_Status NPPublishNotification (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pNotification, PCVOID pData, const UI_32 iLength)
E_CWORD33_Status NPSubscribeToNotification (HANDLE hNPMsgQ, PCSTR pSubscriberName, PCSTR notif_name)
E_CWORD33_Status NPSubscribeToNotifications (HANDLE hNPMsgQ, PCSTR pSubscriberName, UI_32 numNotifications, SubscribeInfo *pSubscribeInfoArray)

E_CWORD33_Status NPUnsubscribeFromNotification (HANDLE hNPMsgQ, PCSTR pSubscriberName, PCSTR pNotification)
E_CWORD33_Status NPUnsubscribeFromNotifications (HANDLE hNPMsgQ, PCSTR pUnSubscriberName, UI_32 numNotifications, SubscribeInfo *pUnSubscribeInfoArray)
E_CWORD33_Status NPReadPersistedData (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR notification)
E_CWORD33_Status NPSavePersistentData (HANDLE hNPMsgQ, PCSTR pPublisherName)
E_CWORD33_Status NPRegisterPersistentFile (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pTag, BOOL bIsUserFile)
E_CWORD33_Status NPSetFilePersistentType (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pTag, E_CWORD33_PersistCategory ePersistCategory)
E_CWORD33_Status NPLoadPersistentFile (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pDstFilePath, PCSTR pTag, HANDLE hUser)
E_CWORD33_Status NPReleasePersistentFile (HANDLE hNPMsgQ, PCSTR pPublisherName, E_CWORD33_ReleaseType eReleaseType, PCSTR pTag, PCSTR pFullFilePath, HANDLE hUser)
E_CWORD33_Status NPPersistentSync (PCSTR SrcName, HANDLE hNPMsgQ, UI_32 sessionid, PCSTR pPublisherName)
E_CWORD33_Status NPSetPersonality (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pUserName)
E_CWORD33_Status NPChangePersonality (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pUserName)
E_CWORD33_Status NPGetReadyStatusOfNPP (HANDLE hNPMsgQ, PCSTR pRequesterName)
E_CWORD33_Status NPClearPersistedData (HANDLE hNPMsgQ, PCSTR pRequesterName, E_CWORD33_ClearPersistence e_CWORD33_ClearPersistenceScope)
E_CWORD33_Status NPRegisterPersistentFolder (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pTag, BOOL bIsUserFolder)
E_CWORD33_Status NPSetFolderPersistentType (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pTag, E_CWORD33_PersistCategory ePersistCategory)
E_CWORD33_Status NPLoadPersistentFolder (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pDstFolderPath, PCSTR pTag, HANDLE hUser)
E_CWORD33_Status NPReleasePersistentFolder (HANDLE hNPMsgQ, PCSTR pPublisherName, E_CWORD33_ReleaseType e_CWORD33_ReleaseType, PCSTR pTag, PCSTR pFullFolderPath, HANDLE hUser)
E_CWORD33_Status NPSynchronousReadPersistentData (PCSTR pAppName, PCSTR notif_name, PVOID pData, UI_32 uiDataSize, const E_CWORD33_PersistCategory ePersistCategory=e_CWORD33_UserData)
E_CWORD33_Status NPSynchronousWritePersistentData (PCSTR pAppName, PCSTR notif_name, PVOID pData, const UI_32 uiDataSize, const E_CWORD33_PersistCategory ePersistCategory=e_CWORD33_UserData)

Detailed Description

Class Documentation

struct _NC_register_notif_msg

A structure to register notification msg

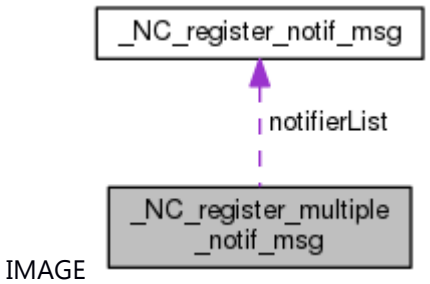
Class Members:

UI_8	dummy[4]	Packing to 32 bit boundary.
UI_32	maxLength	Maximum possible length (bytes) of.
CHAR	notificationName[MAX_STRING_SIZE_NOTIFICATION]	notification event name string
E_CWORD33_NotificationType	persType	Specifies persitence for this notification.

struct _NC_register_multiple_notif_msg

A structure to register multiple notification msg

Collaboration diagram for _NC_register_multiple_notif_msg:



IMAGE

Class Members:

NC_register_notif_msg	notifierList[1]	variable number of structures
UI_32	numNotifications	This has to be UI_32 to avoid holes.

struct _NC_register_immediate_notif_msg

A structure to register immediate notification msg

Class Members:

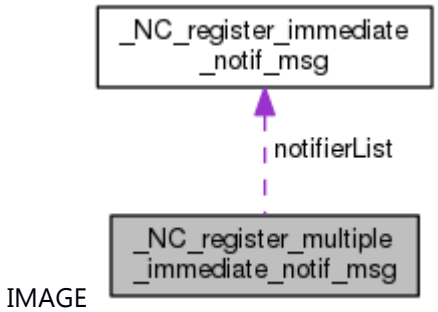
UI_32	delay	Delay time for persistence.
UI_8	dummy[4]	Packing to 32 bit boundary.
UI_32	maxLength	Maximum possible

		length (bytes) of.
CHAR	notificationName[MAX_STRING_SIZE_NOTIFICATION]	Notification Event Name string.
E_CWORD33_NotificationType	persType	Specifies persitence for this notification.

struct _NC_register_multiple_immediate_notif_msg

A structure to register multiple immediate notification msg

Collaboration diagram for _NC_register_multiple_immediate_notif_msg:



IMAGE

Class Members:

NC_register_immediate_notif_msg	notifierList[1]	variable number of structures
UI_32	numNotifications	This has to be UI_32 to avoid holes.

struct _NC_unregister_notif_msg

A structure to unregister notification msg

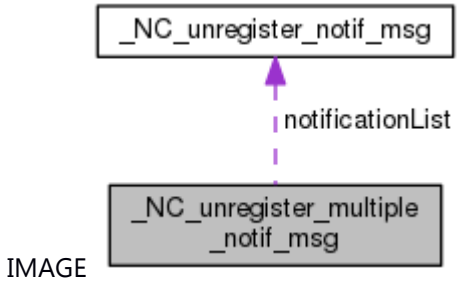
Class Members:

CHAR	notificationName[MAX_STRING_SIZE_NOTIFICATION]	Notification Event Name string.

struct _NC_unregister_multiple_notif_msg

A structure to unregister multiple notification msg

Collaboration diagram for _NC_unregister_multiple_notif_msg:



IMAGE

Class Members:

NC_unregister_notif_msg	notificationList[1]	notification event name List
UI_32	numNotifications	notification event name num

struct _NC_savepersdata_ack

A structure to save pers data

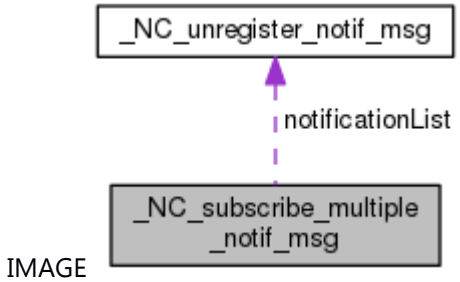
Class Members:

E_CWORD33_Status	eStatus	e_CWORD33_StatusOK is (valid - data stored, anything else is

struct _NC_subscribe_multiple_notif_msg

A structure to subscribe multiple notification msg

Collaboration diagram for _NC_subscribe_multiple_notif_msg:



IMAGE

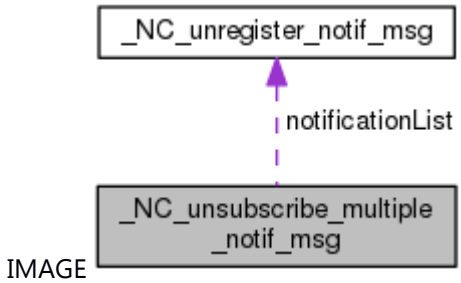
Class Members:

NC_subscribe_msg	notificationList[1]	notification event name List
UI_32	numNotifications	notification event name num

struct _NC_unsubscribe_multiple_notif_msg

A structure to subscribe multiple notification msg

Collaboration diagram for _NC_unsubscribe_multiple_notif_msg:



IMAGE

Class Members:

NC_unsubscribe_frm_notif_msg	notificationList[1]	notification event name List
UI_32	numNotifications	notification event name num

struct __NC_get_persdata_failed_ack

A structure to get pers data failed

Class Members:

CHAR	notificationName[MAX_STRING_SIZE_NOTIFICATION]	notification name

struct _NC_User

A structure to set personality

Class Members:

CHAR	cUsername[MAX_PATH_LENGTH]	name of personality

struct _NC_RegisterPersistentFileMsg

A structure to register persistent file with the notification_persistent_service

Class Members:

BOOL	bIsUserFile	is user file
CHAR	cFileTag[MAX_STRING_SIZE_TAG]	Tag associated with the.

struct _NC_LoadPersistedFileMsg

A structure to specify where to load the file associated with the Tag

Class Members:

CHAR	cFilePath[MAX_PATH_LENGTH]	path to which the file needs to retrieved
CHAR	cFileTag[MAX_STRING_SIZE_TAG]	Tag associated with the file.
CHAR	cUsername[MAX_PATH_LENGTH]	Name of the user for user specific file.

struct _NC_ReleasePersistentFileMsg

A structure to release persistent file

Class Members:

CHAR	cFilePath[MAX_PATH_LENGTH]	path from which the

CHAR	cFileTag[MAX_STRING_SIZE_NOTIFICATION]	Tag associated with the.
CHAR	cUsername[MAX_PATH_LENGTH]	Name of the user for user specific file.
E_CWORD33_ReleaseType	e_CWORD33_ReleaseType	should the file be persisted

struct _NC_RegisterPersistentFolderMsg

A structure to register persistent folder with the notification_persistent_service

Class Members:

BOOL	bIsUserFolder	Check if the folder is of user type.
CHAR	cFolderTag[MAX_STRING_SIZE_TAG]	Tag associated with the.

struct _NC_LoadPersistedFolderMsg

A structure to specify where to load the folder associated with the Tag

Class Members:

CHAR	cFolderPath[MAX_PATH_LENGTH]	path to which the folder needs to retrieved
CHAR	cFolderTag[MAX_STRING_SIZE_TAG]	Tag associated with the folder.
CHAR	cUsername[MAX_PATH_LENGTH]	Name of the user for user specific folder.

struct _NC_ReleasePersistentFolderMsg

A structure to Release Persistent Folder Msg

Class Members:

CHAR	cFolderPath[MAX_PATH_LENGTH]	path from which the folder
CHAR	cFolderTag[MAX_STRING_SIZE_NOTIFICATION]	Tag associated with.
CHAR	cUsername[MAX_PATH_LENGTH]	Name of the user for user.
E_CWORD33_ReleaseType	e_CWORD33_ReleaseType	should the file be

		persisted

struct _NC_LoadPersistedAck

A structure to Load Persisted Ack

Class Members:

CHAR	cTag[MAX_PATH_LENGTH]	tag of the file/folder for which acknowledge is made
E_CWORD33_Status	eStatus	ack the success/failure in

struct _NC_NorPersistentData

A structure to Nor Persistent Data

Class Members:

UI_32	dataSize	data size
CHAR	notificationName[MAX_STRING_SIZE_NOTIFICATION]	Notification Event Name string.
CHAR	pPublisherName[MAX_NAME_SIZE_APP]	Publisher Event Name string.
UI_32	uiDelay	ui Delay
UI_32	uiMaxSize	ui Max Size

struct _NC_ClearPersistedDataReq

A structure to request for deleting data from persistent memory

Class Members:

E_CWORD33_ClearPersistence	ePersistenceData	defines what

struct _NC_ClearPersisteDatadAck

A structure for receiving ack of data deleted from persistent memory

Class Members:

E_CWORD33_Status	eStatus	ack status of ClearPersistedData request

struct _NC_StopMsgData

A structure to request stop message to notification_persistent_service.

Class Members:

E_CWORD33_ShutdownType	eShutdownType	shutdown type
UI_32	uiStopMsgData	Reason for sending stop message to NPPService.

struct _NC_SetPersistType

A structure to set the persist type of file and folder

Class Members:

CHAR	cTag[MAX_PATH_LENGTH]	tag name of the file or folder
E_CWORD33_PersistCategory	ePersistType	persist type

struct _NC_ImmediateWriteAck

A data received with ack sent by notification_persistent_service when immediate notification data is written in persistent memory

Class Members:

E_CWORD33_Status	eStatus	status of request
CHAR	notificationName[MAX_STRING_SIZE_NOTIFICATION]	notification Name

struct _NotificationInfo

A structure to Notification Info

Class Members:

UI_8	dummy[4]	Packing to 32 bit boundary.
UI_32	maxLength	Maximum possible length (bytes) of the message data.
CHAR	notificationName[MAX_STRING_SIZE_NOTIFICATION]	Notification Event Name string.
E_CWORD33_NotificationType	persType	Specifies persitence for this notification.

struct _ImmediateNotificationInfo

A structure to Immediate Notification Info

Class Members:

UI_32	delay	
UI_8	dummy[4]	Packing to 32 bit boundary.
UI_32	maxLength	Maximum possible length (bytes) of the message data.
CHAR	notificationName[MAX_STRING_SIZE_NOTIFICATION]	Notification Event Name string.
E_CWORD33_NotificationType	persType	Specifies persitence for this notification.

struct _SubscribeInfo

A structure to Subscribe Info

Class Members:

CHAR	notificationName[MAX_STRING_SIZE_NOTIFICATION]	Notification Event Name string.

Typedef Documentation

typedef struct _ImmediateNotificationInfo ImmediateNotificationInfo

A structure to Immediate Notification Info

typedef struct _NC_ClearPersisteDatadAck NC_ClearPersisteDatadAck

A structure for receiving ack of data deleted from persistent memory

typedef struct _NC_ClearPersistedDataReq NC_ClearPersistedDataReq

A structure to request for deleting data from persistent memory

typedef struct __NC_get_persdata_failed_ack NC_get_persdata_failed_ack

A structure to get pers data failed

typedef struct _NC_ImmediateWriteAck NC_ImmediateWriteAck

A data received with ack sent by notification_persistent_service when immediate notification data is written in persistent memory

typedef struct _NC_LoadPersistedAck NC_LoadPersistedAck

A structure to Load Persisted Ack

typedef struct _NC_LoadPersistedFileMsg NC_LoadPersistedFileMsg

A structure to specify where to load the file associated with the Tag

typedef struct _NC_LoadPersistedFolderMsg NC_LoadPersistedFolderMsg

A structure to specify where to load the folder associated with the Tag

typedef struct _NC_NorPersistentData NC_NorPersistentData

A structure to Nor Persistent Data

typedef struct _NC_register_immediate_notif_msg NC_register_immediate_notif_msg

A structure to register immediate notification msg

typedef struct _NC_register_multiple_immediate_notif_msg NC_register_multiple_immediate_notif_msg

A structure to register multiple immediate notification msg

typedef struct _NC_register_multiple_notif_msg NC_register_multiple_notif_msg

A structure to register multiple notification msg

typedef struct _NC_register_notif_msg NC_register_notif_msg

A structure to register notification msg

typedef struct _NC_RegisterPersistentFileMsg NC_RegisterPersistentFileMsg

A structure to register persistent file with the notification_persistent_service

typedef struct _NC_RegisterPersistentFolderMsg NC_RegisterPersistentFolderMsg

A structure to register persistent folder with the notification_persistent_service

typedef struct _NC_ReleasePersistentFileMsg NC_ReleasePersistentFileMsg

A structure to release persistent file

typedef struct _NC_ReleasePersistentFolderMsg NC_ReleasePersistentFolderMsg

A structure to Release Persistent Folder Msg

typedef struct _NC_savepersdata_ack NC_savepersdata_ack

A structure to save pers data

typedef struct _NC_SetPersistType NC_SetFilePersistType

A structure to set the persist type of file and folder

typedef struct _NC_StopMsgData NC_StopMsgData

A structure to request stop message to notification_persistent_service.

typedef struct _NC_subscribe_multiple_notif_msg NC_subscribe_multiple_notif_msg

A structure to subscribe multiple notification msg

typedef struct _NC_unregister_multiple_notif_msg NC_unregister_multiple_notif_msg

A structure to unregister multiple notification msg

typedef struct _NC_unregister_notif_msg NC_unregister_notif_msg

A structure to unregister notification msg

typedef struct _NC_unsubscribe_multiple_notif_msg NC_unsubscribe_multiple_notif_msg

A structure to subscribe multiple notification msg

typedef struct _NC_User NC_User

A structure to set personality

typedef struct _NotificationInfo NotificationInfo

A structure to Notification Info

typedef enum _NS_NPServiceEvent NS_NPServiceEvent

notification_persistent_service service event

typedef enum _NS_NPServiceProtocol NS_NPServiceProtocol

notification_persistent_service protocol msg name define

typedef struct _SubscribeInfo SubscribeInfo

A structure to Subscribe Info

Enumeration Type Documentation

enum _NS_NPServiceEvent

notification_persistent_service service event

enum _NS_NPServiceProtocol

notification_persistent_service protocol msg name define

Enumerator

NPS_UNREGISTER_EV_REQ <16+0 Request to register a Notification

NPS_PUBLISH_EV_REQ <1 Request to unregister a Notification

NPS_NOTIFY_EV_REQ <2 Sender request to NP service to publish (send) a notification to subscribers

NPS_SUBSCRIBE_TO_EV_REQ <3 Message to subscribers that the notification was generated by publisher

NPS_UNSUBSCRIBE_FROM_EV_REQ <4 Request to subscribe to a notification

NPS_SET_PERSIST_FILE_PATH_REQ <5 Request to unsubscribe from a notification

NPS_GET_PERS_DATA_REQ <6 Request to add a path to the list of files to be persisted

NPS_GET_PERS_DATA_ACK <7 Request to get persistent data for specified notification

NPS_GET_PERS_FILE_REQ <8 MSG from NPS to requester of persistent data.

NPS_GET_PERS_FILE_ACK <9 Request to get persistent File from specified path

NPS_SAVE_PERS_DATA_REQ <10 MSG from NPS to inform that file was retrieved from persistent mem and copied.

NPS_SAVE_PERS_DATA_ACK <11 Request to save the persistent data to what ever file that has been defined.

NPS_RELEASE_PERS_FILE_REQ <12 MSG from NPS to requester of Save Action.

NPS_NPP_STOP_REQ <13 Request to NPS to persist file in persistent memory

NPS_TST_WAKEUP <14 MSG from system manager to NS_NPS to stop processing. NS_NPS goes to idle state.

NPS_BATCH_SUBSCRIBE_TO_EV_REQ <15

NPS_GET_PERS_DATA_FAILED_ACK <16 Request to subscribe to a batch of notification(s)

NPS_SET_PERSONALITY_REQ <17 Failed to get persistent data

NPS_CHANGE_PERSONALITY_REQ <18 Request to change current user

NPS_USER_CHANGE_REQ <19 Request to change current user

NPS_SET_PERSIST_FOLDER_PATH_REQ <20 MSG from NPS to *CWORD33* applications.

NPS_GET_PERS_FOLDER_REQ <21 Request to add a path to the folder to be persisted

NPS_RELEASE_PERS_FOLDER_REQ <22 Request to get persistent Folder from specified path

NPS_GET_PERS_FOLDER_ACK <23 Request to NPS to persist folder in persistent memory

NPS_NPP_STOP_ACK <24 Ack from NPS to requester to inform that folder was retrieved from persistent mem and copied.

NPS_BATCH_UNSUBSCRIBE_FROM_EV_REQ <25 Ack from NS_NPS to system manager on NPS_NPP_STOP_REQ request

NPS_GET_READYSTATUS_REQ <26

NPS_GET_READYSTATUS_ACK <27 Request from system manager to check the ready status of NPPService

NPS_REGISTER_NOR_EV_REQ <28 Ack from NPPService to system manager, only if NPPService is in ready state

NPS_DELETE_PERSISTED_DATA_REQ <29 Request to register immediate notification

NPS_DELETE_PERSISTED_DATA_ACK <30 Request to delete the persistent data.

NPS_SET_DEFAULT_PERS_DATA <31 Ack of deleting the persistent data.

NPS_SET_NOTFN_PERSISTENT_TYPE <32 Sets the default data for persistent notification.

NPS_SET_FILE_PERSISTENT_TYPE <33 Sets the persistent type of notification.

NPS_SET_FOLDER_PERSISTENT_TYPE <34 Sets the persistent type of file.

NPS_SYNCHRONOUS_WRITE_NOTIFY_REQ <35 Sets the persistent type of folder.

NPS_IMMEDIATE_WRITE_ACK when written using synchronous API.

<36 Updates the immediate notification data in NPPService,

NPS_NPP_SYNC_REQ <38 Processing which synchronizes by NPPService (syncfs)

<37 Sends the ack to publisher of immediate notification after data is written to persistent memory.

Function Documentation

E_CWORD33_Status NPChangePersonality (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pUserName*)

Brief

API to send message to Notification Service to change the current Personality.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*pUserName*	PCSTR - Name of new Personality

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
```

```
20  e_CWORD33_StatusInvldBufSize       = ***,  ///< Buf size too small
21  e_CWORD33_StatusInvldID            = ***,  ///< Unrecognized ID
22  e_CWORD33_StatusCannotRelease      = ***, ///< Cannot release resource
23  e_CWORD33_StatusBadConnection      = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit           = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented     = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy         = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin     = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist     = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault              = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing         = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer        = ***,
35  e_CWORD33_StatusMsgNotProcessed    = ***,
36  e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPChangePersonality

E_CWORD33_Status NPClearPersistedData (HANDLE *hNPMsgQ*, PCSTR *pRequesterName*, E_CWORD33_ClearPersistence *e_CWORD33_ClearPersistenceScope*)

Brief

API to delete the persisted data of NS_NPS. Note: Currently to be used by HMI service only

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of

		Notification service.
in	*pRequesterName*	PCSTR - Name of the requesters message queue.
in	*e_CWORD33_ClearPersistenceScope*	E_CWORD33_ClearPersistence - specifies what data to delete from persistent memory.

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1  typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize      = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease     = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection     = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented    = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy        = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin    = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29   e_CWORD33_StatusThreadNotExist    = ***,  ///< The thread does not exist
30   e_CWORD33_StatusFault             = ***,  ///< A fault occurred while attempting to make call
31   e_CWORD33_StatusServNotFound      = ***,  ///< Service not present in serv dir
32   e_CWORD33_StatusServerInUse       = ***,  ///< Service already processing 1 client request
33   e_CWORD33_StatusDbIndexing        = ***,  ///< Database Indexing in progress
34   e_CWORD33_StatusNullPointer       = ***,
35   e_CWORD33_StatusMsgNotProcessed    = ***,
36   e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37   e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38   e_CWORD33_StatusAccessError       = ***, ///< Error when accessing resource
39   e_CWORD33_StatusDuplicate         = ***, ///< Duplicate entry
40   e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41   e_CWORD33_StatusThreadAlreadyRunning  = ***,
42   e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43   e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44   e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
```

```
45   e_CWORD33_StatusSessionErr      = ***,  ///< Error in session handling
46   e_CWORD33_StatusDBCorrupt       = ***,  ///< Database corrupt
47   e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPClearPersistedData

E_CWORD33_Status NPGetReadyStatusOfNPP (HANDLE *hNPMsgQ*, PCSTR *pRequesterName*)

Brief

To be used by System Manager Service only. SystemManager will use this API to get the ready status of NPPService.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pRequesterName*	PCSTR - Name of the requesters message queue.
in	*pUserName*	PCSTR - Name of new Personality

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
```

```
15  e_CWORD33_StatusInvldHndlType       = ***,  ///< Invalid handle type
16  e_CWORD33_StatusInvldQName          = ***,  ///< Invalid messasge queue name
17  e_CWORD33_StatusMsgQFull            = ***,  ///< Message queue full
18  e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19  e_CWORD33_StatusInvldParam          = ***,  ///< Invalid parameter
20  e_CWORD33_StatusInvldBufSize        = ***,  ///< Buf size too small
21  e_CWORD33_StatusInvldID             = ***,  ///< Unrecognized ID
22  e_CWORD33_StatusCannotRelease       = ***, ///< Cannot release resource
23  e_CWORD33_StatusBadConnection       = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit            = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented      = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy          = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin      = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal    = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist      = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault               = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound        = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse         = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing          = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer         = ***,
35  e_CWORD33_StatusMsgNotProcessed     = ***,
36  e_CWORD33_StatusFileLoadSuccess     = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError       = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError         = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate           = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty           = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF          = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN         = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR          = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr      = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt       = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

_CWORD33_NPGetReadyStatusOfNPP

E_CWORD33_Status NPLoadPersistentFile (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pDstFilePath*, PCSTR *pTag*, HANDLE *hUser*)

Brief

API to send message to Notification Service to copy file from persistent memory to specified path. The caller recieves an acknowledgement once NPS completes file copy

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*pDstFilePath*	PCSTR - Complete file path to which the file should be copied from persistent area.
in	*pTag*	PCSTR - Tag associated with the file that needs to be copied to pDstFilePath.
in	*hUser*	HANDLE - Load the file for the specified user.

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1  typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification  = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize      = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease     = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection     = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented    = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy        = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin    = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal  = ***, ///< Invalid value passed
29   e_CWORD33_StatusThreadNotExist    = ***,  ///< The thread does not exist
30   e_CWORD33_StatusFault             = ***,  ///< A fault occurred while attempting to make call
```

```
31  e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing         = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer        = ***,
35  e_CWORD33_StatusMsgNotProcessed    = ***,
36  e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

_CWORD33_NPLoadPersistentFile

E_CWORD33_Status NPLoadPersistentFolder (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pDstFolderPath*, PCSTR *pTag*, HANDLE *hUser*)

Brief

API to send message to Notification Service to copy folder from persistent memory to specified path. The caller recieves an acknowledgement once NPS completes folder copy.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*pDstFolderPath*	PCSTR - Complete folder path to which the folder should be copied from persistent area
in	*pTag*	PCSTR - Tag associated with the folder that needs to be copied to pDstFolderPath
in	*hUser*	HANDLE - Name of new Personality

Return values:

E_CWORD33_Status	indicates success or error

Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
typedef enum e_CWORD33_Status {
  e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
  e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
  e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
  e_CWORD33_StatusDbResultError      = ***, ///< Database result error
  e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
  e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
  e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
  e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
  e_CWORD33_StatusFail               = ***, ///< Failed
  e_CWORD33_StatusErrOther           = ***, ///< 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;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPLoadPersistentFolder

E_CWORD33_Status NPPersistentSync (PCSTR *SrcName*, HANDLE *hNPMsgQ*, UI_32 *sessionid*, PCSTR *pPublisherName*)

Brief

API to synchronizes by NPPService.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*eReleaseType*	E_CWORD33_ReleaseType - e_CWORD33_NotOnRelease:not on release e_CWORD33_PersistOnShutdown:persist on shutdown e_CWORD33_PersistInstantly:persist instantly
in	*SrcName*	PCSTR - source service name.
in	*sessionid*	UI_32 - session ID

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
```

```
17  e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18  e_CWORD33_StatusInvldNotification  = ***,///< The Notification event not present
19  e_CWORD33_StatusInvldParam       = ***,  ///< Invalid parameter
20  e_CWORD33_StatusInvldBufSize      = ***,  ///< Buf size too small
21  e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22  e_CWORD33_StatusCannotRelease     = ***, ///< Cannot release resource
23  e_CWORD33_StatusBadConnection     = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented    = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy        = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin    = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal  = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist    = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault             = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound      = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse       = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing      = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer     = ***,
35  e_CWORD33_StatusMsgNotProcessed   = ***,
36  e_CWORD33_StatusFileLoadSuccess   = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError     = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError       = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate         = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF          = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN         = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR          = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr    = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt     = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound   = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPPersistentSync

E_CWORD33_Status NPPublishNotification (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pNotification*, PCVOID *pData*, const UI_32 *iLength*)

Brief

API to send message to Notification Service to notify subscribers

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of Publisher message queue
in	*pNotification*	PCSTR - Name of Notification
in	*pData*	PVOID - Data buffer
in	*iLength*	const UI_32 - Size of data buffer

Return values:

e_CWORD33_StatusOK	
e_CWORD33_StatusInvldParam	
e_CWORD33_StatusMsgQFull	
e_CWORD33_StatusErrNoEBAD	
e_CWORD33_StatusErrNoEINTR	
e_CWORD33_StatusInvldBufSize	
e_CWORD33_StatusFail	
e_CWORD33_StatusErrOther	

Preconditons

-none

Change of internal status

-none

Classification

-pubulic

Type

Pub-Sub

See also:

McSendWithSysInfo

E_CWORD33_Status NPReadPersistedData (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *notification*)

Brief

API to requested the persistent data corresponding to the notification if available.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue
in	*notification*	PCSTR - Name of Notification

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
```

```
  e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
  e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
  e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
  e_CWORD33_StatusDbResultError      = ***, ///< Database result error
  e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
  e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
  e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
  e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
  e_CWORD33_StatusFail               = ***, ///< Failed
  e_CWORD33_StatusErrOther           = ***, ///< 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;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

E_CWORD33_NPReadPersistedData

E_CWORD33_Status NPRegisterImmediateNotifications (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, UI_32 *numNotifications*, ImmediateNotificationInfo * *pNotificationArray*)

Brief

API to send message to Notification Service to register a set of immediate notifications

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of Publisher message queue.
in	*numNotifications*	PCSTR - Name of Notification.
in	*pNotificationArray*	ImmediateNotificationInfo - Array of immediate notifications

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1  typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize      = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease     = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection     = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented    = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy        = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin    = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
```

```
29  e_CWORD33_StatusThreadNotExist      = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault               = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound        = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse         = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing          = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer         = ***,
35  e_CWORD33_StatusMsgNotProcessed     = ***,
36  e_CWORD33_StatusFileLoadSuccess     = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError       = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError         = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate           = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty           = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF          = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN         = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR          = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr      = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt       = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPRegisterImmediatePersistNotification

E_CWORD33_Status NPRegisterNotification (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *notif_name*, const UI_32 *max_length*, const E_CWORD33_NotificationType *perstype*)

Brief

API to send message to Notification Service to register a notification

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of Publisher message queue
in	*notif_name*	PCSTR - Name of Notification
in	*max_length*	const UI_32 - Max size of the notification message
in	*perstype*	const E_CWORD33_PersistentVarType - Flag to indicate if it has to be persistent

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize      = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease     = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection     = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented    = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy        = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin    = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29   e_CWORD33_StatusThreadNotExist    = ***,  ///< The thread does not exist
30   e_CWORD33_StatusFault             = ***,  ///< A fault occurred while attempting to make call
31   e_CWORD33_StatusServNotFound      = ***,  ///< Service not present in serv dir
32   e_CWORD33_StatusServerInUse       = ***,  ///< Service already processing 1 client request
33   e_CWORD33_StatusDbIndexing        = ***,  ///< Database Indexing in progress
34   e_CWORD33_StatusNullPointer       = ***,
35   e_CWORD33_StatusMsgNotProcessed    = ***,
36   e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37   e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38   e_CWORD33_StatusAccessError       = ***, ///< Error when accessing resource
39   e_CWORD33_StatusDuplicate         = ***, ///< Duplicate entry
40   e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41   e_CWORD33_StatusThreadAlreadyRunning  = ***,
42   e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43   e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44   e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
45   e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46   e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47   e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

NPRegisterNotifications, _CWORD33_NPRegisterNotification

E_CWORD33_Status NPRegisterNotifications (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, UI_32 *numNotifications*, NotificationInfo * *pNotificationArray*)

Brief

API to send message to Notification Service to register a set of notification

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of Publisher message queue
in	*numNotifications*	PCSTR - Name of Notification
in	*pNotificationArray*	NotificationInfo - Array of notifications

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
typedef enum e_CWORD33_Status {
  e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
  e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
  e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
  e_CWORD33_StatusDbResultError      = ***, ///< Database result error
  e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
  e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
  e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
  e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
  e_CWORD33_StatusFail               = ***, ///< Failed
  e_CWORD33_StatusErrOther           = ***, ///< 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
```

```
23  e_CWORD33_StatusBadConnection       = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented      = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy          = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin      = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal    = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist      = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault               = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound        = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse         = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing          = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer         = ***,
35  e_CWORD33_StatusMsgNotProcessed     = ***,
36  e_CWORD33_StatusFileLoadSuccess     = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError       = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError         = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate           = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty           = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF          = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN         = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR          = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr      = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt       = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPRegisterNotification, NPRegisterNotification

E_CWORD33_Status NPRegisterPersistentFile (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pTag*, BOOL *bIsUserFile*)

Brief

API to send message to Notification Service to add a file path as to be persisted.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*pTag*	PCSTR - A unique identifier for the file path specified, this is used in retrieve/Load the file later.

in	*bIsUserFile*	BOOL - Set TRUE, if registering user file.

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
typedef enum e_CWORD33_Status {
 e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
 e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
 e_CWORD33_StatusDbRecNotFound       = ***, ///< Database record not found
 e_CWORD33_StatusDbResultError       = ***, ///< Database result error
 e_CWORD33_StatusDbExecuteFail       = ***, ///< Database execute fail
 e_CWORD33_StatusSemCloseFail        = ***, ///< Semaphore close failed
 e_CWORD33_StatusSemUnlockFail       = ***, ///< Semaphore unlock failed
 e_CWORD33_StatusSemLockFail         = ***, ///< Semaphore lock failed
 e_CWORD33_StatusFail                = ***, ///< Failed
 e_CWORD33_StatusErrOther            = ***, ///< 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;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

_CWORD33_NPRegisterPersistentFile

E_CWORD33_Status NPRegisterPersistentFolder (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pTag*, BOOL *bIsUserFolder*)

Brief

API to send message to Notification Service to add a folder path as to be persisted.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*pTag*	PCSTR - A unique identifier for the folder path specified, this is used in retrieve/Load the folder later.
in	*bIsUserFolder*	BOOL - Set TRUE, if registering user folder.

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
```

```
16  e_CWORD33_StatusInvldQName       = ***,  ///< Invalid messasge queue name
17  e_CWORD33_StatusMsgQFull         = ***,  ///< Message queue full
18  e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19  e_CWORD33_StatusInvldParam       = ***,  ///< Invalid parameter
20  e_CWORD33_StatusInvldBufSize     = ***,  ///< Buf size too small
21  e_CWORD33_StatusInvldID          = ***,  ///< Unrecognized ID
22  e_CWORD33_StatusCannotRelease    = ***, ///< Cannot release resource
23  e_CWORD33_StatusBadConnection    = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit         = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented   = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy       = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin   = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist   = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault            = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound     = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse      = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing       = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer      = ***,
35  e_CWORD33_StatusMsgNotProcessed    = ***,
36  e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError      = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate        = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound   = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPRegisterPersistentFolder

E_CWORD33_Status NPReleasePersistentFile (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, E_CWORD33_ReleaseType *eReleaseType*, PCSTR *pTag*, PCSTR *pFullFilePath*, HANDLE *hUser*)

Brief

API to send message to Notification Service to notify that the file can be persisted.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*eReleaseType*	E_CWORD33_ReleaseType - e_CWORD33_NotOnRelease:not on release e_CWORD33_PersistOnShutdown:persist on shutdown e_CWORD33_PersistInstantly:persist instantly
in	*pTag*	PCSTR - Tag associated with the file released
in	*pFullFilePath*	PCSTR - Full path name of the file to be persisted
in	*hUser*	HANDLE - Release the file for the specified user.

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1  typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize      = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease     = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection     = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented    = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy        = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin    = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
```

```
29  e_CWORD33_StatusThreadNotExist      = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault               = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound        = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse         = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing          = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer         = ***,
35  e_CWORD33_StatusMsgNotProcessed     = ***,
36  e_CWORD33_StatusFileLoadSuccess     = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError       = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError         = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate           = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty           = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF          = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN         = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR          = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr      = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt       = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPReleasePersistentFile

E_CWORD33_Status NPReleasePersistentFolder (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, E_CWORD33_ReleaseType *e_CWORD33_ReleaseType*, PCSTR *pTag*, PCSTR *pFullFolderPath*, HANDLE *hUser*)

Brief

API to send message to Notification Service to notify that the folder can be persisted.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to the Framework
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*e_CWORD33_ReleaseType*	E_CWORD33_ReleaseType - e_CWORD33_NotOnRelease :not on release e_CWORD33_PersistOnShutdown:persist on shutdown e_CWORD33_PersistInstantly :persist instantly
in	*pTag*	PCSTR - Tag associated with the folder released
in	*pFullFolderPath*	PCSTR - Full path name of the folder to be persisted

in	*hUser*	HANDLE - Name of new Personality

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
typedef enum e_CWORD33_Status {
 e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
 e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
 e_CWORD33_StatusDbRecNotFound       = ***, ///< Database record not found
 e_CWORD33_StatusDbResultError       = ***, ///< Database result error
 e_CWORD33_StatusDbExecuteFail       = ***, ///< Database execute fail
 e_CWORD33_StatusSemCloseFail        = ***, ///< Semaphore close failed
 e_CWORD33_StatusSemUnlockFail       = ***, ///< Semaphore unlock failed
 e_CWORD33_StatusSemLockFail         = ***, ///< Semaphore lock failed
 e_CWORD33_StatusFail                = ***, ///< Failed
 e_CWORD33_StatusErrOther            = ***, ///< 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;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPReleasePersistentFolder

E_CWORD33_Status NPSavePersistentData (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*)

Brief

API to send message to Notification Service to save all persisted data that in Ram to the file system via a file write.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1  typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail             = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK               = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize      = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease     = ***, ///< Cannot release resource
```

```
23  e_CWORD33_StatusBadConnection       = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented      = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy          = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin      = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal    = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist      = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault               = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound        = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse         = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing          = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer         = ***,
35  e_CWORD33_StatusMsgNotProcessed     = ***,
36  e_CWORD33_StatusFileLoadSuccess     = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError       = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError         = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate           = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty           = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF          = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN         = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR          = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr      = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt       = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

_CWORD33_NPSavePersistentData

E_CWORD33_Status NPSetFilePersistentType (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pTag*, E_CWORD33_PersistCategory *ePersistCategory*)

Brief

API to send message to Notification Service to add a file path as to be persisted.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*pTag*	PCSTR - A unique identifier for the file path specified, this is used in retrieve/Load the file later.

in	*ePersistCategory*	E_CWORD33_PersistCategory - persist category

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
typedef enum e_CWORD33_Status {
 e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
 e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
 e_CWORD33_StatusDbRecNotFound       = ***, ///< Database record not found
 e_CWORD33_StatusDbResultError       = ***, ///< Database result error
 e_CWORD33_StatusDbExecuteFail       = ***, ///< Database execute fail
 e_CWORD33_StatusSemCloseFail        = ***, ///< Semaphore close failed
 e_CWORD33_StatusSemUnlockFail       = ***, ///< Semaphore unlock failed
 e_CWORD33_StatusSemLockFail         = ***, ///< Semaphore lock failed
 e_CWORD33_StatusFail                = ***, ///< Failed
 e_CWORD33_StatusErrOther            = ***, ///< 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;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPSetFilePersistentType

E_CWORD33_Status NPSetFolderPersistentType (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pTag*, E_CWORD33_PersistCategory *ePersistCategory*)

Brief

API to send message to Notification Service to set the persist category of file.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*pTag*	PCSTR - A unique identifier for the file path specified, this is used in retrieve/Load the file later
in	*ePersistCategory*	E_CWORD33_PersistCategory - persist category

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
```

```
19  e_CWORD33_StatusInvldParam       = ***,  ///< Invalid parameter
20  e_CWORD33_StatusInvldBufSize      = ***,  ///< Buf size too small
21  e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22  e_CWORD33_StatusCannotRelease     = ***, ///< Cannot release resource
23  e_CWORD33_StatusBadConnection     = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented    = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy        = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin    = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal  = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist    = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault             = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound      = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse       = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing        = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer       = ***,
35  e_CWORD33_StatusMsgNotProcessed   = ***,
36  e_CWORD33_StatusFileLoadSuccess   = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError     = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError       = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate         = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty         = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF        = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN       = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR        = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr    = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt     = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

_CWORD33_NPSetFolderPersistentType

E_CWORD33_Status NPSetPersistentNotfnType (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pNotification*, E_CWORD33_PersistCategory *ePersistCategory*)

Brief

API to send message to Notification Service to set the persist category of notification.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.

in	*pPublisherName*	PCSTR - Service Name
in	*pNotification*	PCSTR - Name of Notification
in	*ePersistCategory*	E_CWORD33_PersistCategory - Persistent category

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize      = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease     = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection     = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented    = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy        = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin    = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal  = ***, ///< Invalid value passed
29   e_CWORD33_StatusThreadNotExist    = ***,  ///< The thread does not exist
30   e_CWORD33_StatusFault             = ***,  ///< A fault occurred while attempting to make call
31   e_CWORD33_StatusServNotFound      = ***,  ///< Service not present in serv dir
32   e_CWORD33_StatusServerInUse       = ***,  ///< Service already processing 1 client request
33   e_CWORD33_StatusDbIndexing        = ***,  ///< Database Indexing in progress
34   e_CWORD33_StatusNullPointer       = ***,
35   e_CWORD33_StatusMsgNotProcessed   = ***,
36   e_CWORD33_StatusFileLoadSuccess   = ***, ///< File Load Success
37   e_CWORD33_StatusFileLoadError     = ***, ///< File Load Error
38   e_CWORD33_StatusAccessError       = ***, ///< Error when accessing resource
39   e_CWORD33_StatusDuplicate         = ***, ///< Duplicate entry
40   e_CWORD33_StatusMsgQEmpty         = ***, ///< Message queue empty
41   e_CWORD33_StatusThreadAlreadyRunning  = ***,
42   e_CWORD33_StatusErrNoEBADF        = ***,  ///< Bad file descriptor
43   e_CWORD33_StatusErrNoEAGAIN       = ***,  ///< Resource unavailable, try again
44   e_CWORD33_StatusErrNoEINTR        = ***,  ///< Interrupted system call
45   e_CWORD33_StatusSessionErr    = ***,  ///< Error in session handling
46   e_CWORD33_StatusDBCorrupt     = ***,  ///< Database corrupt
47   e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

Classification

Public

Type

Sync only

See also:

CWORD33_NPSetPersistentNotfnType

E_CWORD33_Status NPSetPersistNotfnDefaultValue (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pNotification*, PCVOID *pData*, const UI_32 *iLength*)

Brief

API to send message to Notification Service to set default value of persistent category in notification.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Service Name
in	*pNotification*	PCSTR - Name of Notification
in	*pData*	PVOID - Data buffer
in	*iLength*	const UI_32 - Size of data buffer

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
```

```
17  e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18  e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19  e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20  e_CWORD33_StatusInvldBufSize       = ***,  ///< Buf size too small
21  e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22  e_CWORD33_StatusCannotRelease      = ***, ///< Cannot release resource
23  e_CWORD33_StatusBadConnection      = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented     = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy         = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin     = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist     = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault             = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing       = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer      = ***,
35  e_CWORD33_StatusMsgNotProcessed    = ***,
36  e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty           = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF          = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN         = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR          = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPSetPersistNotfnDefaultValue

E_CWORD33_Status NPSetPersonality (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pUserName*)

Brief

API to send message to Notification Service to set the Personality.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of publisher message queue.
in	*pUserName*	PCSTR - Name of new Personality

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1  typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound       = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError       = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail       = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail        = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail       = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail         = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail               = ***, ///< Failed
11   e_CWORD33_StatusErrOther           = ***, ///< Unknown error
12   e_CWORD78_StatusOK                 = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf           = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle        = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType      = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName         = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull           = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam         = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize       = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID            = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease      = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection      = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit           = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented     = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy         = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin     = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29   e_CWORD33_StatusThreadNotExist     = ***,  ///< The thread does not exist
30   e_CWORD33_StatusFault              = ***,  ///< A fault occurred while attempting to make call
31   e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32   e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33   e_CWORD33_StatusDbIndexing         = ***,  ///< Database Indexing in progress
34   e_CWORD33_StatusNullPointer        = ***,
35   e_CWORD33_StatusMsgNotProcessed    = ***,
36   e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37   e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38   e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39   e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40   e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41   e_CWORD33_StatusThreadAlreadyRunning  = ***,
42   e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43   e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44   e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
45   e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46   e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
```

```
47   e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPSetPersonality

E_CWORD33_Status NPSubscribeToNotification (HANDLE *hNPMsgQ*, PCSTR *pSubscriberName*, PCSTR *notif_name*)

Brief

API to send message to Notification Service to add to subscription list for.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pSubscriberName*	PCSTR - Name of subscriber message queue
in	*notif_name*	PCSTR - Name of Notification

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
```

```
18  e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19  e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20  e_CWORD33_StatusInvldBufSize       = ***,  ///< Buf size too small
21  e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22  e_CWORD33_StatusCannotRelease      = ***, ///< Cannot release resource
23  e_CWORD33_StatusBadConnection      = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit           = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented     = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy         = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin     = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist     = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault              = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing       = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer      = ***,
35  e_CWORD33_StatusMsgNotProcessed    = ***,
36  e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty           = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF          = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN         = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR          = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

_CWORD33_NPSubscribeToNotification

E_CWORD33_Status NPSubscribeToNotifications (HANDLE *hNPMsgQ*, PCSTR *pSubscriberName*, UI_32 *numNotifications*, SubscribeInfo * *pSubscribeInfoArray*)

Brief

API to send message to Notification Service to add multiple subscriptions for a set of notification.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pSubscriberName*	PCSTR - Name of subscriber message queue
in	*numNotifications*	UI_32 - Number of Notifications
in	*pSubscribeInfoArray*	SubscribeInfo* - Array of SubscribeInfo

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
typedef enum e_CWORD33_Status {
  e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
  e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
  e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
  e_CWORD33_StatusDbResultError      = ***, ///< Database result error
  e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
  e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
  e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
  e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
  e_CWORD33_StatusFail               = ***, ///< Failed
  e_CWORD33_StatusErrOther           = ***, ///< 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
```

```
46   e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47   e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

_CWORD33_NPSubscribeToNotifications

E_CWORD33_Status NPSynchronousReadPersistentData (PCSTR *pAppName*, PCSTR *notif_name*, PVOID *pData*, UI_32 *uiDataSize*, const E_CWORD33_PersistCategory *ePersistCategory* = e_CWORD33_UserData)

Brief

API to synchronously read persistent data corresponding to a notification before NPPService is available

Parameters:

in	*pAppName*	PCSTR - Application/ thread name
in	*notif_name*	
in	*pData*	PVOID - Pointer to the buffer where read data will be written.
in	*uiDataSize*	UI_32 - Length of the data buffer.
in	*ePersistCategory*	const E_CWORD33_PersistCategory - Persist Category of Notification.

Return values:

e_CWORD33_StatusOK	
e_CWORD33_StatusInvldParam	
e_CWORD33_StatusFail	

Preconditons

none

Change of internal status

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

Classification

Public

Type

Set Get

See also:

NPSynchronousWritePersistentData

E_CWORD33_Status NPSynchronousWritePersistentData (PCSTR *pAppName*, PCSTR *notif_name*, PVOID *pData*, const UI_32 *uiDataSize*, const E_CWORD33_PersistCategory *ePersistCategory* = e_CWORD33_UserData)

Brief

API to synchronously write persistent data corresponding to a notification before NPPService is available

Parameters:

in	*pAppName*	PCSTR - Application/ thread name
in	*notif_name*	
in	*pData*	PVOID - Pointer to the buffer where read data will be written.
in	*uiDataSize*	UI_32 - Length of the data buffer.
in	*ePersistCategory*	const E_CWORD33_PersistCategory - Persist Category of Notification.

Return values:

e_CWORD33_StatusOK	
e_CWORD33_StatusInvldParam	
e_CWORD33_StatusFail	

Preconditons

none

Change of internal status

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

Classification

Public

Type

Set Get

See also:

NPSynchronousWritePersistentData

E_CWORD33_Status NPUnRegisterNotification (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, PCSTR *pNotification*)

Brief

API to send message to Notification Service to remove a notification

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of Publisher message queue
in	*pNotification*	PCSTR - Name of Notification

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1  typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound       = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError       = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail       = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail        = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail       = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail         = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail               = ***, ///< Failed
11   e_CWORD33_StatusErrOther           = ***, ///< Unknown error
12   e_CWORD78_StatusOK                 = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf           = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle        = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType      = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName         = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull           = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam         = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize       = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID            = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease      = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection      = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit           = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented     = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy         = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin     = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29   e_CWORD33_StatusThreadNotExist     = ***,  ///< The thread does not exist
30   e_CWORD33_StatusFault              = ***,  ///< A fault occurred while attempting to make call
31   e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32   e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33   e_CWORD33_StatusDbIndexing         = ***,  ///< Database Indexing in progress
34   e_CWORD33_StatusNullPointer        = ***,
35   e_CWORD33_StatusMsgNotProcessed    = ***,
36   e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37   e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38   e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39   e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40   e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41   e_CWORD33_StatusThreadAlreadyRunning  = ***,
42   e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43   e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44   e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
45   e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46   e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
```

```
47   e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPUnRegisterNotification, NPUnRegisterNotifications

E_CWORD33_Status NPUnRegisterNotifications (HANDLE *hNPMsgQ*, PCSTR *pPublisherName*, UI_32 *numNotifications*, NotificationInfo * *pNotificationArray*)

Brief

API to send message to Notification Service to remove a set of notification

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pPublisherName*	PCSTR - Name of Publisher message queue
in	*numNotifications*	UI_32 - Number of Notification
in	*pNotificationArray*	NotificationInfo - Array of notifications

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail              = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK                = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle       = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType     = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName        = ***,  ///< Invalid messasge queue name
```

```
17  e_CWORD33_StatusMsgQFull          = ***,  ///< Message queue full
18  e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19  e_CWORD33_StatusInvldParam        = ***,  ///< Invalid parameter
20  e_CWORD33_StatusInvldBufSize       = ***,  ///< Buf size too small
21  e_CWORD33_StatusInvldID           = ***,  ///< Unrecognized ID
22  e_CWORD33_StatusCannotRelease      = ***, ///< Cannot release resource
23  e_CWORD33_StatusBadConnection      = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit          = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented     = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy         = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin     = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist     = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault             = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing       = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer      = ***,
35  e_CWORD33_StatusMsgNotProcessed    = ***,
36  e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty           = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF          = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN         = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR          = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

CWORD33_NPUnRegisterNotifications

E_CWORD33_Status NPUnsubscribeFromNotification (HANDLE *hNPMsgQ*, PCSTR *pSubscriberName*, PCSTR *pNotification*)

Brief

API to send message to Notification Service to remove from subscription list for given notification.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pSubscriberName*	PCSTR - Name of subscriber message queue
in	*pNotification*	PCSTR - Name of Notification

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1  typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound       = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError       = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail       = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail        = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail       = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail         = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail               = ***, ///< Failed
11   e_CWORD33_StatusErrOther           = ***, ///< Unknown error
12   e_CWORD78_StatusOK                 = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf           = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle        = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType      = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName         = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull           = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification   = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam         = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize       = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID            = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease      = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection      = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit           = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented     = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy         = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin     = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29   e_CWORD33_StatusThreadNotExist     = ***,  ///< The thread does not exist
30   e_CWORD33_StatusFault              = ***,  ///< A fault occurred while attempting to make call
31   e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32   e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33   e_CWORD33_StatusDbIndexing         = ***,  ///< Database Indexing in progress
34   e_CWORD33_StatusNullPointer        = ***,
35   e_CWORD33_StatusMsgNotProcessed    = ***,
36   e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37   e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38   e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39   e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40   e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41   e_CWORD33_StatusThreadAlreadyRunning  = ***,
42   e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43   e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44   e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
45   e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46   e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
```

```
47   e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

_CWORD33_NPUnsubscribeFromNotification

E_CWORD33_Status NPUnsubscribeFromNotifications (HANDLE *hNPMsgQ*, PCSTR *pUnSubscriberName*, UI_32 *numNotifications*, SubscribeInfo * *pUnSubscribeInfoArray*)

Brief

API to send message to Notification Service to unsubscribe multiple subscriptions for a set of notification.

Parameters:

in	*hNPMsgQ*	HANDLE - Handle to message queue of Notification service.
in	*pSubscriberName*	PCSTR - Name of subscriber message queue
in	*numNotifications*	UI_32 - Number of Notifications
in	*pUnSubscribeInfoArray*	SubscribeInfo - Array of notifications

Return values:

E_CWORD33_Status	indicates success or error
Success(e_CWORD33_StatusOK)	
Error(anything	else)

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
5   e_CWORD33_StatusDbResultError      = ***, ///< Database result error
6   e_CWORD33_StatusDbExecuteFail      = ***, ///< Database execute fail
7   e_CWORD33_StatusSemCloseFail       = ***, ///< Semaphore close failed
8   e_CWORD33_StatusSemUnlockFail      = ***, ///< Semaphore unlock failed
9   e_CWORD33_StatusSemLockFail        = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail             = ***, ///< Failed
11   e_CWORD33_StatusErrOther          = ***, ///< Unknown error
12   e_CWORD78_StatusOK               = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf          = ***,  ///< Invalid buffer
```

```
14  e_CWORD33_StatusInvldHandle        = ***,  ///< Invalid handle
15  e_CWORD33_StatusInvldHndlType      = ***,  ///< Invalid handle type
16  e_CWORD33_StatusInvldQName         = ***,  ///< Invalid messasge queue name
17  e_CWORD33_StatusMsgQFull           = ***,  ///< Message queue full
18  e_CWORD33_StatusInvldNotification  = ***,///< The Notification event not present
19  e_CWORD33_StatusInvldParam         = ***,  ///< Invalid parameter
20  e_CWORD33_StatusInvldBufSize       = ***,  ///< Buf size too small
21  e_CWORD33_StatusInvldID            = ***,  ///< Unrecognized ID
22  e_CWORD33_StatusCannotRelease      = ***, ///< Cannot release resource
23  e_CWORD33_StatusBadConnection      = ***, ///< Could not locate resource
24  e_CWORD33_StatusExit           = ***,  ///< Normal application termination
25  e_CWORD33_StatusNotImplemented     = ***,  ///< incomplete feature
26  e_CWORD33_StatusThreadBusy         = ***,  ///< Joined thread is already being joined
27  e_CWORD33_StatusThreadSelfJoin     = ***,  ///< Thread is joining itself
28  e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29  e_CWORD33_StatusThreadNotExist     = ***,  ///< The thread does not exist
30  e_CWORD33_StatusFault              = ***,  ///< A fault occurred while attempting to make call
31  e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32  e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33  e_CWORD33_StatusDbIndexing         = ***,  ///< Database Indexing in progress
34  e_CWORD33_StatusNullPointer        = ***,
35  e_CWORD33_StatusMsgNotProcessed    = ***,
36  e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37  e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38  e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39  e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40  e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41  e_CWORD33_StatusThreadAlreadyRunning  = ***,
42  e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43  e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44  e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
45  e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46  e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47  e_CWORD33_StatusDBFileNotFound   = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

_CWORD33_NPUnsubscribeFromNotifications

Native

class CNSConfigReader
Configuration File Read Service Class. class CNSConfigWriter
File Write Service Class. class CNSConfigParser
File both Reade and Write Service Class. struct _stLogEvent_
struct _stLogCount_
struct __CWORD33_LOGPARAM
_CWORD33_LOG parameter *More...*
struct tag_Change_Log_Parameters
struct _TNS_EnableRcsLogging
struct __CWORD33_LOGPPARAM
Performance LOG parameter. *More...*
struct __CWORD33_LOGSYSEVENTPARAM
System event LOG parameter. *More...*
struct _NSRingBufferHdr
struct _NSRingBufferMtx
class CNSRingBuffer
CNSRingBuffer. struct SMsgHeader
struct SReplyHeader
struct _NSSharedBuffer
class CNSSharedMem
this file has the C_CWORD33_Guard class definitions class CNSSharedMemReader
read shared memory class CNSSharedMemWriter
write shared memory struct _NSTimerInfo
Timer info, defines the initial start of a timer, the repeat timer values and the cmd id for a timer. *More...*
class NSTimer
Handle Timer. struct Int2Type< v >
struct Type2Type< T >
struct TList< T, N >
struct TList< T, 1 >
struct FSig< Sig >
struct FSig< R(*)() >
struct FSig< R(*)(T1) >
struct FSig< R(*)(T1, T2) >
struct FSig< R(*)(T1, T2, T3) >
struct FSig< R(*)(T1, T2, T3, T4) >
struct FSig< R(*)(T1, T2, T3, T4, T5) >
struct FSig< R(*)(T1, T2, T3, T4, T5, T6) >
struct FSig< R(*)(T1, T2, T3, T4, T5, T6, T7) >

class Accumulator< T >

Accumultor type. class MemTraits< T >

class RaiseExceptionPolicy< RsrcTraits >
class CheckForErrorPolicy< RsrcTraits >
class ResourceMgr< T, RsrcTraits, ErrorPolicy >
class IFunctor< R >
class CFunctor0< TFn >
class CFunctor1< TFn, Arg1 >
class CFunctor2< TFn, Arg1, Arg2 >
class CFunctor3< TFn, Arg1, Arg2, Arg3 >
class CFunctor4< TFn, Arg1, Arg2, Arg3, Arg4 >
class CFunctor5< TFn, Arg1, Arg2, Arg3, Arg4, Arg5 >
class CFunctor6< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6 >
class CFunctor7< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7 >
class CMutex

Mutex Lock Class. class CRWLock

Read/Write Lock Class. class C_CWORD33_Version

Version Info Class. class CTestCaseData

CTestCaseData : Class used to fill testcase data from XML. class XMLParser

Macros

#define **EVENTLOG_MSGQ_NAME** "/EvntLogQue"
#define **_CWORD33_LOG_MSGQ_NAME** "/NSLog"
#define **_CWORD33_LOG_SHAREDMEM_NAME** "/_CWORD33__debug.log.1"
#define **_CWORD33_LOG_SHAREDMEM_SIZE** 5242880
#define **MASTER_ZONE_COUNT** 16U
#define **BITS_IN_BYTE** (8U)
#define **BITS_IN_TZONE** ((UI_32)(sizeof(T_CWORD33_Zone) * BITS_IN_BYTE))
#define **TZONE_COUNT** ((UI_32)MASTER_ZONE_COUNT)
#define **BITS_IN_ZONE_MASK** ((UI_32)(BITS_IN_TZONE * TZONE_COUNT))
#define **DEBUG_ZONE_COUNT** BITS_IN_ZONE_MASK
#define **ZONE_MASK_ARRAY_ELTS** ((UI_32)TZONE_COUNT)
#define **USER_ZONES_COUNT** BITS_IN_ZONE_MASK -10
#define **ZONE_TEXT_SIZE** 24
#define **NS_RCS_LOGGER_PLUGIN_Q** "/NsRcsLoggerPlugin"
#define _CWORD33_LOGZONES {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

extern _CWORD33_LOG parameter

#define _CWORD33__SET_ZONES()
#define **ZONE_END** UINT_MAX
#define **GET_ZONE_INDEX**(zone) (((UI_32)zone) >> 5)
#define **GET_ZONE_BIT_MASK**(zone) ((T_CWORD33_Zone)(1U << (((UI_32)zone) & 0x1F)))
#define ZONEMASK(n) ((T_CWORD33_Zone)(n))
#define IS_ZONE_SET(set_zone) (NsLogIsZoneSet(set_zone) == TRUE)
#define _CWORD33_LOG(zone, funcname, print_fmt, args...) (zone != ZONEMASK(11))?((IS_ZONE_SET(zone))?TEXT_ZONE(zone, funcname, print_fmt, ## args):((void)0)):((void)(0))

#define **_CWORD33_LOG_TIME**(zone, str) ((IS_ZONE_SET(zone))?NsLogTime(zone, str):((void)(0)))

#define **_CWORD33_LOG_DATA**(zone, data, size) ((IS_ZONE_SET(zone))?NsLogData(zone, data, size):((void)(0)))

#define _CWORD33_LOG_EVTCNT(zone, CntId, EvtId, n...) NsLog_EvtCnt(CntId, EvtId, ##n)

_CWORD33_LOG_EVTCNT

#define _CWORD33_LOG_EVT(zone, EvtId, n...) NsLog_Evt(EvtId, ##n)

_CWORD33_LOG_EVT

#define _CWORD33_LOG_CNT(zone, CntId, n...) NsLog_Cnt(CntId, ##n)

_CWORD33_LOG_CNT

#define TEXT(funcname, args...) TEXT_ZONE(BITS_IN_ZONE_MASK, funcname, ## args)

Deprecated API. Not use.

#define **GET_MACRO**(_1, _2, _3, _4, _5, _6, _7, _8, _9, _10, _11, _12, _13, _14, _15, _16, _17, _18, _19, _20, _21, _22, _23, _24, _25, _26, _27, _28, _29, _30, _31, _32, _33, _34, _35, _36, _37, _38, _39, _40, _41, _42, _43, _44, _45, _46, _47, _48, _49, _50, _51, _52, _53, _54, _55, _56, _57, _58, _59, _60, _61, _62, _63, _64, _65, _66, _67, _68, _69, _70, _71, _72, _73, _74, _75, _76, _77, _78, NAME, ...) NAME

#define **NsLogFunc**(...)

#define **TEXT_ZONE**(zone_index, funcname, ...) NsLogFunc(__VA_ARGS__)(__LINE__, zone_index, funcname, __VA_ARGS__)

#define TEXT_EVTCNT(CntId, EvtId, n...) NsLog_EvtCnt(CntId, EvtId, ##n)

Deprecated API. Not use.

#define TEXT_EVT(EvtID, n...) NsLog_Evt(EvtID, ##n)

Deprecated API. Not use.

#define TEXT_CNT(CntId, n...) NsLog_Cnt(CntId, ##n)

Deprecated API. Not use.

#define _CWORD33_LOG0 _CWORD33_LOG

Deprecated API. Use _CWORD33_LOG instead.

#define ZONE_NS_FUNC ZONEMASK(0)

Note: These bits are reserved for Framework logging and debugging.

#define **ZONE_NS_DIS** ZONEMASK(1)

#define **ZONE_NS__CWORD77_** ZONEMASK(2)

#define **ZONE_NS_IMP_INFO** ZONEMASK(3)

#define **ZONE_NS_SM_DEV_INFO** ZONEMASK(4)

#define **ZONE_NS_SM_USR_INFO** ZONEMASK(5)

#define **ZONE_NS_INFO** ZONEMASK(6)

#define **ZONE_NS_DEBUG_DUMP** ZONEMASK(7)

#define **ZONE_NS_WAR** ZONEMASK(8)

#define **ZONE_NS_ERR** ZONEMASK(9)

#define **ZONE_DEBUG_DUMP** ZONE_NS_DEBUG_DUMP

#define **ZONE_TEXT_0** "NS_Func"

#define **ZONE_TEXT_1** "NS_Dispatcher"

#define **ZONE_TEXT_2** "NS__CWORD77_"

#define **ZONE_TEXT_3** "NS_Reserved"

#define **ZONE_TEXT_4** "NS_StateMachine_Dev"
#define **ZONE_TEXT_5** "NS_StateMachine_Usr"
#define **ZONE_TEXT_6** "NS_Info"
#define **ZONE_TEXT_7** "NS_Debug_Dump"
#define **ZONE_TEXT_8** "NS_Warning"
#define **ZONE_TEXT_9** "NS_Error"
#define ZONE_PERFORMANCE ZONEMASK(64)

Note: These bits are reserved for special ZONE.

#define **ZONE_SCREEN_TRANS** ZONEMASK(65)
#define **ZONE_SWDL** ZONEMASK(66)
#define **ZONE_SYSTEMDATA** ZONEMASK(67)
#define **ZONE_LOG_SH** ZONEMASK(68)
#define **ZONE_LOG_SYS** ZONEMASK(69)
#define **ZONE_LOG__CWORD52_** ZONEMASK(70)
#define **ZONE_SECURE** ZONEMASK(71)
#define **ZONE_CAN_FILTER** ZONEMASK(72)
#define **ZONE_COMM_PROCESS** ZONEMASK(73)
#define **ZONE_CONNECT_DEVICE** ZONEMASK(74)
#define **ZONE_SP_ZONE_75** ZONEMASK(75)
#define **ZONE_SP_ZONE_76** ZONEMASK(76)
#define **ZONE_SP_ZONE_77** ZONEMASK(77)
#define **ZONE_SP_ZONE_78** ZONEMASK(78)
#define **ZONE_SP_ZONE_79** ZONEMASK(79)
#define **ZONE_SP_ZONE_80** ZONEMASK(80)
#define **ZONE_SP_ZONE_81** ZONEMASK(81)
#define **ZONE_SP_ZONE_82** ZONEMASK(82)
#define **ZONE_SP_ZONE_83** ZONEMASK(83)
#define **ZONE_SP_ZONE_84** ZONEMASK(84)
#define **ZONE_SP_ZONE_85** ZONEMASK(85)
#define **ZONE_SP_ZONE_86** ZONEMASK(86)
#define **ZONE_SP_ZONE_87** ZONEMASK(87)
#define **ZONE_SP_ZONE_88** ZONEMASK(88)
#define **ZONE_SP_ZONE_89** ZONEMASK(89)
#define **ZONE_SP_ZONE_90** ZONEMASK(90)
#define **ZONE_SP_ZONE_91** ZONEMASK(91)
#define **ZONE_SP_ZONE_92** ZONEMASK(92)
#define **ZONE_SP_ZONE_93** ZONEMASK(93)
#define **ZONE_SP_ZONE_94** ZONEMASK(94)
#define **ZONE_SP_ZONE_95** ZONEMASK(95)
#define ZONE_COMM_SYS ZONEMASK(96)

Note: These bits are reserved for special ZONE.

#define **ZONE_COMM_SH** ZONEMASK(97)
#define **ZONE_COMM_USB** ZONEMASK(98)
#define **ZONE_SP_ZONE_99** ZONEMASK(99)
#define **ZONE_SP_ZONE_100** ZONEMASK(100)
#define **ZONE_SP_ZONE_101** ZONEMASK(101)
#define **ZONE_SP_ZONE_102** ZONEMASK(102)

#define **ZONE_SP_ZONE_103** ZONEMASK(103)
#define **ZONE_SP_ZONE_104** ZONEMASK(104)
#define **ZONE_SP_ZONE_105** ZONEMASK(105)
#define **ZONE_SP_ZONE_106** ZONEMASK(106)
#define **ZONE_SP_ZONE_107** ZONEMASK(107)
#define **ZONE_SP_ZONE_108** ZONEMASK(108)
#define **ZONE_SP_ZONE_109** ZONEMASK(109)
#define **ZONE_SP_ZONE_110** ZONEMASK(110)
#define **ZONE_SP_ZONE_111** ZONEMASK(111)
#define **ZONE_SP_ZONE_112** ZONEMASK(112)
#define **ZONE_SP_ZONE_113** ZONEMASK(113)
#define **ZONE_SP_ZONE_114** ZONEMASK(114)
#define **ZONE_SP_ZONE_115** ZONEMASK(115)
#define **ZONE_SP_ZONE_116** ZONEMASK(116)
#define **ZONE_SP_ZONE_117** ZONEMASK(117)
#define **ZONE_SP_ZONE_118** ZONEMASK(118)
#define **ZONE_SP_ZONE_119** ZONEMASK(119)
#define **ZONE_SP_ZONE_120** ZONEMASK(120)
#define **ZONE_SP_ZONE_121** ZONEMASK(121)
#define **ZONE_SP_ZONE_122** ZONEMASK(122)
#define **ZONE_SP_ZONE_123** ZONEMASK(123)
#define **ZONE_SP_ZONE_124** ZONEMASK(124)
#define **ZONE_SP_ZONE_125** ZONEMASK(125)
#define **ZONE_SP_ZONE_126** ZONEMASK(126)
#define **ZONE_SP_ZONE_127** ZONEMASK(127)
#define **_CWORD33__RES_ABNMLMON** "_CWORD33__RES_ABNMLMON"
#define **_CWORD33__RES_TIMER** "_CWORD33__RES_TIMER"
#define **BAD_MEM_ID** (0)
#define **TIMER_QUE** "TIMER"
#define **MAX_SERVICE_NAME** 15
#define NS_INVALID_RETURN -1
#define NS_SETBIT(x, i) ((x) |= (1 << (i)))
#define NS_CLEARBIT(x, i) ((x) &= ~(1 << (i)))
#define NS_TOGGLEBIT(x, i) ((x) ^= (1 << (i)))
#define NS_ISBITSET(x, i) (((x) & (1 << (i))) != ***)
#define _CWORD33__GET_VERSION() "undefined_undefined_00.00.00"
#define _CWORD33__APP_VERSION_MAJOR() (0)
#define _CWORD33__APP_VERSION_MINOR() (0)
#define _CWORD33__APP_VERSION_REVISION() (0)
#define _CWORD33__APP_VERSION_BUILDVER() "undefined"
#define _CWORD33__APP_VERSION_PRODUCTID() "undefined"

Typedefs

typedef enum _Datatype_ **Datatype**
typedef enum _SystemPhase_ **SystemPhase**
typedef enum _NStoSS_LOGGERSERVICEPROTOCOL **NStoSS_loggerserviceprotocol**
typedef struct _stLogEvent_ **st_LogEvent**
typedef struct _stLogCount_ **st_LogCount**

typedef UI_32 T_CWORD33_Zone

typedef of ZONE

typedef T_CWORD33_Zone T_CWORD33_ZoneMask[ZONE_MASK_ARRAY_ELTS]

Array of mask options.

typedef enum _T_CWORD33_LoggerSeverity **T_CWORD33_LoggerSeverity**
typedef struct __CWORD33_LOGPARAM _CWORD33_LOGPARAM

_CWORD33_LOG parameter

typedef struct __CWORD33_LOGPARAM * **LP_CWORD33_LOGPARAM**
typedef enum _NS_LOGGER_METHODS NS_LoggingMethod_t

_CWORD33_LOG output direction

typedef struct tag_Change_Log_Parameters **CHANGELOGPARAMS**
typedef enum _NS_RCS_LOGGER_SETTINGS **NS_RCS_LOGGER_SETTINGS**
typedef struct _TNS_EnableRcsLogging **TNS_EnableRcsLogging**
typedef unsigned int TMemID

< Standard CWORD3 Types

typedef void(* TimerCb) (UI_16 cmd)

Timer Function Pointer definition Detailed description of the class.

typedef struct _NSTimerInfo NSTimerInfo

Timer info, defines the initial start of a timer, the repeat timer values and the cmd id for a timer.

typedef enum _NSTimerCallbackMechanism eNSTimerCallbackMechanism

Enum Types for valid Callback Mechanisms for a NS_Timer.

Enumerations

enum **_Datatype_** { **COMMON**, **EVENT_SPECIFIC** }
enum **_SystemPhase_** { **STARTUP** = ***, **NORMAL** = ***, **SHUTDOWN** = *** }
enum **_NStoSS_LOGGERSERVICEPROTOCOL** { **SS_MSG_EVTLOG** = ***, **SS_MSG_LOGGERCNT**, **SS_MSG_LOGGER_CNT_EVTLOG** }
enum _T_CWORD33_LoggerSeverity { **_CWORD33__SEVERITY_LOW** = ***, _CWORD33__SEVERITY_DEBUG2, _CWORD33__SEVERITY_DEBUG1, _CWORD33__SEVERITY_INFO, _CWORD33__SEVERITY_WARN, _CWORD33__SEVERITY_ERROR, _CWORD33__SEVERITY_FATAL, _CWORD33__SEVERITY_ALWAYS }

enum _NS_LOGGER_METHODS { **LPRINT** = ***, LMSGQ = ***, LSLOGGER = ***, LSHAREDMEM = *** }*_CWORD33_LOG output direction*

enum **_NS_RCS_LOGGER_SETTINGS** { **NS_RCS_LOGGER_PLUGIN_ADD_APPNAME** = *** }

enum _NSTimerCallbackMechanism { **CALLBACK_MESSAGE** }*Enum Types for valid Callback Mechanisms for a NS_Timer.*

Functions

int _CWORD33_GetResource (const char *mod, const char *key, long *resource)
int _CWORD33_AcquireResouce (const char *mod, const char *key, long *resource)
int _CWORD33_ReleaseResouce (const char *mod, const char *key)
int _CWORD33_SearchResourseKey (const char *mod, long resource, const char **key)
int _CWORD33_RegistResouce (const char *mod, const char *key, long resource, int init_counter)

int _CWORD33_UnregistResouce (const char *mod, const char *key)
void NsLog (const UI_16 p_lLine_i, const UI_16 f_uiZoneIndex, PCSTR p_pstrClassName_i, PCSTR lpszFormat,...)
void NsLog0 (const UI_16 p_lLine_i, const UI_16 f_uiZoneIndex, PCSTR p_pstrClassName_i, PCSTR lpszFormat)
void NsLogTime (const UI_16 f_uiZoneIndex, PCSTR lpszFormat)
void NsLogData (const UI_16 f_uiZoneIndex, PCSTR data, UI_32 size)
void NsLogSet_CWORD33_LogParams (_CWORD33_LOGPARAM *p__CWORD33_LogParams)
void NsLogSetProcessName (PCSTR p_strProcessName_i)
void NsLogSetControlMask (T_CWORD33_ZoneMask p_ulNSLogControl_i)
BOOL NsLogIsZoneSet (UI_32 set_zone)
void NsLogGetControlMask (T_CWORD33_ZoneMask p_Zonemask_i)
void NsLogSetLogMethod (UI_8 p_eMethod_i)
UI_8 NsLogGetLogMethod (void)
void NsLogInitialize (void)
void NsLogSet_CWORD33_LogFlag (UI_8 flag_id, UI_8 mode)
E_CWORD33_Status NsLogGet_CWORD33_LogFlag (UI_8 flag_id, UI_8 *mode)
void NsLogSetRealtimeLog (UI_8 mode)
void NsLogGetRealtimeLog (UI_8 *mode)
void NsLogSetSeverity (T_CWORD33_LoggerSeverity p_eLogSeverity_i)
T_CWORD33_LoggerSeverity NsLogGetSeverity (void)
void NsLog_EvtCnt (UI_16 Cnt_Id, UI_16 Evt_Id, UI_8 n,...)
void NsLog_Evt (UI_16 Evt_Id, UI_8 nu,...)
void NsLog_Cnt (UI_16 Cnt_Id, UI_8 nu,...)
UI_8 NsLogDetermineLogMethod (PCSTR output_type)
VOID NsLogGetZoneTextList (CHAR f_cZoneList[][ZONE_TEXT_SIZE])
void NsLogSetZones (UI_32 f_uiZoneCount,...)
void **NsLogParseZones** (_CWORD33_LOGPARAM *p__CWORD33_LogParams, UI_32 f_uiZoneCount,...)
UI_32 NSLogGet_CWORD33_logFileTotalNum (void)
PCSTR NSLogGet_CWORD33_logFileName (UI_32 index)
int NSLogGet_CWORD33_logIndex (PCSTR filename)
VOID NsForceClose (void)
HANDLE McOpenReceiver (PCSTR name)
HANDLE McOpenReceiverNotBlocked (PCSTR name)
HANDLE McOpenSyncReceiver (PCSTR name)
HANDLE McOpenSender (PCSTR name)
HANDLE McOpenSenderNotBlocked (PCSTR name)
HANDLE McOpenSyncSender (PCSTR name)
HANDLE **McOpenSenderChild** (PCSTR name, pthread_t childid)
E_CWORD33_Status **McJoinChild** (HANDLE hChildApp)
E_CWORD33_Status **McGetChildThreadPriority** (HANDLE hChildApp, PSI_32 threadPrio)
E_CWORD33_Status McReceive (HANDLE hMessage, PSTR source, UI_32 *cmd, UI_32 length, PVOID data)
E_CWORD33_Status McReceiveWithSession (HANDLE hMessage, PSTR source, UI_32 *cmd, UI_32 *sessionid, UI_32 length, PVOID data)
UI_32 McGetLength (PVOID data)
PVOID McGetDataPointer (PVOID data)

E_CWORD33_Status McGetDataOfSize (PVOID data, PVOID to, UI_32 uiSize)
E_CWORD33_Status McGetDataOfSizeWithSMRetain (PVOID data, PVOID to, UI_32 uiSize)
E_CWORD33_Status McGetSysInfoData (PVOID data, PVOID to)
E_CWORD33_Status McClearData (PVOID data)
E_CWORD33_Status McSend (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 length, PCVOID data)
E_CWORD33_Status McSendWithSession (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 length, PCVOID data, UI_32 sessionid)
E_CWORD33_Status McSendWithPriority (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 length, PCVOID data, E_CWORD33_MessagePriorties priority, UI_32 sessionid)
E_CWORD33_Status McInvokeSync (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 msgLength, PCVOID msgData, UI_32 sessionid, HANDLE hRcvMessage, UI_32 responseLength, PVOID responseData, UI_32 *receivedLength)
E_CWORD33_Status McSendSyncResponse (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 seq_id, E_CWORD33_Status ret_val, UI_32 length, PCVOID data)
E_CWORD33_Status McCreateInvokerName (PCSTR source, UI_32 sessionid, PSTR invokerName, UI_32 size)
E_CWORD33_Status McClose (HANDLE hMessage)
TMemID McGetDataUSID (PVOID pData)
PCSTR McGetMsgSrc (PVOID data)
E_CWORD33_Status McForward (HANDLE hMessage, PCSTR source, UI_32 iCmd, TMemID USID)
void McFlushReceiver (HANDLE hMessage)
PCSTR McGetQueueName (HANDLE hMessage)
int McGetQueueFD (HANDLE hMessage)
E_CWORD33_Status McTranslateError (int error)
E_CWORD33_Status McZcSetParam (HANDLE handle, UI_32 cmd, UI_32 length)
PVOID McZcGetBuf (HANDLE handle)
E_CWORD33_Status McZcSend (HANDLE hMessage)
HANDLE McZcOpenSender (PCSTR source)
E_CWORD33_Status McZcClose (HANDLE handle)
TMemID SetDataToShared (const void *data, UI_32 dataBytes, const char *from, const char *to)
E_CWORD33_Status GetDataFromShared (TMemID id, void *data, UI_32 dataMaxBytes)
UI_32 GetLengthOfDataFromShared (TMemID id)
E_CWORD33_Status DiscardDataFromShared (TMemID id)
time_t WholeSeconds (UI_32 ms)

Helper methods that convert time provided in MS. mseconds.

UI_32 RemainderMs (UI_32 ms)
UI_64 MSToNS (UI_32 ms)
HANDLE NS_TimerCreate (NSTimerInfo timer_info, eNSTimerCallbackMechanism cbMech, HANDLE sndMqHndl)
E_CWORD33_Status NS_TimerDelete (HANDLE hTimer)
E_CWORD33_Status NS_TimerSetTime (HANDLE hTimer, NSTimerInfo timer_info)
E_CWORD33_Status NS_TimerGetTime (HANDLE hTimer, NSTimerInfo *timer_info)
void NS_TimerDebugOn (BOOL FlagState)
DEFINE_EXCEPTION (lock_error, std::runtime_error)
DEFINE_EXCEPTION (lock_creation_error, lock_error)
DEFINE_EXCEPTION (lock_acquireread_error, lock_error)
DEFINE_EXCEPTION (lock_acquirewrite_error, lock_error)

DEFINE_EXCEPTION (lock_release_error, lock_error)

enum _EPLOG_TIME_FORMAT { EPLOG_TIME_FORMAT_USEC, EPLOG_TIME_FORMAT_MSEC, EPLOG_TIME_FORMAT_SEC }*Performance logging.*

enum _EPLOG_OUTPUT_OPTION { EPLOG_SLOG = ***, EPLOG_MSGQ = ***, EPLOG_TRACEEVENT = *** }

typedef enum _EPLOG_TIME_FORMAT EPLOG_TIME_FORMAT

Performance logging.

typedef enum _EPLOG_OUTPUT_OPTION **EPLOG_OUTPUT_OPTION**

typedef struct __CWORD33_LOGPPARAM _CWORD33_LOGPPARAM

Performance LOG parameter.

typedef struct __CWORD33_LOGPPARAM * **L_CWORD33_LOGPPARAM**

typedef struct __CWORD33_LOGSYSEVENTPARAM _CWORD33_LOGSYSEVENTPARAM

System event LOG parameter.

typedef struct __CWORD33_LOGSYSEVENTPARAM * **L_CWORD33_LOGSYSEVENTPARAM**

_CWORD33_LOGPPARAM **g__CWORD33_LogPParams**

_CWORD33_LOGSYSEVENTPARAM **g__CWORD33_LogSysEventParams**

VOID NSLogPrintPerformanceLog (const UI_16 f_ui16Line, PCSTR f_cFuncName, PCSTR __format,...)

VOID NSLogEnablePLog (BOOL f_bEnable)

BOOL NsLogIsPLogEnabled (void)

VOID NSLogSetPlogTimeFormat (EPLOG_TIME_FORMAT f_ePlogTimeFormat)

VOID NSLogSetPlogOutputOptions (UI_8 f_uiPlogOutputOption)

VOID NSLogSysEvent (const UI_16 f_ui16Line, PCSTR f_cFuncName, PCSTR __format,...)

VOID NSLogEnableSysEventLog (BOOL f_bEnable)

BOOL NSLogIsSysEventLogEnabled (void)

#define **NS_PLOGQ** "/NSplog"

#define **NS_SYSEVENTLOGQ** "/NSSysEventLog"

#define **PLOG_TEXT**(args...) NSLogPrintPerformanceLog(__LINE__, __FUNCTION__, ## args)

#define **SYSEVENTLOG_TEXT**(args...) NSLogSysEvent(__LINE__, __FUNCTION__, ## args)

#define **IS_PLOG_ENABLED**() NsLogIsPLogEnabled()

#define **IS_SYSEVENTLOG_ENABLED**() NSLogIsSysEventLogEnabled()

#define _CWORD33_LOG_PERFORMANCE_DEBUG(print_fmt, args...) ((IS_PLOG_ENABLED()) ? PLOG_TEXT(print_fmt, ## args): ((void)0))

_CWORD33_LOG_PERFORMANCE_DEBUG

#define _CWORD33_LOG_PERFORMANCE(print_fmt, args...) ((IS_PLOG_ENABLED()) ? PLOG_TEXT(print_fmt, ## args): ((void)0))

_CWORD33_LOG_PERFORMANCE

#define _CWORD33_LOG_SYSTEMEVENT(print_fmt, args...)

typedef struct _NSRingBufferHdr **NSRingBufferHdr**

typedef struct _NSRingBufferMtx **NSRingBufferMtx**

enum ESharedMemCommand { **eSharedMemCommandWrite** = ***, **eSharedMemCommandRead** = ***, **eSharedMemCommandSize** = ***, **eSharedMemCommandDelete** = *** }

Header for system-provided messages

typedef struct _pulse_t TPosixHdr

< *Header file for Template*

DEFINE_HANDLE_TYPE (SSHMemID, TShMemID)

#define **NS_SHAREDMEM_NAME** "NS_SharedMem"

#define **NS_SHAREDMEM_SOCK_QUEUE** 10

typedef struct _NSSharedBuffer **NSSharedBufferHdr**

#define **NSTEST_FAIL_SHAREDMEM_OPEN** "NSTEST_FAIL_SHAREDMEM_OPEN"

enum _SystemModeProtocol { SYSTEM_ON_INITIALIZATION = PROTOCOL_THREAD_INITIALIZATION, SYSTEM_ON_WAKEUP = PROTOCOL_THREAD_WAKEUP, SYSTEM_ON_SHUTDOWN = PROTOCOL_THREAD_SHUTDOWN, SYSTEM_ON_DESTROY = PROTOCOL_THREAD_DESTROY }*System Mode Protocol.*

typedef enum _SystemModeProtocol SystemModeProtocol

System Mode Protocol.

E_CWORD33_Status NSSharedMemTransmitLogOpen ()

E_CWORD33_Status NSSharedMemTransmitLogClose ()

SI_32 NSSharedMemReadTransmitLog (PSTR f_pBuffer, const UI_32 f_uiLength, const BOOL f_bBlock)

SI_32 NSSharedMemWriteTransmitLog (PCSTR f_pBuffer, const UI_32 f_uiLength)

E_CWORD33_Status NSSharedMemDumpTransmitLogToFile (PCSTR f_pPath, PUI_32 f_puiDumpSize)

BOOL NSSharedMemTransmitLogIsOpen ()

#define **TRANSMIT_LOG_SHAREDMEM_NAME** "/TransmitLogShBuf"

#define **TRANSMIT_LOG_SHAREDMEM_SIZE** 5242880

template<class T , UI_32 N> const T * **ArrBeg** (const T(&arr)[N])

template<class T , UI_32 N> const T * **ArrEnd** (const T(&arr)[N])

template<class O , class I > O **SimpleCast** (I i)

template<class R , class B > R **UnalignedRet** (B *b)

template<class Cont , class Gen > Cont **genRange** (UI_32 N, Gen genFn)

template<class OCont , class IIter , class MapFn > OCont **mapRange** (const IIter &begin, const IIter &end, MapFn fn)

template<typename T > Accumulator< T > MakeAccumulator (T n)

Accumulator utility function.

template<class TFn > CFunctor0< TFn > **functor** (TFn fn)

template<class TFn , class Arg1 > CFunctor1< TFn, Arg1 > **functor** (TFn fn, Arg1 arg1)

template<class TFn , class Arg1 , class Arg2 > CFunctor2< TFn, Arg1, Arg2 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2)

template<class TFn , class Arg1 , class Arg2 , class Arg3 > CFunctor3< TFn, Arg1, Arg2, Arg3 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3)

template<class TFn , class Arg1 , class Arg2 , class Arg3 , class Arg4 > CFunctor4< TFn, Arg1, Arg2, Arg3, Arg4 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4)

template<class TFn , class Arg1 , class Arg2 , class Arg3 , class Arg4 , class Arg5 > CFunctor5< TFn, Arg1, Arg2, Arg3, Arg4, Arg5 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4, Arg5 arg5)

template<class TFn , class Arg1 , class Arg2 , class Arg3 , class Arg4 , class Arg5 , class Arg6 > CFunctor6< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4, Arg5 arg5, Arg6 arg6)
template<class TFn , class Arg1 , class Arg2 , class Arg3 , class Arg4 , class Arg5 , class Arg6 , class Arg7 > CFunctor7< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4, Arg5 arg5, Arg6 arg6, Arg7 arg7)
#define **DISALLOW_COPY_AND_ASSIGN**(TypeName)
#define **DEFINE_EXCEPTION**(name, base)

Detailed Description

Class Documentation

struct _stLogEvent_

Class Members:

UI_16	cnt_ID	
UI_8	data[4]	
UI_8	event_id	
UI_8	grp_ID	
SystemPhase	phase	
UI_32	ts	
Datatype	typeofdata	

struct _stLogCount_

Class Members:

UI_16	cnt_id	
SystemPhase	phase	

struct __CWORD33_LOGPARAM

_CWORD33_LOG parameter

Class Members:

CHAR	cZones[USER_ZONES_COUNT][ZONE_TEXT_SIZE]	<output options (r NS_LoggingMethod_t)
T_CWORD33_LoggerSeverity	eSeverity	<mask optio ZONEMASK(10)...ZONEMASK((ref. T_CWORD33_Zone)
UI_8	uiLogOptions	
T_CWORD33_ZoneMask	uiZoneMask	<mask text ZONE_TEXT_10

		ZONE_TEXT_n

struct tag_Change_Log_Parameters

Class Members:

T_CWORD33_LoggerSeverity	eSeverity	
UI_8	uiLogOptions	
T_CWORD33_ZoneMask	uiZoneMask	

struct _TNS_EnableRcsLogging

Class Members:

BOOL	m_bEnable	
CHAR	m_cAppName[MAX_NAME_SIZE_APP]	

struct __CWORD33_LOGPPARAM

Performance LOG parameter.

Class Members:

BOOL	bIsPLogEnabled	PLog enabled or disabled.
EPLOG_TIME_FORMAT	eTimeFormat	Time format when PLog is printed.
UI_64	ui64GapInuS	Gap in micro seconds.
UI_8	uiLogOutputOptions	Output log options.

struct __CWORD33_LOGSYSEVENTPARAM

System event LOG parameter.

Class Members:

BOOL	bIsSysEventLogEnabled	SysEvent log enabled or disabled.

struct _NSRingBufferHdr

Class Members:

BOOL	m_bIsFull	
UI_32	m_uiReadPtr	
UI_32	m_uiRingBufferSize	
UI_32	m_uiUnReadSize	
UI_32	m_uiWritePtr	

struct _NSRingBufferMtx

Class Members:

int	m_lid	
pthread_mutex_t	m_tBufMutex	

struct SMsgHeader

Class Members:

uint16_t	cmd	
uintptr_t	id	
UI_32	size	

struct SReplyHeader

Class Members:

UI_32	size	
E_CWORD33_Status	status	

struct _NSSharedBuffer

Class Members:

BOOL	m_bIsFull	
pthread_mutex_t	m_tBufMutex	
pthread_cond_t	m_tCondVar	
UI_32	m_uiReadPtr	
UI_32	m_uiShMemSize	
UI_32	m_uiUnReadSize	
UI_32	m_uiWritePtr	

struct _NSTimerInfo

Timer info, defines the initial start of a timer, the repeat timer values and the cmd id for a timer.

Class Members:

UI_16	iCmd	
UI_64	rpt_nsec	
time_t	rpt_sec	
UI_64	t_nsec	
time_t	t_sec	

struct Int2Type

template<int v>

struct Int2Type< v >

Class Members:

	__unnamed__	

struct Type2Type

template<class T>

struct Type2Type< T >

Class Members:

typedef T	Type	

struct FSig

template<class Sig>

struct FSig< Sig >

Macro Definition Documentation

#define _CWORD33__APP_VERSION_BUILDVER() "undefined"

define macro for the building version

#define _CWORD33__APP_VERSION_MAJOR() (0)

define macro for the major number of the version

#define _CWORD33__APP_VERSION_MINOR() (0)

define macro for the minor number of the version

#define _CWORD33__APP_VERSION_PRODUCTID() "undefined"

define macro for product version

#define _CWORD33__APP_VERSION_REVISION() (0)

define macro for the revision number

#define _CWORD33__GET_VERSION() "undefined_undefined_00.00.00"

define macro for version string

#define _CWORD33__SET_ZONES()

Value:{UI_32 LogArr[] = {_CWORD33_LOGAPPZONES}; \
NsLogParseZones(&g__CWORD33_LogParams, (UI_32)_countof(LogArr), _CWORD33_LOGAPPZONES); \
NsLogSet_CWORD33_LogParams(&g__CWORD33_LogParams);}

Brief

This macro API register _CWORD33_LOGAPPZONES.

Return values:

none	

Prerequisite

none

Change of internal state

Classification

Public

Type

No match

See also:

NsLogSet_CWORD33_LogParams

#define _CWORD33_LOG(zone, funcname, print_fmt, args...) (zone != ZONEMASK(11))?((IS_ZONE_SET(zone))?TEXT_ZONE(zone, funcname, print_fmt, ## args):((void)0)):((void)(0))

Brief

Developers should use the _CWORD33_LOG for logging.

Parameters:

in	*zone*	Defined zone in a process (e.g. ZONE_FUNC)
in	*funcname*	Name of the class or method where the log was written (e.g. "main")
in	*print_fmt*	string similar to that of a printf statement
in	*args*	Data arguments(option)

Return values:

none	

Prerequisite

Change of internal state

Classification

Public

Type

No match

See also:

ZONEMASK, IS_ZONE_SET, NsLog, NsLog0

#define _CWORD33_LOG0 _CWORD33_LOG

Deprecated API. Use _CWORD33_LOG instead.

Used to log data that contains trailing arguments

Example: _CWORD33_LOG0(ZONE_INFO, **FUNCTION** , "testers return value is [%d]!", rtnValue);

#define _CWORD33_LOG_CNT(zone, CntId, n...) NsLog_Cnt(CntId, ##n)

_CWORD33_LOG_CNT

Counter logger mechanism.(Send data to SS event logger Queue)

Note:

The ZONE is not used.

Parameters:

in	*zone*	Defined zone in a process (e.g. ZONE_FUNC)
in	*CntId*	UI_16 - ID of Event from comm layer
in	*n*	UI_8 - number of variable arguments inputted

Deprecated API. Not use.

#define _CWORD33_LOG_EVT(zone, EvtId, n...) NsLog_Evt(EvtId, ##n)

_CWORD33_LOG_EVT

Event logger mechanism.(Send data to SS event logger Queue)

Note:

The ZONE is not used.

Parameters:

in	*zone*	Defined zone in a process (e.g. ZONE_FUNC)
in	*EvtId*	UI_16 - ID of Event from comm layer
in	*n*	UI_8 - number of variable arguments inputted

Deprecated API. Not use.

#define _CWORD33_LOG_EVTCNT(zone, CntId, EvtId, n...) NsLog_EvtCnt(CntId, EvtId, ##n)

_CWORD33_LOG_EVTCNT

Deprecated API. Not use.

#define _CWORD33_LOG_PERFORMANCE(print_fmt, args...) ((IS_PLOG_ENABLED()) ? PLOG_TEXT(print_fmt, ## args): ((void)0))

_CWORD33_LOG_PERFORMANCE

_CWORD33_LOG_PERFORMANCE is a logging API provided specifically for logging performance metrics.

Note:

This macro API is disable in Linux system It should be used in few locations within an application. (Max 5 times per application). It should never be used in place of _CWORD33_LOG.

Parameters:

in	*print_fmt*	string similar to that of a printf statement
in	*args*	Data arguments

#define _CWORD33_LOG_PERFORMANCE_DEBUG(print_fmt, args...) ((IS_PLOG_ENABLED()) ? PLOG_TEXT(print_fmt, ## args): ((void)0))

_CWORD33_LOG_PERFORMANCE_DEBUG

Prints a performance log in PosixBasedOS001 system log. These logs can be viewed by using utility sloginfo.

Note:

This macro API is disable in Linux system This macro API is disable if defined "___CWORD33_LOG_NDEBUG_LEVEL2__"

Parameters:

in	*print_fmt*	PCSTR - Input string
in	*args*	string similar to that of a printf statement and data argments

#define _CWORD33_LOG_SYSTEMEVENT(print_fmt, args...)

Value:((IS_SYSEVENTLOG_ENABLED()) ?\
SYSEVENTLOG_TEXT(print_fmt, ## args): ((void)0))

_CWORD33_LOG_SYSTEMEVENT

Parameters:

in	*print_fmt*	string similar to that of a printf statement
in	*args*	Data arguments

#define _CWORD33_LOGZONES {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

extern _CWORD33_LOG parameter

Global variable that's used by the Native Services Logger. An application will setup this variable in their main, values are user defined and provide user control of what should and shouldn't get logged.

```
1 _CWORD33_LOGPARAM g__CWORD33_LogParams =
2 {
3  _CWORD33_LOGOPTIONS,
4   {
5     ZONE_TEXT_10, ZONE_TEXT_11, ZONE_TEXT_12,
6     ZONE_TEXT_13, ZONE_TEXT_14, ZONE_TEXT_15,
7     ZONE_TEXT_16, ZONE_TEXT_17, ZONE_TEXT_18,
8     ZONE_TEXT_19, ZONE_TEXT_20, ZONE_TEXT_21,
9     ZONE_TEXT_22, ZONE_TEXT_23, ZONE_TEXT_24,
10     ZONE_TEXT_25, ZONE_TEXT_26, ZONE_TEXT_27,
11    ZONE_TEXT_28, ZONE_TEXT_29, ZONE_TEXT_30,
12    ZONE_TEXT_31, ZONE_TEXT_32, ZONE_TEXT_33,
13    ZONE_TEXT_34, ZONE_TEXT_35, ZONE_TEXT_36,
14    ZONE_TEXT_37, ZONE_TEXT_38, ZONE_TEXT_39,
15    ZONE_TEXT_40, ZONE_TEXT_41, ZONE_TEXT_42,
16    ZONE_TEXT_43, ZONE_TEXT_44, ZONE_TEXT_45,
17    ZONE_TEXT_46, ZONE_TEXT_47, ZONE_TEXT_48,
18    ZONE_TEXT_49, ZONE_TEXT_50, ZONE_TEXT_51,
19    ZONE_TEXT_52, ZONE_TEXT_53, ZONE_TEXT_54,
```

```
20    ZONE_TEXT_55
21    },
22    _CWORD33_LOGZONES
23 };
```

if _CWORD33_LOGZONES not defined in makefile

g__CWORD33_LogParams if _CWORD33_LOGZONES not defined in makefile

#define DEFINE_EXCEPTION(name, base)

Value:class name : public base \
{ \
public: \
  name (const char* str = "") : base (str) \
  {} \
}

#define DISALLOW_COPY_AND_ASSIGN(TypeName)

Value: TypeName(const TypeName&); \
  void operator=(const TypeName&)

#define IS_ZONE_SET(set_zone) (NsLogIsZoneSet(set_zone) == TRUE)

Brief

check wheather the _CWORD33_LOG ZONE is set by your application process.

Parameters:

in	*set_zone*	UI_32 - checking zone_index

Return values:

TRUE	zone is enable
FALSE	zone is disable

Prerequisite

none

Change of internal state

Classification

Public

Type

No match

See also:

NsLogIsZoneSet

#define NS_CLEARBIT(x, i) ((x) &= ~(1 << (i)))

Macro for setting the i bit of x to zero

#define NS_INVALID_RETURN -1

Macro for invalid/undefined return

#define NS_ISBITSET(x, i) (((x) & (1 << (i))) != *)**

Macro for checking whether the i bit of x is one or not

#define NS_SETBIT(x, i) ((x) |= (1 << (i)))

Macro for setting the i bit of x to one

#define NS_TOGGLEBIT(x, i) ((x) ^= (1 << (i)))

Macro for making the exclusive-OR operator on the i bit of x with one

#define NsLogFunc(...)

Value:
```
GET_MACRO(__VA_ARGS__, \
    NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, \
    NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, \
    NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, \
    NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, \
    NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, \
    NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, \
    NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, \
    NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog, NsLog0)
```

#define ZONEMASK(n) ((T_CWORD33_Zone)(n))

Brief

the Macro for zone index mask

Parameters:

in	*n*	zone index

Return values:

num	UI_32 - zone index

Prerequisite

none

Change of internal state

none

Classification

Public

Type

No match

See also:

none

Typedef Documentation

typedef enum _NS_LOGGER_METHODS NS_LoggingMethod_t

_CWORD33_LOG output direction

Note:

LMSGQ and LSLOGGER are disabled

typedef unsigned int TMemID

< Standard *CWORD3* Types

Type of memory buffer identifiers

Enumeration Type Documentation

enum _EPLOG_OUTPUT_OPTION

Enumerator

EPLOG_SLOG log to standard logging i.e. slogger

EPLOG_MSGQ log to message queue NSPlog

EPLOG_TRACEEVENT log to trace events

enum _EPLOG_TIME_FORMAT

Performance logging.

Enumerator

EPLOG_TIME_FORMAT_USEC microseconds

EPLOG_TIME_FORMAT_MSEC miliseconds

EPLOG_TIME_FORMAT_SEC seconds

enum **_NS_LOGGER_METHODS**

_CWORD33_LOG output direction

Note:
LMSGQ and LSLOGGER are disabled

Enumerator

LMSGQ <Print to the console(similar to printf)

LSLOGGER <Disabled

LSHAREDMEM <Disabled

enum **_SystemModeProtocol**

System Mode Protocol.

Enumerator

SYSTEM_ON_INITIALIZATION Initialize thread after creation.

SYSTEM_ON_WAKEUP Wakeup Thread after sleep.

SYSTEM_ON_SHUTDOWN Stop the thread.

SYSTEM_ON_DESTROY Destroy the thread.

enum **_T_CWORD33_LoggerSeverity**

Enumerator

_CWORD33__SEVERITY_DEBUG2 <all enable(default)

_CWORD33__SEVERITY_DEBUG1 <enable _CWORD33_LOG_DEBUG() and _CWORD33_LOG_DEBUG2() and all the follows

_CWORD33__SEVERITY_INFO <enable _CWORD33_LOG_DEBUG1() and all the follows

_CWORD33__SEVERITY_WARN <enable _CWORD33_LOG_INFO() and all the follows

_CWORD33__SEVERITY_ERROR <enable _CWORD33_LOG_WARN() and all the follows

_CWORD33__SEVERITY_FATAL <enable _CWORD33_LOG_ERROR() and all the follows

_CWORD33__SEVERITY_ALWAYS <enable _CWORD33_LOG_FATAL() and _CWORD33_LOG_ALWAYS()

<enable _CWORD33_LOG_ALWAYS()

Function Documentation

int _CWORD33_AcquireResouce (const char * *mod*, const char * *key*, long * *resource*)

Summary

This function is used to acquire the resourceid associated with the mod and the key.

Parameters:

[IN]	mod const char * - the mod of resource
[IN]	key const char * - the key of resource
[OUT]	resource long * - resource ID

Return values:

ref_counter	the counter of user

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

int _CWORD33_GetResource (const char * *mod*, const char * *key*, long * *resource*)

Summary

This function is used to get the resourceid associated with the mod and the key.

Parameters:

[IN]	mod const char * - the mod of resource
[IN]	key const char * - the key of resource
[OUT]	resource long * - resource ID

Return values:

-1	fail
0	success

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

int _CWORD33_RegistResouce (const char * *mod*, const char * *key*, long *resource*, int *init_counter*)

Summary

This function is used to register the resource associated with the mod and the key.

Parameters:

[IN]	mod const char * - the mod of resource
[IN]	key const char * - the key of resource
[IN]	resource const char * - resource ID
[IN]	init_counter long - resource user counter

Return values:

-1	fail
0	success

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

int _CWORD33_ReleaseResouce (const char * *mod*, const char * *key*)

Summary

This function is used to release the resourceid associated with the mod and the key.

Parameters:

[IN]	mod const char * - the mod of resource
[IN]	key const char * - the key of resource

Return values:

ref_counter	the counter of user

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

int _CWORD33_SearchResourseKey (const char * *mod*, long *resource*, const char ** *key*)

Summary

This function is used to get the key associated with the mod and the resourceid.

Parameters:

[IN]	mod const char * - the mod of resource
[IN]	resource long - resource ID
[OUT]	key const char * - the key of resource

Return values:

-1	fail
0	success

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

int _CWORD33_UnregistResouce (const char * *mod*, const char * *key*)

Summary

This function is used to unregister the resource associated with the mod and the key.

Parameters:

[IN]	mod const char * - the mod of resource
[IN]	key const char * - the key of resource

Return values:

0	success

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

DEFINE_EXCEPTION (lock_error , std::runtime_error)

define the lock_error class inherit from std::runtime_error

DEFINE_EXCEPTION (lock_creation_error , lock_error)

define the lock_creation_error class inherit from lock_error

DEFINE_EXCEPTION (lock_acquireread_error , lock_error)

define the lock_acquireread_error class inherit from lock_error

DEFINE_EXCEPTION (lock_acquirewrite_error , lock_error)

define the lock_acquirewrite_error class inherit from lock_error

DEFINE_EXCEPTION (lock_release_error , lock_error)

define the lock_release_error class inherit from lock_error

E_CWORD33_Status DiscardDataFromShared (TMemID *id*)

Summary

Discard a chunk of memory in SharedMem

Parameters:

in	*id*	13 TMemID - MemID of the shared data

Return values:

e_CWORD33_StatusOK	- Success
e_CWORD33_StatusInvldParam	-Invalid parameter
e_CWORD33_StatusInvldID	-Invalid id
e_CWORD33_StatusFail	-Some sort of error occurred

Preconditions

Change of the internal state

None

Classification

Public

Type

sync only

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

None

E_CWORD33_Status GetDataFromShared (TMemID *id*, void * *data*, UI_32 *dataMaxBytes*)

Summary

Retrieve a large chunk of memory using SharedMem service

Parameters:

in	*id*	14 TMemID - MemID of the shared data
in	*data*	15 void* - Buffer to store data in, must be at least as large as buffer associated with id
in	*dataMaxBytes*	16 UI_32 - Size of data - system will never copy more than dataMaxBytes bytes into data

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusInvldID	Invalid id
e_CWORD33_StatusFail	Some sort of error occurred

Preconditions

Change of the internal state

None

Classification

Public

Type

sync only

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

None

UI_32 GetLengthOfDataFromShared (TMemID *id*)

Summary

Query the minimum buffer size, in bytes, necessary to hold a chunk of memory in SharedMem

Parameters:

in	*id*	17 TMemID - MemID of the shared data

Return values:

length	- Minimum buffer size, in bytes, required to hold the data associated with id.

Preconditions

Change of the internal state

None

Classification

Public

Type

sync only

TMemIDResult
Except 0:Success
0:Failure

See also:
None

template<typename T > Accumulator<T> MakeAccumulator (T *n*)

Accumulator utility function.

Given a value n of type T, returns an accumulator of type Accumulator<T> initialized to value n

E_CWORD33_Status McClearData (PVOID *data*)

Brief
Clears the data that may be pending for the a message in shared memory.

Parameters:

in	*data*	PVOID - pointer to the data was received from the McReceive Call

Return values:

E_CWORD33_Status	indicates if the data was clear successfully
e_CWORD33_StatusOK	- success
e_CWORD33_StatusInvldBuf	- invalid buffer
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusInvldID	- invalid ID
e_CWORD33_StatusFail	- other errors

Preconditons

Message is received by McReceive and so on.

Change of internal status

none

Classification
Public

Type
Sync only

See also:
McReceive, McReceiveWithSession

E_CWORD33_Status McClose (HANDLE *hMessage*)

Brief

Close a connection to a Receiver message queue.

Parameters:

in	*hMessage*	HANDLE - handle that will be closed

Return values:

E_CWORD33_Status	indicates if the close was successfully
e_CWORD33_StatusOKsuccess	
e_CWORD33_StatusInvldHandle	- If hMessage is NULL or invalid
e_CWORD33_StatusInvldHndlType-if	hMessage is invalid

Preconditons

message queue(McOpenSenderMcOpenReceiver) is created in advance

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenSender, McOpenReceiver, McOpenSyncReceiver

E_CWORD33_Status McCreateInvokerName (PCSTR *source*, UI_32 *sessionid*, PSTR *invokerName*, UI_32 *size*)

Brief

Create synchronous invoke response MQ name.

Parameters:

in	*source*	PCSTR - source service name
in	*sessionid*	UI_32 - session id
in	*invokerName*	PSTR - create invoker name buffer
in	*size*	UI_32 - InvokerName buffer size

Return values:

E_CWORD33_Status	indicates if the close was successfully
e_CWORD33_StatusOK	
e_CWORD33_StatusInvldParam	-if parameter is invalid

Preconditons

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenSyncReceiver

void McFlushReceiver (HANDLE *hMessage*)

Brief

Flush's all data on HANDLE's Receiver message queue.

Parameters:

in	*hMessage*	HANDLE - handle to the recvMessage queue that will be flushed

Return values:

void	

Preconditons

message queue(McOpenReceiver)is created in advance

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenSender, McOpenReceiver, McOpenSyncReceiver

E_CWORD33_Status McForward (HANDLE *hMessage*, PCSTR *source*, UI_32 *iCmd*, TMemID *USID*)

Brief

Forward a Message to indicate that there is a Shared Memory

Parameters:

in	*hMessage*	HANDLE - handle to the send message queue, McOpenSender
in	*source*	PCSTR - the sender of this forwarding message.

in	*iCmd*	UI_32 - Command Id message to forward
in	*USID*	UI_32 - Unique Shared memory Id

Return values:

E_CWORD33_Status	indicates if the close was successfully CONST char* - constant pointer to source message queue
Success(e_CWORD33_StatusOK)	
Failed(anything	else)

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

E_CWORD33_Status McGetDataOfSize (PVOID *data*, PVOID *to*, UI_32 *uiSize*)

Brief

Get the message data, this may be from a queue or shared memory where the maximum size of the data should not exceed the maximum size passed in. And deletes the data if it is stored in shared memory.

Parameters:

in	*data*	PVOID - pointer to the data was received from the McReceive Call
in	*to*	PVOID - pointer to the data to be received
in	*uiSize*	UI_32 - maximum size of the buffer to which the received data is copied

Return values:

E_CWORD33_Status	indicates if the message was sent successfully
e_CWORD33_StatusOK	- success
e_CWORD33_StatusInvldBuf	- invalid buffer
e_CWORD33_StatusInvldHandle	- invalid handle
e_CWORD33_StatusInvldHndlType	- invalid handle type
e_CWORD33_StatusInvldBufSize	- invalid buffer size
e_CWORD33_StatusErrOther	- invalid shared memory ID specified by received message

Preconditons

Message is received by McReceive and so on.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McGetLength

E_CWORD33_Status McGetDataOfSizeWithSMRetain (PVOID *data*, PVOID *to*, UI_32 *uiSize*)

Brief

Get the message data, this may be from a queue or shared memory where the maximum size of the data should not exceed the maximum size passed in. Does not delete the data if it is stored in shared memory.

Parameters:

in	*data*	PVOID - pointer to the data was received from the McReceive Call
in	*to*	PVOID - pointer to the data to be received
in	*uiSize*	UI_32 - maximum size of the buffer to which the received data is copied

Return values:

E_CWORD33_Status	indicates if the message was sent successfully
e_CWORD33_StatusOK	- success
e_CWORD33_StatusInvldBuf	- invalid buffer
e_CWORD33_StatusInvldHandle	- invalid handle
e_CWORD33_StatusInvldHndlType	- invalid handle type
e_CWORD33_StatusInvldBufSize	- invalid buffer size
e_CWORD33_StatusErrOther	- invalid shared memory ID specified by received message

Preconditons

Message is received by McReceive and so on.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McReceive

PVOID McGetDataPointer (PVOID *data*)

Brief

Get header address of data from received message.

Parameters:

in	*data*	PVOID - received message data

Return values:

PVOID	- data address of received message

Preconditons

Message is received by McReceive and so on.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McReceive, McReceiveWithSession, McGetDataOfSize

TMemID McGetDataUSID (PVOID *pData*)

Brief

Get the USID is the Unique Shared Memory ID.e. from the message that was received.

Parameters:

in	*pData*	void* - pointer to the data was received from the McReceive Call

Return values:

TMemID	Type of memory buffer identifiers(unsigned int).
NoneZero(Success)	
Zero(Failed)	

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

UI_32 McGetLength (PVOID *data*)

Brief

Get the length of data that needs to be retrieved.

Parameters:

in	*data*	PVOID - pointer to the data was received from the McReceive Call

Return values:

UI_32	indicates the number of bytes that in the message

if return is 0 or less then invalid data was passed.

Preconditons

Message is received by McReceive and so on.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McReceive, McReceiveWithSession, McGetDataOfSize

PCSTR McGetMsgSrc (PVOID *data*)

Brief

Gets the constant pointer to source message queue.

Parameters:

in	*data*	PVOID - pointer to the data to be received

Return values:

PCSTR	message queue's address CONST char* - constant pointer to source message queue
NotNull(Success)	
Null(Failed)	

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

int McGetQueueFD (HANDLE *hMessage*)

Brief

Gets the fd of the message queue.

Parameters:

in	*hMessage*	HANDLE - handle of the message queue.(handle created with McOpenReceiver or McOpenSender)

Return values:

int	fd - fd of the message queue if handle is valid else -1

Preconditons

- Dispatcher for application is created and initialized(_CWORD33_CreateDispatcherWithoutLoop) in advance

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenReceiver, McOpenSender

PCSTR McGetQueueName (HANDLE *hMessage*)

Brief

Gets the name of the message queue.

Parameters:

in	*hMessage*	HANDLE - handle of the message queue, (handle created

		with McOpenReceiver or McOpenSender)

Return values:

PCSTR	name - name of the message queue if handle is valid else NULL

Preconditons

Dispatcher for application should be created and initilized(_CWORD33_CreateDispatcherWithoutLoop)in advance

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenReceiver, McOpenSender

E_CWORD33_Status McGetSysInfoData (PVOID *data*, PVOID *to*)

Brief

Gets the data from system info buffer from message header.

Parameters:

in	*data*	PVOID - pointer to the data was received from the McReceive Call
out	*to*	PVOID - pointer to the data to be received

Return values:

E_CWORD33_Status	indicates if the close was successfully
e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldBuf	- Invalid buffer
e_CWORD33_StatusInvldHandle	- Invalid handle

Preconditons

Message is received by McReceive and so on.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McReceive, McReceiveWithSession

E_CWORD33_Status McInvokeSync (HANDLE *hMessage*, PCSTR *source*, UI_32 *cmd*, UI_32 *msgLength*, PCVOID *msgData*, UI_32 *sessionid*, HANDLE *hRcvMessage*, UI_32 *responseLength*, PVOID *responseData*, UI_32 * *receivedLength*)

Brief

Synchronous Invoke. Send data to message queue, and Receive data from message queue.

Parameters:

in	*hMessage*	HANDLE - handle to the send message queue, McOpenSender
in	*source*	PCSTR - app (You) the sender
in	*cmd*	UI_32 - command id aka the message that's being sent
in	*msgLength*	UI_32 - length of the data buffer provided
in	*msgData*	PCVOID - pointer to the data to be sent
in	*sessionid*	UI_32 - session ID
in	*hRcvMessage*	HANDLE - handle to the receive message queue, McOpenSyncReceiver
in	*responseLength*	UI_32 - length of the responnce receive buffer provided
out	*responseData*	PVOID - pointer to the data to be receive
out	*receivedLength*	UI_32* - length of the data to be receive

Return values:

E_CWORD33_Status	indicates invoke response or message sent error or response receive error
e_CWORD33_StatusOK	- success
e_CWORD33_StatusInvldParam	- Invalid parameter
e_CWORD33_StatusInvldHandle	- Invalid handle
e_CWORD33_StatusInvldHndlType	- Invalid handle type
e_CWORD33_StatusInvldQName	- Invalid message queue name
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_StatusInvldBuf	- Invalid buffer
e_CWORD33_StatusBadConnection	- Can't connect with Socket
e_CWORD33_StatusFail	- Some sort of error occurred
e_CWORD33_StatusErrOther	- other errors(Failed to open/allocate shared memory)

Preconditons

Message queue specified by argument hMessage and hRcvMessage is useful.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenSender, McOpenSyncReceiver, _CWORD33_InvokeSync

HANDLE McOpenReceiver (PCSTR *name*)

Brief

Opens a handle to a Receiver message queue.

Parameters:

in	*name*	PCSTR - name of the message queue you want to receiver messages on

Return values:

Handle	to a recevier's message queue (If INVALID_HANDLE is returned its an error)

Preconditons

none

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenSender, McReceive, McClose

HANDLE McOpenReceiverNotBlocked (PCSTR *name*)

Brief

Opens a handle to a Receiver message queue. Operations on this queue are non-blocking if queue is empty or full.

Parameters:

in	*name*	PCSTR - name of the message queue you want to receiver messages on

Return values:

Handle	to a recevier's message queue (If INVALID_HANDLE is returned its an error)

Preconditons

none

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenSender, McReceive, McClose, McOpenSenderNotBlocked

HANDLE McOpenSender (PCSTR *name*)

Brief

The opens a handle for sending messages to another message queue.

Parameters:

in	*name*	PCSTR - name of the message queue you want to send messages too

Return values:

Handle	for sending messages to a queue (If INVALID_HANDLE is returned its an error)

Preconditons

none

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenReceiver, McSend, McClose

HANDLE McOpenSenderNotBlocked (PCSTR *name*)

The opens a handle for sending messages to another message queue. operations on this queue are non-blocking if queue is empty or full.

Parameters:

in	*name*	PCSTR - name of the message queue you want to send messages too

Returns:

HANDLE handle - for sending messages to a queue (If INVALID_HANDLE is returned its an error)

HANDLE McOpenSyncReceiver (PCSTR *name*)

Brief

Opens a handle to a response Receiver message queue.

Parameters:

in	*name*	PCSTR - name of the message queue you want to receiver messages on

Return values:

Handle	to a recevier's message queue (If INVALID_HANDLE is returned its an error)

Preconditons

none

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McCreateInvokerName, McInvokeSync, McClose

HANDLE McOpenSyncSender (PCSTR *name*)

Brief

The opens a handle for sending responce to another message queue.

Parameters:

in	*name*	PCSTR - name of the message queue you want to send messages too

Return values:

Handle	for sending response to a queue (If INVALID_HANDLE is returned its an error)

Preconditons

none

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenReceiver, McSend, McClose

E_CWORD33_Status McReceive (HANDLE *hMessage*, PSTR *source*, UI_32 * *cmd*, UI_32 *length*, PVOID *data*)

Brief

Retrieves data from a message queue.

Parameters:

in	*hMessage*	HANDLE - handle to the recvMessage, McOpenRecevier
out	*source*	PSTR - app that sent this message
out	*cmd*	UI_32* - command message that has been received.
in	*length*	UI_32 - length of the data buffer provided
out	*data*	PVOID - pointer to the data to be received

Return values:

E_CWORD33_Status	indicates if the message was sent successfully
e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	- Invalid handle
e_CWORD33_StatusInvldBuf	- Invalid buffer
e_CWORD33_StatusInvldQName	- Invalid message queue name
e_CWORD33_StatusInvldParam	- Invalid parameter
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	- Invalid handle of message queue for receiving message

Preconditons

Message queue(McOpenReceiver etc.) for receiving message is created.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenReceiver, McSend

E_CWORD33_Status McReceiveWithSession (HANDLE *hMessage*, PSTR *source*, UI_32 * *cmd*, UI_32 * *sessionid*, UI_32 *length*, PVOID *data*)

Brief

Retrieves data from a message queue.

Parameters:

in	*hMessage*	HANDLE - handle to the recvMessage, McOpenRecevier
out	*source*	PSTR - app that sent this message
out	*cmd*	UI_32* - command message that has been received.
out	*sessionid*	UI_32* - Session ID
in	*length*	UI_32 - length of the data buffer provided
out	*data*	PVOID - pointer to the data to be received

Return values:

E_CWORD33_Status	indicates if the message was sent successfully
e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldHandle	- Invalid handle
e_CWORD33_StatusInvldBuf	- Invalid buffer
e_CWORD33_StatusInvldQName	- Invalid message queue name
e_CWORD33_StatusInvldParam	- Invalid parameter
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	- Invalid handle of message queue for receiving message

Preconditons

Message queue(McOpenReceiver etc.) for receiving message is created.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenReceiver, McSendWithSession

E_CWORD33_Status McSend (HANDLE *hMessage*, PCSTR *source*, UI_32 *cmd*, UI_32 *length*, PCVOID *data*)

Brief

Sends data to a message queue.

Parameters:

in	*hMessage*	HANDLE - handle to the send message queue, McOpenSender
in	*source*	PCSTR - app (You) the sender
in	*cmd*	UI_32 - command id aka the message that's being sent
in	*length*	UI_32 - length of the data buffer provided
in	*data*	PCVOID - pointer to the data to be sent

Return values:

E_CWORD33_Status	indicates if the message was sent successfully
e_CWORD33_StatusOK	- Success
e_CWORD33_StatusNullPointer	- NULL pointer specified
e_CWORD33_StatusInvldHandle	- Invalid handle
e_CWORD33_StatusInvldHndlType	- Invalid type of handle
e_CWORD33_StatusInvldQName	- Illegal Message Queue name
e_CWORD33_StatusMsgQFull	- Message queue is full
e_CWORD33_StatusErrNoEBADF	I- nvalid 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 errors(Failed to open/allocate shared memory)
e_CWORD33_StatusInvldParam	- Invalid parameter

Preconditons

Message queue(McOpenSender) for sending message is created.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenSender, McSendWithSession, McReceive

E_CWORD33_Status McSendSyncResponse (HANDLE *hMessage*, PCSTR *source*, UI_32 *cmd*, UI_32 *seq_id*, E_CWORD33_Status *ret_val*, UI_32 *length*, PCVOID *data*)

Brief

Sends response to a message queue.

Parameters:

in	*hMessage*	HANDLE - handle to the send message queue, McOpenSyncSender
in	*source*	PCSTR - app (You) the sender
in	*cmd*	UI_32 - command id aka the message that's being sent
in	*seq_id*	UI_32 - response sequence ID
in	*ret_val*	E_CWORD33_Status - response status
in	*length*	UI_32 - length of the data buffer provided
in	*data*	PCVOID - pointer to the data to be sent

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	- If handle is NULL or invalid
e_CWORD33_StatusInvldHndlType-if	handle type is invalid
e_CWORD33_StatusInvldQName	- if MQ name is invalid
e_CWORD33_StatusMsgQFull	- if MQ is full
e_CWORD33_StatusErrNoEBADF	- if fd is invalid
e_CWORD33_StatusErrNoEINTR	- if system call(signal) interupt occurs
e_CWORD33_StatusInvldBufSize	-if buffer size is invalid
e_CWORD33_StatusInvldBuf	-if buffer is invalid
e_CWORD33_StatusFail	- any other error
e_CWORD33_StatusErrOther	-other error(memory open/allocate fail)

Preconditons

none

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenSender, McOpenSyncReceiver, McInvokeSync, _CWORD33_SetSyncResponseData

E_CWORD33_Status McSendWithPriority (HANDLE *hMessage*, PCSTR *source*, UI_32 *cmd*, UI_32 *length*, PCVOID *data*, E_CWORD33_MessagePriorties *priority*, UI_32 *sessionid*)

Brief

Sends data to a message queue, inserts based on priority given.

Parameters:

in	*hMessage*	HANDLE - handle to the send message queue, McOpenSender
in	*source*	PCSTR - app (You) the sender
in	*cmd*	UI_32 - command id aka the message that's being sent
in	*length*	UI_32 - length of the data buffer provided
in	*data*	PCVOID - pointer to the data to be sent
in	*sessionid*	UI_32 - session id

Return values:

E_CWORD33_Status	indicates if the message was sent successfully
e_CWORD33_StatusOK(success)	
e_CWORD33_StatusFail(anything	else)

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

E_CWORD33_Status McSendWithSession (HANDLE *hMessage*, PCSTR *source*, UI_32 *cmd*, UI_32 *length*, PCVOID *data*, UI_32 *sessionid*)

Brief

Sends data to a message queue.

Parameters:

in	*hMessage*	HANDLE - handle to the send message queue, McOpenSender
in	*source*	PCSTR - app (You) the sender
in	*cmd*	UI_32 - command id aka the message that's being sent
in	*length*	UI_32 - length of the data buffer provided

in	*data*	PCVOID - pointer to the data to be sent
in	*sessionid*	UI_32 - session id

Return values:

E_CWORD33_Status	indicates if the message was sent successfully
e_CWORD33_StatusOK	- Success
e_CWORD33_StatusNullPointer	- NULL pointer specified
e_CWORD33_StatusInvldHandle	- Invalid handle
e_CWORD33_StatusInvldHndlType	- Invalid type of handle
e_CWORD33_StatusInvldQName	- Illegal Message Queue name
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 errors(Failed to open/allocate shared memory)
e_CWORD33_StatusInvldParam	- Invalid parameter

Preconditons

Message queue(McOpenSender) for sending message is created.

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McOpenSender, McSend, McReceiveWithSession, _CWORD33_SendMsg

E_CWORD33_Status McTranslateError (int *error*)

Brief

Translates global error variables into FW E_CWORD33_Status

Parameters:

in	*error*	int - error variable

Return values:

E_CWORD33_Status	indicates *CWORD33* error type

E_CWORD33_Status Enum

```
1 typedef enum e_CWORD33_Status {
2   e_CWORD33_StatusEmptyMediaList    = ***, ///< Empty media list
3   e_CWORD33_StatusSessionLimitMaxedOut  = ***,  ///< Maximum session limit reached
4   e_CWORD33_StatusDbRecNotFound      = ***, ///< Database record not found
```

```
 5  e_CWORD33_StatusDbResultError       = ***, ///< Database result error
 6  e_CWORD33_StatusDbExecuteFail       = ***, ///< Database execute fail
 7  e_CWORD33_StatusSemCloseFail        = ***, ///< Semaphore close failed
 8  e_CWORD33_StatusSemUnlockFail       = ***, ///< Semaphore unlock failed
 9  e_CWORD33_StatusSemLockFail         = ***, ///< Semaphore lock failed
10   e_CWORD33_StatusFail               = ***, ///< Failed
11   e_CWORD33_StatusErrOther           = ***, ///< Unknown error
12   e_CWORD78_StatusOK                 = ***,  ///< Success / Pass / OK
13   e_CWORD33_StatusInvldBuf           = ***,  ///< Invalid buffer
14   e_CWORD33_StatusInvldHandle        = ***,  ///< Invalid handle
15   e_CWORD33_StatusInvldHndlType      = ***,  ///< Invalid handle type
16   e_CWORD33_StatusInvldQName         = ***,  ///< Invalid messasge queue name
17   e_CWORD33_StatusMsgQFull           = ***,  ///< Message queue full
18   e_CWORD33_StatusInvldNotification  = ***,///< The Notification event not present
19   e_CWORD33_StatusInvldParam         = ***,  ///< Invalid parameter
20   e_CWORD33_StatusInvldBufSize       = ***,  ///< Buf size too small
21   e_CWORD33_StatusInvldID            = ***,  ///< Unrecognized ID
22   e_CWORD33_StatusCannotRelease      = ***, ///< Cannot release resource
23   e_CWORD33_StatusBadConnection      = ***, ///< Could not locate resource
24   e_CWORD33_StatusExit           = ***,  ///< Normal application termination
25   e_CWORD33_StatusNotImplemented     = ***,  ///< incomplete feature
26   e_CWORD33_StatusThreadBusy         = ***,  ///< Joined thread is already being joined
27   e_CWORD33_StatusThreadSelfJoin     = ***,  ///< Thread is joining itself
28   e_CWORD78_StatusThreadInvalidVal   = ***, ///< Invalid value passed
29   e_CWORD33_StatusThreadNotExist     = ***,  ///< The thread does not exist
30   e_CWORD33_StatusFault              = ***,  ///< A fault occurred while attempting to make call
31   e_CWORD33_StatusServNotFound       = ***,  ///< Service not present in serv dir
32   e_CWORD33_StatusServerInUse        = ***,  ///< Service already processing 1 client request
33   e_CWORD33_StatusDbIndexing         = ***,  ///< Database Indexing in progress
34   e_CWORD33_StatusNullPointer        = ***,
35   e_CWORD33_StatusMsgNotProcessed    = ***,
36   e_CWORD33_StatusFileLoadSuccess    = ***, ///< File Load Success
37   e_CWORD33_StatusFileLoadError      = ***, ///< File Load Error
38   e_CWORD33_StatusAccessError        = ***, ///< Error when accessing resource
39   e_CWORD33_StatusDuplicate          = ***, ///< Duplicate entry
40   e_CWORD33_StatusMsgQEmpty          = ***, ///< Message queue empty
41   e_CWORD33_StatusThreadAlreadyRunning  = ***,
42   e_CWORD33_StatusErrNoEBADF         = ***,  ///< Bad file descriptor
43   e_CWORD33_StatusErrNoEAGAIN        = ***,  ///< Resource unavailable, try again
44   e_CWORD33_StatusErrNoEINTR         = ***,  ///< Interrupted system call
45   e_CWORD33_StatusSessionErr     = ***,  ///< Error in session handling
46   e_CWORD33_StatusDBCorrupt      = ***,  ///< Database corrupt
47   e_CWORD33_StatusDBFileNotFound    = ***  ///< Database file not found
48 } E_CWORD33_Status, *PE_CWORD33_Status;
```

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

E_CWORD33_Status McZcClose (HANDLE *handle*)

Brief

close the handle to message queue

Parameters:

in	*handle*	HANDLE - handle to the send message queue, handle got through McZcOpenSender

Return values:

E_CWORD33_Status	indicates if the message was close successfully
e_CWORD33_StatusOK	
e_CWORD33_StatusInvldHandle	- If hMessage is NULL or invalid
e_CWORD33_StatusInvldHndlType	If handle type is invalid

Preconditons

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McZcOpenSender

PVOID McZcGetBuf (HANDLE *handle*)

Brief

get the fist address of buffer for sending

Parameters:

in	*handle*	HANDLE - handle to the send message queue, McZcOpenSender

Return values:

PVOID	the first address of buffer for data-sending
NULL	NULL pointer(fail to get)

Preconditons

message queue is created by McZcOpenSender in advance

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McZcGetBuf

HANDLE McZcOpenSender (PCSTR *source*)

Brief

create message-sending queuqe

Parameters:

in	*source*	PCSTR - app (You) the sender

Return values:

Handle	for sending messages to a queue (If INVALID_HANDLE is returned its an error)
NULL	NULL pointer(fail to get)

Preconditons

none

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McZcSetParam, McZcGetBuf, McZcSend, McZcClose

E_CWORD33_Status McZcSend (HANDLE *hMessage*)

Brief

send message

Parameters:

in	*hMessage*	HANDLE - handle to the sendMessage, McZcOpenSender

Return values:

E_CWORD33_Status	indicates if the message was sent successfully
e_CWORD33_StatusOK	success
e_CWORD33_StatusInvldHandle	- If hMessage is NULL or invalid
e_CWORD33_StatusInvldBuf	- buffer is invalid
e_CWORD33_StatusInvldHndlType	-handle type is invalid
e_CWORD33_StatusMsgQFull	- message queue is full
e_CWORD33_StatusErrNoEBADF	- file descriptor invalid
e_CWORD33_StatusErrNoEINTR	-system call(signal) interupt
e_CWORD33_StatusInvldBufSize	-invalid buffer size

Preconditons

message queuqe is created by McZcOpenSender

Change of internal status

none

Classification

Public

Type

Method

See also:

McZcOpenSender, McZcSetParam, McZcGetBuf

E_CWORD33_Status McZcSetParam (HANDLE *handle*, UI_32 *cmd*, UI_32 *length*)

Brief

set variable parameter to messeg-sending header

Parameters:

in	*handle*	HANDLE - handle to the send message queue, McZcOpenSender
in	*cmd*	UI_32 -cmd for sending(ID to identify protocol of service)
in	*length*	UI_32 -size of buffer for message-sending

Return values:

E_CWORD33_Status	indicates if the message was set successfully
e_CWORD33_StatusOK	sucess
e_CWORD33_StatusInvldHandle	- If hMessage is NULL or invalid
e_CWORD33_StatusInvldBuf	- invalid buffer
e_CWORD33_StatusInvldBufSize	- invalid buffer size

Preconditons

message queue is created by McZcOpenSender in advance

Change of internal status

none

Classification

Public

Type

Sync only

See also:

McZcGetBuf, McZcSend

UI_64 MSToNS (UI_32 *ms*)

Brief

Converts mil seconds to nano seconds.

Parameters:

in	*ms*	UI_32 - time value in mil seconds

Return values:

Number	of nano seconds there are in mil seconds in the number passed.

Preconditons

Change of internal status

Classification

Type

See also:

none

HANDLE NS_TimerCreate (NSTimerInfo *timer_info*, eNSTimerCallbackMechanism *cbMech*, HANDLE *sndMqHndl*)

Brief

to create an instance of a timer object, and start it

Parameters:

in	*timer_info*	NSTimerInfo - timer value represents an absolute expiration timeout
in	*cbMech*	eNSTimerCallbackMechanism - CALLBACKMECHANISM_ENUM - specifies the callback mechanism associated with the timer CALLBACK_MESSAGE will send a message to a message queue

in	*sndMqHndl*	HANDLE - implies a MESSAGE QUE HANDLE generated by McOpenSender() is specified

Return values:

Handle	to the timer object or NULL on failure

Preconditons

Change of internal status

Classification

Type

See also:

NS_TimerDelete,NS_TimerSetTime

void NS_TimerDebugOn (BOOL *FlagState*)

Brief

to make a flag for the debugging process. for debug.

Parameters:

in	*FlagState*	BOOL - DebugFlag state

Return values:

none	

Preconditons

Change of internal status

Classification

Type

See also:

none

E_CWORD33_Status NS_TimerDelete (HANDLE *hTimer*)

Brief

to delete an instance of a timer object

Parameters:

in	*hTimer*	HANDLE - a timer handle that was created via NS_TimerCreate

Return values:

E_CWORD33_Status	indicates if the timer handle was delete successfully
e_CWORD33_StatusOK	
e_CWORD33_StatusFail	
e_CWORD33_StatusInvldParam	

Preconditons

Change of internal status

Classification

Type

See also:

NS_TimerCreate, NS_TimerSetTime

E_CWORD33_Status NS_TimerGetTime (HANDLE *hTimer*, NSTimerInfo * *timer_info*)

Brief

to retrieve the time until a timer expires

Parameters:

in	*hTimer*	HANDLE - handle to timer that your trying to get information about
out	*timer_info*	NSTimerInfo* - a structure to store the delay time until the timer expires

Return values:

E_CWORD33_Status	indicates if the timer info was get successfully
e_CWORD33_StatusOK	
e_CWORD33_StatusInvldHandle	
e_CWORD33_StatusFail	

Preconditons

Change of internal status

Classification

Type

See also:

NS_TimerCreate, NS_TimerSetTime

E_CWORD33_Status NS_TimerSetTime (HANDLE *hTimer*, NSTimerInfo *timer_info*)

Brief

to start or stop a timer by setting the time interval associated with a timer, value of 0 in t_sec & t_nsec, will stop the timer.

Parameters:

in	*hTimer*	HANDLE - handle to timer that your specifying the timer to be set
in	*timer_info*	NSTimerInfo - timer value represents an absolute expiration timeout

Return values:

E_CWORD33_Status	indicates if the timer info was set successfully
e_CWORD33_StatusOK	
e_CWORD33_StatusInvldHandle	
e_CWORD33_StatusFail	

Preconditons

the timer must have already been created with NS_TimerCreate

Change of internal status

Classification

Type

See also:

NS_TimerCreate, NS_TimerGetTime

VOID NsForceClose (void)

Brief

clear log forcely.

Return values:

none	return void

Prerequisite

none

Change of internal state

none

Classification

Public

Type

No match

See also:

none

void NsLog (const UI_16 *p_lLine_i*, const UI_16 *f_uiZoneIndex*, PCSTR *p_pstrClassName_i*, PCSTR *lpszFormat*, ...)

Brief

Writes a log to the Native Services logging mechanism

Parameters:

in	*p_lLine_i*	const UI_16 - Line number(065535) that this log was was written on
in	*f_uiZoneIndex*	const UI_64 - Mask value(0511) to check against
in	*p_pstrClassName_i*	PCSTR - Name of the class or method where the log was written
in	*lpszFormat*	PCSTR - string similar to that of a printf statement
in	*arg*	list ... - Data arguments(option)

Return values:

none	void - there is no return.

Prerequisite

_CWORD33_LOG parameter is set throuth NsLogSet_CWORD33_LogParams

Change of internal state

None

Classification

Public

Type

No match

See also:

none

void NsLog0 (const UI_16 *p_lLine_i*, const UI_16 *f_uiZoneIndex*, PCSTR *p_pstrClassName_i*, PCSTR *lpszFormat*)

Brief

Writes a log to the Native Services logging mechanism

Parameters:

in	*p_lLine_i*	const UI_16 - Line number(065535) that this log was was written on

in	*f_uiZoneIndex*	const UI_64 - Mask value(0511) to check against
in	*p_pstrClassName_i*	PCSTR - Name of the class or method where the log was written
in	*lpszFormat*	PCSTR - string similar to that of a printf statement
in	*arg*	list ... - Data arguments(option)

Return values:

none	void - there is no return.

Prerequisite

_CWORD33_LOG parameter is set throuth NsLogSet_CWORD33_LogParams

Change of internal state

None

Classification

Public

Type

Sync only

See also:

none

void NsLog_Cnt (UI_16 *Cnt_Id*, UI_8 *nu*, ...)

Brief

Counter logger mechanism.(Send data to SS event logger Queue)

Parameters:

in	*Cnt_Id*	UI_16 - ID of Counter from comm layer
in	*nu*	UI_8 - number of variable arguments inputted
in	...	... - Data arguments

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

none

Classification

Public

Type

Fire and Forget

See also:

none

void NsLog_Evt (UI_16 *Evt_Id*, UI_8 *nu*, ...)

Brief

Event logger mechanism.(Send data to SS event logger Queue)

Parameters:

in	*Evt_Id*	UI_16 - ID of Event from comm layer
in	*nu*	UI_8 - number of variable arguments inputted
in	...	... - Data arguments

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

none

Classification

Public

Type

Fire and Forget

See also:

none

void NsLog_EvtCnt (UI_16 *Cnt_Id*, UI_16 *Evt_Id*, UI_8 *n*, ...)

Brief

Event and count logger.

Parameters:

in		

void NsLogData (const UI_16 *f_uiZoneIndex*, PCSTR *data*, UI_32 *size*)

Brief

log print-out function

Parameters:

in	*f_uiZoneIndex*	const UI_64 - Mask value(0511) to check against
in	*data*	PCSTR -log print-out data
in	*size*	(0ULONG_MAX) UI_32 - data size (0ULONG_MAX)

Return values:

none	void - there is no return.

Prerequisite

_CWORD33_LOG has already set by NsLogSet_CWORD33_LogParams

Change of internal state

None

Classification

Public

Type

No match

See also:

none

UI_8 NsLogDetermineLogMethod (PCSTR *output_type*)

Brief

Determines the control log method based on the string passed. If the string passed doesn't match any of the key words: slogger or msgr or console the method will return a value of zero. This string may be in the form of the following: "msgq|console" would cause the output to go to both the NSLog message queue and the console.

Parameters:

in	*output_type*	PCSTR - The key words:

Return values:

1	... 0xF Success
0	the string passed doesn't match any of the key words

Prerequisite

none

Change of internal state

none

Classification

Public

Type

No match

See also:

none

VOID NSLogEnablePLog (BOOL *f_bEnable*)

NSLogEnablePLog Enables/Disables the performance logging

Parameters:

in	*f_bEnable*	BOOL - TRUE- Enable performance logging, FALSE - Disable

Returns:

none

VOID NSLogEnableSysEventLog (BOOL *f_bEnable*)

NSLogEnableSysEventLog Enables/Disables the system event logging

Parameters:

in	*f_bEnable*	BOOL - TRUE- Enable system event logging, FALSE - Disable

Returns:

none

PCSTR NSLogGet_CWORD33_logFileName (UI_32 *index*)

Brief

get the name of log file

Parameters:

in	*index*	UI_32 - Index of a list of _CWORD33_log file names.(0 NSLogGet_CWORD33_logFileTotalNum()-1)

Return values:

file	name -file with specific index is exsisted
NULL	-file with specific index is not exsisted

Prerequisite

set log file using NsLogSet_CWORD33_LogParams

Change of internal state

none

Classification

Public

Type

No macth

See also:

none

UI_32 NSLogGet_CWORD33_logFileTotalNum (void)

Brief

get total number of log file

Return values:

num	UI_32 - Total numger of _CWORD33_log files.

Prerequisite

set log file using NsLogSet_CWORD33_LogParams

Change of internal state

none

Classification

Public

Type

No match

See also:

none

E_CWORD33_Status NsLogGet_CWORD33_LogFlag (UI_8 *flag_id*, UI_8 * *mode*)

Brief

get LogLevel of _CWORD33_LOG flag

Parameters:

in	*flag_id*	UI_8 - index for _CWORD33_LOG flag(0FLAG_LIST(declared in _CWORD33_LOG CONFIG file)
out		

int NSLogGet_CWORD33_logIndex (PCSTR *filename*)

Brief

get index of log file

Parameters:

in	*filename*	PCSTR - Name of _CWORD33_log file.

Return values:

0	log file specific logfile is not exsisted
index	of log file specific logfile is exsisted

Prerequisite

set log file using NsLogSet_CWORD33_LogParams

Change of internal state

none

Classification

Public

Type

No match

See also:

none

void NsLogGetControlMask (T_CWORD33_ZoneMask *p_Zonemask_i*)

Brief

Get the control word that you currently have set.

Parameters:

out	*p_Zonemask_i*	T_CWORD33_ZoneMask - current mask value.

T_CWORD33_ZoneMask Array

```
1 typedef T_CWORD33_Zone T_CWORD33_ZoneMask[ZONE_MASK_ARRAY_ELTS];
```

T_CWORD33_Zone typedef

```
1 UI_32 T_CWORD33_Zone;  // 32bit
```

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

Classification

Public

Type

No match

See also:

NsLogSetControlMask

UI_8 NsLogGetLogMethod (void)

Brief

Get the logging method.

Parameters:

None	

Return values:

-	return the way you log your data. UI_8 - Log data typeNS_LoggingMethod_t Enum.

NS_LoggingMethod_t Enum

```
1 typedef enum _NS_LOGGER_METHODS {
2   LPRINT      = ***,       // Print to the console(similar to printf)
3   LMSGQ       = ***,       // Disabled
4   LSLOGGER    = ***,       // Disabled
5   LSHAREDMEM  = ***        // Output to shared memory. this is the default logging method.
6  } NS_LoggingMethod_t;
```

Prerequisite

none

Change of internal state

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

NsLogSetLogMethod

void NsLogGetRealtimeLog (UI_8 * *mode*)

Brief

get the setting of real-time log output

Parameters:

out	*mode*	UI_8 * -the setting of real-time log output

available settings of real-time log output default value is REALTIME_LOG in _CWORD33_LOG CFG file (_CWORD33_LOG_REALTIMELOG_MODE_OFF when _CWORD33_LOG CFG file get failed)

```
1 #define _CWORD33_LOG_REALTIMELOG_MODE_UART       1     // : UARTOUT
```

```
2 #define _CWORD33_LOG_REALTIMELOG_MODE_USB        2     // : USB OUT
3 #define _CWORD33_LOG_REALTIMELOG_MODE_USB_DISABLE 0x82  // : USB OFF
4 #define _CWORD33_LOG_REALTIMELOG_MODE_ETHER      3     // : Ether OUT
5 #define _CWORD33_LOG_REALTIMELOG_MODE_OFF        0     // : ALL OFF
6 #define _CWORD33_LOG_REALTIMELOG_MODE_FREEZE     0xFF  // : FREEZE
```

Return values:

none	void - there is no return

Prerequisite

none

Change of internal state

none

Classification

Public

Type

Sync only

See also:

NsLogSetRealtimeLog

T_CWORD33_LoggerSeverity NsLogGetSeverity (void)

Brief

Get the log severity.

Parameters:

none	

Return values:

Log	Severity T_CWORD33_LoggerSeverity - Log severity.

T_CWORD33_LoggerSeverity Enum

```
1 typedef enum _T_CWORD33_LoggerSeverity {
2   _CWORD33__SEVERITY_LOW  = ***,  /// <all enable(default)
3   _CWORD33__SEVERITY_DEBUG2,    /// <enable _CWORD33_LOG_DEBUG() and _CWORD33_LOG_DEBUG2()
and all the follows
4   _CWORD33__SEVERITY_DEBUG1,    /// <enable _CWORD33_LOG_DEBUG1() and all the follows
5   _CWORD33__SEVERITY_INFO,      /// <enable _CWORD33_LOG_INFO() and all the follows
6   _CWORD33__SEVERITY_WARN,      /// <enable _CWORD33_LOG_WARN() and all the follows
7   _CWORD33__SEVERITY_ERROR,     /// <enable _CWORD33_LOG_ERROR() and all the follows
8   _CWORD33__SEVERITY_FATAL,     /// <enable _CWORD33_LOG_FATAL() and _CWORD33_LOG_ALWAYS()
9   _CWORD33__SEVERITY_ALWAYS     /// <enable _CWORD33_LOG_ALWAYS()
10  } T_CWORD33_LoggerSeverity;
```

Prerequisite

none

Change of internal state

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

NsLogSetSeverity

VOID NsLogGetZoneTextList (CHAR *f_cZoneList*[][ZONE_TEXT_SIZE])

PUI_32 - CHAR cZones[USER_ZONES_COUNT][ZONE_TEXT_SIZE]

Brief

Get the zone data list.

Parameters:

out	*f_cZoneList*	CHAR [][ZONE_TEXT_SIZE] - zone text list(0 ... 511)

PUI_32 - Pointer to preallocated array of CHAR cZones[USER_ZONES_COUNT][ZONE_TEXT_SIZE];

Return values:

none	VOID - There is no return

Prerequisite

none

Change of internal state

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

NsLogSetZones

void NsLogInitialize (void)

Brief

Initialize the log level and real-time log out-put setting

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

none

Classification

Public

Type

No match

See also:

none

BOOL NsLogIsPLogEnabled (void)

Brief

Is Enables/Disables the performance logging

Return values:

TRUE	current state of plog enable.
FALSE	current state of plog disable.

Prerequisite

Change of internal state

Classification

Public

Type

No match

See also:

NSLogEnablePLog

BOOL NSLogIsSysEventLogEnabled (void)

NSLogIsSysEventLogEnabled Is Enables/Disables the system event logging

Returns:

SysEventLogEnabled BOOL - current state of sys event log(enable/disable).

BOOL NsLogIsZoneSet (UI_32 *set_zone*)

Brief

check wheather the _CWORD33_LOG ZONE is set by your application process.

Parameters:

in	*set_zone*	UI_32 - checking zone_index

Return values:

TRUE	zone is enable
FALSE	zone is disable

Prerequisite

none

Change of internal state

Classification

Public

Type

No match

See also:

none

VOID NSLogPrintPerformanceLog (const UI_16 *f_ui16Line*, PCSTR *f_cFuncName*, PCSTR *__format*, ...)

Brief

This macro API is disable in Linux system

Parameters:

in	*f_ui16Line*	const UI_16 - line number
in	*f_cFuncName*	PCSTR - Function name. It takes **PRETTY_FUNCTION**
in	*print_fmt*	PCSTR - Input string

Return values:

none	

Prerequisite

Change of internal state

Classification

Public

Type

No match

See also:

none

void NsLogSet_CWORD33_LogFlag (UI_8 *flag_id*, UI_8 *mode*)

Brief

set LogLevel of _CWORD33_LOG flag

Parameters:

in	*flag_id*	UI_8 - index for _CWORD33_LOG flag(0FLAG_LIST(declared in _CWORD33_LOG CONFIG file)
in	*mode*	UI_8 - LogLevel

avaliable LogLevel
_CWORD33_LOG_FLAG_MODE_RELEASE
_CWORD33_LOG_FLAG_MODE_DEBUG

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

none

Classification

Public

Type

Sync only

See also:

NsLogGet_CWORD33_LogFlag

void NsLogSet_CWORD33_LogParams (_CWORD33_LOGPARAM * *p__CWORD33_LogParams*)

Brief

Sets the _CWORD33_LOG parameter of your application based on the value passed in.

Parameters:

in	*p__CWORD33_LogParams*	_CWORD33_LOGPARAM * - _CWORD33_LOG parameter in process

_CWORD33_LOGPARAM Structure

```
1 typedef struct __CWORD33_LOGPARAM {
2   UI_8 uiLogOptions;                          // output options (ref. NS_LoggingMethod_t )
3   CHAR cZones[USER_ZONES_COUNT][ZONE_TEXT_SIZE];  // mask text ZONE_TEXT_10 ... ZONE_TEXT_n
4   // mask options ZONEMASK(10)...ZONEMASK(n) (ref. T_CWORD33_Zone)
5   T_CWORD33_ZoneMask uiZoneMask;
6   T_CWORD33_LoggerSeverity eSeverity;                 // severity level(ref. T_CWORD33_LoggerSeverity)
7  } _CWORD33_LOGPARAM, *LP_CWORD33_LOGPARAM;
```

T_CWORD33_ZoneMask Array

```
1 typedef T_CWORD33_Zone T_CWORD33_ZoneMask[ZONE_MASK_ARRAY_ELTS];
```

T_CWORD33_Zone typedef

```
1 UI_32 T_CWORD33_Zone;  // 32bit
```

T_CWORD33_LoggerSeverity Enum

```
1 typedef enum _T_CWORD33_LoggerSeverity {
2   _CWORD33__SEVERITY_LOW  = ***,  // all enable(default)
3   _CWORD33__SEVERITY_DEBUG2,      // enable _CWORD33_LOG_DEBUG() and _CWORD33_LOG_DEBUG2()
and all the follows
4   _CWORD33__SEVERITY_DEBUG1,      // enable _CWORD33_LOG_DEBUG1() and all the follows
5   _CWORD33__SEVERITY_INFO,        // enable _CWORD33_LOG_INFO() and all the follows
6   _CWORD33__SEVERITY_WARN,        // enable _CWORD33_LOG_WARN() and all the follows
7   _CWORD33__SEVERITY_ERROR,       // enable _CWORD33_LOG_ERROR() and all the follows
8   _CWORD33__SEVERITY_FATAL,       // enable _CWORD33_LOG_FATAL() and _CWORD33_LOG_ALWAYS()
9   _CWORD33__SEVERITY_ALWAYS       // enable _CWORD33_LOG_ALWAYS()
10 } T_CWORD33_LoggerSeverity;
```

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

none

Classification

Public

Type

No match

See also:

none

void NsLogSetControlMask (T_CWORD33_ZoneMask *p_ulNSLogControl_i*)

Brief

Sets the control word that should be logged

Parameters:

in	*p_ulNSLogControl_i*	T_CWORD33_ZoneMask - Set mask to the new value.

T_CWORD33_ZoneMask Array

```
1 typedef T_CWORD33_Zone T_CWORD33_ZoneMask[ZONE_MASK_ARRAY_ELTS];
```

T_CWORD33_Zone typedef

```
1 UI_32 T_CWORD33_Zone;  // 32bit
```

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

none

Classification

Public

Type

No match

See also:

NsLogGetControlMask

void NsLogSetLogMethod (UI_8 *p_eMethod_i*)

Brief

Sets the logging place, LPRINT, LSHAREDMEM or any combination of these. If this is never called the default is LSHAREDMEM.

Parameters:

in	*p_eMethod_i*	UI_8 - Where you would like to log your data.

NS_LoggingMethod_t Enum

```
1 typedef enum _NS_LOGGER_METHODS {
2   LPRINT      = ***,       // Print to the console(similar to printf)
3   LMSGQ       = ***,       // Disabled
4   LSLOGGER    = ***,       // Disabled
5   LSHAREDMEM  = ***        // Output to shared memory. this is the default logging method.
6  } NS_LoggingMethod_t;
```

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

-switch the out-put place for NsLog API

Classification

Public

Type

No match

See also:

NsLogGetLogMethod

VOID NSLogSetPlogOutputOptions (UI_8 *f_uiPlogOutputOption*)

NSLogSetPlogOutputOptions Set the output options for performance logging.

Parameters:

in	*f_uiPlogOutputOption*	UI_8 - Output option. Value is generated by or'ed value of enum EPLOG_OUTPUT_OPTION e.g. EPLOG_SLOG | EPLOG_MSGQ

Returns:

none

VOID NSLogSetPlogTimeFormat (EPLOG_TIME_FORMAT *f_ePlogTimeFormat*)

NSLogSetPlogTimeFormat Set the time format for performance logging.

Parameters:

in	*f_ePlogTimeFormat*	EPLOG_TIME_FORMAT - Time format for performance logging

Returns:

none

void NsLogSetProcessName (PCSTR *p_strProcessName_i*)

Brief

Sets the name of your application based on the value passed in.

Parameters:

in	*p_strProcessName_i*	PCSTR - name of your application process.(string array ended by NULL below 20byte)

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

none

Classification

Public

Type

No match

See also:

none

void NsLogSetRealtimeLog (UI_8 *mode*)

Brief

real-time log output setting

Parameters:

in	*mode*	UI_8 - set mode for Real time log

available setting for real-time log output default value is REALTIME_LOG in _CWORD33_LOG CFG file (_CWORD33_LOG_REALTIMELOG_MODE_OFF when _CWORD33_LOG CFG file get failed)

```
1 #define _CWORD33_LOG_REALTIMELOG_MODE_UART       1     // : UARTOUT
2 #define _CWORD33_LOG_REALTIMELOG_MODE_USB        2     // : USB OUT
3 #define _CWORD33_LOG_REALTIMELOG_MODE_USB_DISABLE 0x82  // : USB OFF
4 #define _CWORD33_LOG_REALTIMELOG_MODE_ETHER      3     // : Ether OUT
5 #define _CWORD33_LOG_REALTIMELOG_MODE_OFF        0     // : ALL OFF
6 #define _CWORD33_LOG_REALTIMELOG_MODE_FREEZE     0xFF  // : FREEZE
```

Return values:

none	void - there is no return

Prerequisite

none

Change of internal state

none

Classification

Public

Type

Sync only

See also:

NsLogGetRealtimeLog

void NsLogSetSeverity (T_CWORD33_LoggerSeverity *p_eLogSeverity_i*)

Brief

Set log severity level

Parameters:

in	*p_eLogSeverity_i*	T_CWORD33_LoggerSeverity - Log severity

T_CWORD33_LoggerSeverity Enum

```
1 typedef enum _T_CWORD33_LoggerSeverity {
2   _CWORD33__SEVERITY_LOW  = ***,  /// <all enable(default)
3   _CWORD33__SEVERITY_DEBUG2,    /// <enable _CWORD33_LOG_DEBUG() and _CWORD33_LOG_DEBUG2()
and all the follows
4   _CWORD33__SEVERITY_DEBUG1,    /// <enable _CWORD33_LOG_DEBUG1() and all the follows
5   _CWORD33__SEVERITY_INFO,      /// <enable _CWORD33_LOG_INFO() and all the follows
6   _CWORD33__SEVERITY_WARN,      /// <enable _CWORD33_LOG_WARN() and all the follows
7   _CWORD33__SEVERITY_ERROR,     /// <enable _CWORD33_LOG_ERROR() and all the follows
8   _CWORD33__SEVERITY_FATAL,     /// <enable _CWORD33_LOG_FATAL() and _CWORD33_LOG_ALWAYS()
9   _CWORD33__SEVERITY_ALWAYS     /// <enable _CWORD33_LOG_ALWAYS()
10  } T_CWORD33_LoggerSeverity;
```

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

The internal state is not changed.

Classification

Public

Type

Sync only

See also:

NsLogGetSeverity

void NsLogSetZones (UI_32 *f_uiZoneCount*, ...)

Brief

set _CWORD33_LOG ZONE

Parameters:

in	*f_uiZoneCount*	UI_32 - number of zone
in	...	arg_list - zone list (0 ... 511)

Return values:

none	void - there is no return.

Prerequisite

none

Change of internal state

none

Classification

Public

Type

Set Get

See also:

none

VOID NSLogSysEvent (const UI_16 *f_ui16Line*, PCSTR *f_cFuncName*, PCSTR *__format*, ...)

NSLogSysEvent() Prints system event log in message queue NSSyseventlog.

Parameters:

in	*f_ui16Line*	const UI_16 - line number
in	*f_cFuncName*	PCSTR - Function name. It takes **PRETTY_FUNCTION**
in	*__format*	PCSTR - Input string

Returns:

none

void NsLogTime (const UI_16 *f_uiZoneIndex*, PCSTR *lpszFormat*)

Brief

Print out real-time in log using Native Service mechanism

Parameters:

in	*f_uiZoneIndex*	const UI_64 - Mask value(0511) to check against
in	*lpszFormat*	PCSTR - string similar to that of a printf statement

Return values:

none	void - there is no return.

Prerequisite

parameter _CWORD33_LOG has already set using NsLogSet_CWORD33_LogParams

Change of internal state

None

Classification

Public

Type

No match

See also:

none

E_CWORD33_Status NSSharedMemDumpTransmitLogToFile (PCSTR *f_pPath*, PUI_32 *f_puiDumpSize*)

Brief

Dump the transmit logging shared memory to file

Parameters:

in	*f_pPath*	PCSTR - full file path
in	*f_puiSize*	PCSTR - Size of data dumped to file

Return values:

e_CWORD33_StatusOK	if shared memory is closed

e_CWORD33_StatusFileLoadError	file path is incorrect
e_CWORD33_StatusSemLockFail/	e_CWORD33_StatusSemUnLockFail - mutex locking/unlocking error
e_CWORD33_StatusFail	shared memory is not open

Preconditons

None

Change of internal status

-Open shared memory object

Classification

public

Type

sync only

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

None

SI_32 NSSharedMemReadTransmitLog (PSTR *f_pBuffer*, const UI_32 *f_uiLength*, const BOOL *f_bBlock*)

Brief

Reads transmit log from the shared memory buffer.

Parameters:

in	*f_pBuffer*	PSTR - pointer to the buffer in which data to be read.
in	*f_uiLength*	UI_32 - length of the data to be read.
in	*f_bBlock*	BOOL - TRUE - blocking call FALSE - non blocking call

Return values:

Except	NS_SHM_ERROR - Indicates the number of bytes read.
NS_SHM_ERROR	-if shared memory is not opened.

Preconditons

-None

Change of internal status

None

Classification

Public

Type

sync only

l_iReadSize:Result

NS_SHM_ERROR:Failure

Except NS_SHM_ERROR:Success

See also:

NSSharedMemWriteTransmitLog

E_CWORD33_Status NSSharedMemTransmitLogClose ()

Brief

Close the transmit logging shared memory

Parameters:

in		

BOOL NSSharedMemTransmitLogIsOpen ()

Brief

This function is used to check whether the shared memory buffer is opened or not.

Return values:

TRUE	- Open
FALSE	- Not open

Preconditons

None

Change of internal status

-None

Classification

public

Type

Sync only

See also:

none

E_CWORD33_Status NSSharedMemTransmitLogOpen ()

Brief

Open the shared memory for transmit logging.

Parameters:

in		

SI_32 NSSharedMemWriteTransmitLog (PCSTR *f_pBuffer*, const UI_32 *f_uiLength*)

Brief

Write transmit log into the shared memory buffer.

Parameters:

in	*f_pBuffer*	PSTR - pointer to the buffer containing the data to be written.
in	*f_uiLength*	UI_32 - length of the data buffer to be written.

Return values:

Except	NS_SHM_ERROR - Indicates the number of bytes written,
NS_SHM_ERROR	-if shared memory is not opened.

Preconditons

-None

Change of internal status

None

Classification

Public

Type

sync only

l_iReadSize:Result

NS_SHM_ERROR:Failure

Except NS_SHM_ERROR:Success

See also:

NSSharedMemReadTransmitLog

UI_32 RemainderMs (UI_32 *ms*)

Brief

Get the number of remaining milseconds out of whole second for the number passed.

Parameters:

in	*ms*	UI_32 - time value in mil seconds

Return values:

Number	of remaining mil seconds in the number passed.

Preconditons

Change of internal status

Classification

Type

See also:

none

TMemID SetDataToShared (const void * *data*, UI_32 *dataBytes*, const char * *from*, const char * *to*)

Summary

Store a large chunk of memory using SharedMem service

Parameters:

in	*data*	18 const void* - data to be stored in shared memory
in	*dataBytes*	19 UI_32 - length of the data (in bytes) that is stored in shared memory
in	*from*	20 const char * - start address
in	*to*	21 const char * - end address

Return values:

TMemID	- unique MemID of the data stored in the shared memory, 0 if error occurs

Preconditions

None

Change of the internal state

shared memory changed

Classification

Public

Type

sync only

TMemID:Result
Except 0:Success
0:Failure

See also:
None

time_t WholeSeconds (UI_32 *ms*)

Helper methods that convert time provided in MS. mseconds.

Brief
Get the number of whole seconds in the milsecond number that was passed.

Parameters:

in	*ms*	UI_32 - time value in mil seconds

Return values:

Number	of whole seconds in the number passed.

Preconditons

Change of internal status

Classification

Type

See also:
none

Utility

#define **_NSFW_MSG_LEN_** (40)
#define **_NSFW_SYSINFO_FLAG_** (4)
#define **_NSFW_SYSINFO_SIZE_** (64)
#define **NSFW_GET_MESSAGE**(msg)
#define **NTFY_LOGGER_SETCONTROLMASK** "LoggerService/SetControlMask"

Typedefs

typedef std::vector< std::string > TFileList
typedef TFileList::iterator TFileListIterator
typedef enum _COMMANDS COMMANDS
Logger Utility Commands.

Enumerations

enum _COMMANDS { **SET_LOG_MASK** = ***, **GET_LOG_MASK**, **SET_LOG_OUT_OPT**, **GET_LOG_OUT_OPT**, **GET_LOG_MASK_RESPONSE**, **GET_LOG_OUT_OPT_RESPONSE**, **SET_MSGTX_RX_ON_AND_TX_OFF**, **SET_MSGTX_RX_OFF_AND_TX_ON**, **SET_MSGTX_RX_AND_TX_ON**, **SET_MSGTX_RX_AND_TX_OFF**, **kDebugDumpRequest**, **SET_LOG_SEVERITY**, **GET_LOG_SEVERITY**, **GET_LOG_SEVERITY_RESPONSE**, **NSRCS_SET_LOG_SETTINGS_REQ**, **NSRCS_SET_LOG_SETTINGS_RESP**, **NSRCS_GET_LOG_SETTINGS_REQ**, **NSRCS_GET_LOG_SETTINGS_RESP**, **NSRCS_SET_LOG_MASK_REQ**, **NSRCS_SET_LOG_MASK_RESP**, **NSRCS_SET_LOG_OUT_OPT_REQ**, **NSRCS_SET_LOG_OUT_OPT_RESP**, **NSRCS_SET_LOG_SEVERITY_REQ**, **NSRCS_SET_LOG_SEVERITY_RESP** }*Logger Utility Commands.*

Functions

UI_16 ConvertEndian_UI16 (UI_16 f_value)
SI_16 ConvertEndian_SI16 (SI_16 f_value)
UI_32 ConvertEndian_UI32 (UI_32 f_value)
SI_32 ConvertEndian_SI32 (SI_32 f_value)
UI_64 ConvertEndian_UI64 (UI_64 f_value)
SI_64 ConvertEndian_SI64 (SI_64 f_value)
E_CWORD33_Status CalculateCRC16 (PCSTR f_c_file_name, UI_16 &f_ui_check_sum)
E_CWORD33_Status CalculateCRC32 (PCSTR f_c_file_name, UI_32 &f_ui_check_sum)
E_CWORD33_Status GetFileList (TFileList *f_pv_tfile_list, PCSTR f_pc_path)
BOOL DoesDirecotryExist (std::string f_c_dir_path)
E_CWORD33_Status CreateDirectory (std::string f_c_dir_path)

Detailed Description

Macro Definition Documentation

#define NSFW_GET_MESSAGE(msg)

Value:((*((unsigned int *)msg) & _NSFW_SYSINFO_FLAG_) ? \
((unsigned char *)msg) + _NSFW_MSG_LEN_ + _NSFW_SYSINFO_SIZE_ : \
((unsigned char *)msg) + _NSFW_MSG_LEN_)

Typedef Documentation

typedef std::vector< std::string > TFileList

an abbreviation for std::vector<std::string>

typedef TFileList::iterator TFileListIterator

a abbreviation for std::vector<std::string>::iterator

Function Documentation

E_CWORD33_Status CalculateCRC16 (PCSTR *f_c_file_name*, UI_16 & *f_ui_check_sum*)

Brief

Calculate the CRC value of the file

Parameters:

in	*f_c_file_name*	PCSTR - the file to be calculated
out	*f_ui_check_sum*	UI_16 - the CRC value of the file

Return values:

e_CWORD33_StatusOK	: Success
e_CWORD33_StatusFileLoadError	: File Load Error
e_CWORD33_StatusInvldParam	: Invalid Param

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

E_CWORD33_Status CalculateCRC32 (PCSTR *f_c_file_name*, UI_32 & *f_ui_check_sum*)

Brief

Calculate the CRC value of the file

Parameters:

in	*f_c_file_name*	PCSTR - the file to be calculated
out	*f_ui_check_sum*	UI_32 - the CRC value of the file

Return values:

e_CWORD33_StatusOK	: Success
e_CWORD33_StatusFileLoadError	: File Load Error
e_CWORD33_StatusInvldParam	: Invalid Param

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

SI_16 ConvertEndian_SI16 (SI_16 *f_value*)

Brief

Convert the SI_16(signed 16bit) value to big-endian.

Parameters:

in	*f_value*	SI_16 - The value to be converted

Return values:

the	big-endian value of the input param f_value

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

SI_32 ConvertEndian_SI32 (SI_32 *f_value*)

Brief

Convert the SI_32(signed 32bit) value to big-endian.

Parameters:

in	*f_value*	SI_32 - The value to be converted

Return values:

the	big-endian value of the input param f_value

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

SI_64 ConvertEndian_SI64 (SI_64 *f_value*)

Brief

Convert the SI_64(signed 64bit) value to big-endian.

Parameters:

in	*f_value*	SI_64 - The value to be converted

Return values:

the	big-endian value of the input param f_value

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

UI_16 ConvertEndian_UI16 (UI_16 *f_value*)

Brief

Convert the UI_16(unsigned 16bit) value to big-endian.

Parameters:

in	*f_value*	UI_16 - The value to be converted

Return values:

the	big-endian value of the input param f_value

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

UI_32 ConvertEndian_UI32 (UI_32 *f_value*)

Brief

Convert the UI_32(unsigned 32bit) value to big-endian.

Parameters:

in	*f_value*	UI_32 - The value to be converted

Return values:

the	big-endian value of the input param f_value

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

UI_64 ConvertEndian_UI64 (UI_64 *f_value*)

Brief

Convert the UI_64(unsigned 64bit) value to big-endian.

Parameters:

in	*f_value*	UI_64 - The value to be converted

Return values:

the	big-endian value of the input param f_value

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

E_CWORD33_Status CreateDirectory (std::string *f_c_dir_path*)

Brief

Create the directory f_c_dir_path.

Parameters:

in	*f_c_dir_path*	std::string - the directory to be created

Return values:

e_CWORD33_StatusOK	: Success
e_CWORD33_StatusFail	: Abnormal Error

Prerequisite

None

Change of internal state

Classification

Public

Type

Sync Only

See also:

BOOL DoesDirecotryExist (std::string *f_c_dir_path*)

Brief

Check whether the directory is existing or not.

Parameters:

in	*f_c_dir_path*	std::string - the directory to be checked

Return values:

TRUE	the directory is existing. FALSE: the directory is not existing.

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

E_CWORD33_Status GetFileList (TFileList * *f_pv_tfile_list*, PCSTR *f_pc_path*)

Brief

Get the name of files existing in a specific directory

Parameters:

in	*f_pc_path*	PCSTR - the directory path
out	*f_pv_tfile_list*	TFileList - store the name of the files in the specific directory

Return values:

e_CWORD33_StatusOK	: Success
e_CWORD33_StatusInvldParam	: Invalid Param
e_CWORD33_StatusNullPointer	: NULL Pointer
e_CWORD33_StatusFail	: Abnormal Error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

Nsrcs

class CPassThruInDataHandler

Class: CPassThruInDataHandler. class CPassThruOutDataHandler

Data Handler. struct _TNSRCS_SetLogSettingsReq

struct _TNSRCS_SetLogSettingsResp
struct _TNSRCS_GetLogSettingsReq
struct _TNSRCS_GetLogSettingsResp
struct _TNSRCS_SetLogMaskResp
struct _TNSRCS_SetLogOutOptResp
struct _TNSRCS_SetLogSeverityResp
class CNSRcsPlugin

***CNSRcsPlugin*. Macros**

#define **SIZE_OF_PASSTHRU_DATATYPE_IDENTIFIER** 1
#define **SIZE_OF_PASSTHRU_PARAM_BUFFERTYPE_HEADER** 3

#define **PASSTHRU_DATA_HEADER_LEN** 8

Typedefs

typedef UI_8 **NSRCS_BOOL**
typedef struct _TNSRCS_SetLogSettingsReq **TNSRCS_SetLogSettingsReq**
typedef struct _TNSRCS_SetLogSettingsResp **TNSRCS_SetLogSettingsResp**
typedef struct _TNSRCS_GetLogSettingsReq **TNSRCS_GetLogSettingsReq**
typedef struct _TNSRCS_GetLogSettingsResp **TNSRCS_GetLogSettingsResp**
typedef struct _TNSRCS_SetLogMaskResp **TNSRCS_SetLogMaskResp**
typedef struct _TNSRCS_SetLogOutOptResp **TNSRCS_SetLogOutOptResp**
typedef struct _TNSRCS_SetLogSeverityResp **TNSRCS_SetLogSeverityResp**
typedef E_CWORD78_Status(* **TFPNSRcsSendPassthruData**) (UI_8 f_ui8clientId, PVOID f_data, UI_16 f_ui16PayloadLength)

Detailed Description

Class Documentation

struct _TNSRCS_SetLogSettingsReq

Class Members:

T_CWORD33_ZoneMask	m_pui32ZoneMask	
T_CWORD33_LoggerSeverity	m_si32Severity	
UI_8	m_ui8ClientId	
UI_8	m_ui8OutputLogOption	

struct _TNSRCS_SetLogSettingsResp

Class Members:

T_CWORD33_ZoneMask	m_pui32ZoneMask	
T_CWORD33_LoggerSeverity	m_si32Severity	
UI_8	m_ui8ClientId	
UI_8	m_ui8OutputLogOption	
UI_8	m_ui8SetStatus	

struct _TNSRCS_GetLogSettingsReq

Class Members:

UI_8	m_ui8ClientId	

struct _TNSRCS_GetLogSettingsResp

Class Members:

CHAR	m_cZoneList[BITS_IN_ZONE_MASK][ZONE_TEXT_SIZE]	
T_CWORD33_ZoneMask	m_pui32ZoneMask	
T_CWORD33_LoggerSeverity	m_si32Severity	
UI_8	m_ui8ClientId	
UI_8	m_ui8OutputLogOption	

struct _TNSRCS_SetLogMaskResp

Class Members:

T_CWORD33_ZoneMask	m_pui32ZoneMask	
UI_8	m_ui8ClientId	
UI_8	m_ui8SetStatus	

struct _TNSRCS_SetLogOutOptResp

Class Members:

UI_8	m_ui8ClientId	
UI_8	m_ui8OutputLogOption	
UI_8	m_ui8SetStatus	

struct _TNSRCS_SetLogSeverityResp

Class Members:

SI_32	m_si32Severity	
UI_8	m_ui8ClientId	
UI_8	m_ui8SetStatus	

Namespace Documentation

CWORD33 Namespace Reference

Namespaces

ns

Detailed Description

the namespace *CWORD33*

CWORD33::ns Namespace Reference

Namespaces

utility

Detailed Description

the namespace ns

CWORD33::ns::utility Namespace Reference

Namespaces

utility_private

Functions

BOOL buildVersionsMatch (PCSTR build_version)
PCSTR getEnvironmentBuildVersion ()
PCSTR getLibraryBuildVersion ()

Detailed Description

the namespace utility

Function Documentation

BOOL _CWORD33_::ns::utility::buildVersionsMatch (PCSTR *build_version*)

Brief

Check whether the input param build_version is matched or not.

Parameters:

in	*build_version*	PCSTR - The version to be checked

Return values:

TRUE	match; FALSE: does not match

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

getEnvironmentBuildVersion getLibraryBuildVersion

PCSTR _CWORD33_::ns::utility::getEnvironmentBuildVersion ()

Brief

Get the build version from environment variable.

Return values:

the	build version string

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

PCSTR _CWORD33_::ns::utility::getLibraryBuildVersion ()

Brief

Get the library build version string

Return values:

the	library buidling version string

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

_BUILD_VERSION

CWORD33::ns::utility::utility_private Namespace Reference

Functions

PCSTR niceBuildVersion (PCSTR build_version)

Detailed Description

the namespace utility_private

Function Documentation

PCSTR _CWORD33_::ns::utility::utility_private::niceBuildVersion (PCSTR *build_version*)

Brief

Check whether the input param is empty or not.

Parameters:

in	*build_version*	PCSTR - the build version

Return values:

the	input param build_version or "<Not defined>"

Prerequisite

NULL

Change of internal state

NULL

Classification

Public

Type

Sync Only

See also:

Class Documentation

Accumulator< T > Class Template Reference

Accumultor type.

#include <ns_utility.hpp>

Public Member Functions

Accumulator (T n)
template<typename U > **Accumulator** (const Accumulator< U > &u)
T **operator()** (T i=1)
template<typename U > T **operator()** (U i)

Friends

template<typename U > class **Accumulator**

Detailed Description

template<typename T>

class Accumulator< T >

Accumultor type.

An accumulator is an object that keeps an internal counter, and will increment its counter by n, which defaults to 1, each time operator() is called on it. Eample usage:

Accumulator< int > a(42); // initialize a to 42 int v1 = a(); // < v1 == *** int v2 = a(); // < v2 == *** int v4 = a(10); // < v4 == *** int v5 = a(); // < v5 == ***

The documentation for this class was generated from the following file:
22 ns_utility.hpp

CWORD33::framework::args Class Reference

Public Types

enum { **name**, **priority** }

Member Enumeration Documentation

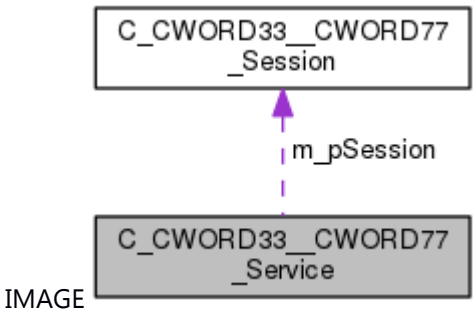
anonymous enum

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

23 _CWORD78__thread_priority.h

C_CWORD33__CWORD77_Service Class Reference

#include <_CWORD78___CWORD77__service_if.h>
Collaboration diagram for C_CWORD33__CWORD77_Service:



IMAGE

Public Member Functions

C_CWORD33__CWORD77_Service ()
virtual ~C_CWORD33__CWORD77_Service ()
virtual E_CWORD33_Status HandleServiceMessage (UI_32 cmdID)=0
E_CWORD33_Status OpenServiceOnAvailability (HANDLE f_hApp, const std::string &f_sServiceName)
E_CWORD33_Status CloseServiceOnUnavailability (HANDLE hApp)
void SetResponseCallback (ResponseServiceTo_CWORD77_ fptr)
void SetSession (C_CWORD33__CWORD77_Session *f_pSessionObj)
E_CWORD33_Status SubscribeNotifications (HANDLE hApp)
E_CWORD33_Status UnSubscribeNotifications (HANDLE hApp)
virtual void EnableSession (UI_32 f_uiNum, UI_32 f_uiFirst,...)=0
E_CWORD33_Status OnServiceNtf (HANDLE hApp)
void OpenSessionAcks (HANDLE f_hApp)
void Set_CWORD77_OpenSessionACK (SessionAckTo_CWORD77_ fptr)

Protected Member Functions

void SendMessageToSession (UI_32 f_uiSessionType, UI_32 f_uiSrvProtocol)

application framework handle

void OpenSession (HANDLE f_hApp, const std::string &f_sServiceName)
void CloseSession (HANDLE f_hApp)
void AddNotification (PCSTR f_pNotification)
virtual E_CWORD33_Status AttachOpenSessionAckCallback (HANDLE hApp)=0
void OpenSessionRetry (UI_32 f_ui32SessionType)

Protected Attributes

std::map< UI_32, C_CWORD33__CWORD77_Session * > m_SessTypeToSessObj
Map of session type to session object.

ResponseServiceTo_CWORD77_ m_cbResponse

Call back function pointer.

SessionAckTo_CWORD77_ **m_cbSessionACK**

std::vector< _CWORD33_NotificationCallbackHandler > m_vServiceNotifications

vector of _CWORD33_notification callbacks

C_CWORD33__CWORD77_Session * m_pSession

Pointer to an instance of I_CWORD77_Session.

std::string **m_cServiceName**

BOOL **m_bServiceAvailable**

UI_8 **m_ui8SessionRetryCount**

HANDLE **m_hApp**

Detailed Description

class: C_CWORD33__CWORD77_Service Description: This is interface class for all services classes.

Constructor & Destructor Documentation

C_CWORD33__CWORD77_Service::C_CWORD33__CWORD77_Service ()

Brief

Constructor for C_CWORD33__CWORD77_Service

Return values:

none	

Preconditions

-No preconditions

Change of internal status

-The internal state is not changed.

Classification

-public

type

-None

See also:

~C_CWORD33__CWORD77_Service

virtual C_CWORD33__CWORD77_Service::~C_CWORD33__CWORD77_Service ()[virtual]

Brief

Destructor for C_CWORD33__CWORD77_Service

Return values:

none	

Preconditons

-C_CWORD33__CWORD77_ServiceinstanceisdeclaredinConstructor

Change of internal status

-The internal status is not changed

Classification

-Pubulic

Type

-None

See also:

C_CWORD33__CWORD77_Service

Member Function Documentation

void C_CWORD33__CWORD77_Service::AddNotification (PCSTR *f_pNotification*)[protected]

Brief

Function to add notification

Parameters:

in	*f_pNotification*	PCSTR - Notification name

Return values:

none	

Preconditons

SessionobejectisdeclaredinC_CWORD33__CWORD77_Service::SetSession()

Change of internal status

The internal state is not changed.

Classification

-public

Type

-None

See also:

SetSession,OnServiceNtf,SubscribeNotifications,UnSubscribeNotifications

virtual E_CWORD33_Status C_CWORD33__CWORD77_Service::AttachOpenSessionAckCallback (HANDLE *hApp*)[protected], [pure virtual]

AttachOpenSessionAckCallback virtual function to attche open session ack callback

Parameters:

in	*hApp*	HANDLE - Handle to the Framework

Returns:

status E_CWORD33_Status - Success or Error

E_CWORD33_Status C_CWORD33__CWORD77_Service::CloseServiceOnUnavailability (HANDLE *hApp*)

Brief

function to close session when service unavailable.

Parameters:

in	*f_hApp*	f_hApp - Handle to the Framework

f_hApp HANDLE*

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK or Error

Preconditons

nopreconditions

Change of internal status

set m_bServiceAvailable FALSE.

Classification

public

Type

sync only

See also:

CloseSession

void C_CWORD33__CWORD77_Service::CloseSession (HANDLE *f_hApp*)[protected]

Brief

Function to close sessions

Parameters:

in	*f_hApp*	f_hApp - Handle to the Framework

f_hApp HANDLE*

Return values:

none	

Preconditons

none

Change of internal status

The internal state is not changed.

Classification

public

Type

sync only

See also:

none

virtual void C_CWORD33__CWORD77_Service::EnableSession (UI_32 *f_uiNum*, UI_32 *f_uiFirst*, ...)[pure virtual]

Brief

virtual function to enable session in service

Parameters:

in	*f_uiNum*	f_uiNum - Number of arguments

f_uiNum UI_32

Parameters:

in	*f_uiFirst*	f_uiFirst - first argument

f_uiFirst UI_32

Return values:

None	

Preconditons

Change of internal status

The internal state is not changed.

Classification

public

Type

None

See also:

none

virtual E_CWORD33_Status C_CWORD33__CWORD77_Service::HandleServiceMessage (UI_32 *cmdID*)[pure virtual]

Brief

HandleMessage virtual function to process message coming from *CWORD77*

Parameters:

in	*cmdID*	cmdID - Request ID

cmdID UI_32

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK or Error

Preconditons

nopreconditions

Change of internal status

The internal state is not changed.

Classification

public

Type

none

See also:

none

E_CWORD33_Status C_CWORD33__CWORD77_Service::OnServiceNtf (HANDLE *hApp*)

Brief

unsubscribe notifications

Parameters:

in	*hApp*	hApp - Handle to the Framework.

hApp HANDLE*

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK or e_CWORD33_StatusNullPointer

Preconditons

nopreconditions

Change of internal status

none.

Classification

public

Type

sync only

See also:

_CWORD33_GetMsgDataOfSize, SetRespNotfnDataIn_CWORD77_DataPool

E_CWORD33_Status C_CWORD33__CWORD77_Service::OpenServiceOnAvailability (HANDLE *f_hApp*, const std::string & *f_sServiceName*)

Brief

function to open session with specified Service name when service available.

Parameters:

in	*f_hApp*	f_hApp - Handle to the Framework

f_hApp HANDLE*

Parameters:

in	*f_sServiceName*	f_sServiceName - service name

f_sServiceName - const std::string &

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK or Error

Preconditions

nopreconditions

Change of internal status

save Framework handler into m_hApp.

Classification

public

Type

sync only

See also:

AttachOpenSessionAckCallback, OpenSession

void C_CWORD33__CWORD77_Service::OpenSession (HANDLE *f_hApp*, const std::string & *f_sServiceName*)[protected]

Brief

Function to open sessions with specified Service Name

Parameters:

in	*f_hApp*	f_hApp - Handle to the Framework

f_hApp HANDLE*

Parameters:

in	*f_sServiceName*	f_sServiceName - Service Name

f_sServiceName - const std::string&

Return values:

none	

Preconditons

none

Change of internal status

Save service name into member variable.

Classification

public

Type

sync only

See also:

none

void C_CWORD33__CWORD77_Service::OpenSessionAcks (HANDLE *f_hApp*)

Brief

Acknowledgement process of open session

Parameters:

in	*f_hApp*	HANDLE - Handle to the Application

Return values:

none	

Preconditons

Set Session object in C_CWORD33__CWORD77_Service::SetSession()

Set Callback function in
C_CWORD33__CWORD77_Service::Set_CWORD77_OpenSessionACK()

Change of internal status

none

Classification

public

Type

none

See also:

SetSession,Set_CWORD77_OpenSessionACK

void C_CWORD33__CWORD77_Service::OpenSessionRetry (UI_32 *f_ui32SessionType*)[protected]

OpenSessionRetry This function sends open session request to the service m_cServiceName

Parameters:

in	*f_ui32SessionType*	UI_32 - Session type to open

Returns:

none

void C_CWORD33__CWORD77_Service::SendMessageToSession (UI_32 *f_uiSessionType*, UI_32 *f_uiSrvProtocol*)[protected]

application framework handle

Brief

Function to send message to a session

Parameters:

in	*f_uiSessionType*	UI_32 - session type
in	*f_uiSrvProtocol*	UI_32 - Protocol ID

Return values:

none	

Preconditons

C_CWORD33__CWORD77_Service::SetSession()

Change of internal status

The internal state is not changed.

Classification

public

Type

none

See also:

SetSession,_CWORD33_SendMsg

void C_CWORD33__CWORD77_Service::Set_CWORD77_OpenSessionACK (SessionAckTo_CWORD77_ *fptr*)

Brief

Attach open session ack callback function to service.

Parameters:

in	*fptr*	fptr - Callback Function pointer of notify open session ack.

fptr SessionAckTo_CWORD77_*

Preconditons

nopreconditions

Change of internal status

set callback function into m_cbSessionACK..

Classification

public

Type

sync only

See also:

none

void C_CWORD33__CWORD77_Service::SetResponseCallback (ResponseServiceTo_CWORD77_ *fptr*)

Brief

Set Call back function into Sessions.

Parameters:

in	*fptr*	fptr - Pointer to call back function

fptr ResponseServiceTo_CWORD77_*

Return values:

void C_CWORD33__CWORD77_Service::SetSession (C_CWORD33__CWORD77_Session * *f_pSessionObj*)

Brief

API to set session objects in service.

Parameters:

in	*f_pSessionObj*	f_pSessionObj - Pointer to an instance of session object.

f_pSessionObj C_CWORD33__CWORD77_Session*

Return values:

E_CWORD33_Status C_CWORD33__CWORD77_Service::SubscribeNotifications (HANDLE *hApp*)

Brief

Subscribe notifications.

Parameters:

in	*hApp*	hApp - Handle to the Framework.

hApp HANDLE*

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK or e_CWORD33_StatusInvldParam

Preconditons

nopreconditions

Change of internal status

save pointer of notifications into framework handler.

Classification

public

Type

sync only

See also:

_CWORD33_SubscribeNotificationWithCallback

E_CWORD33_Status C_CWORD33__CWORD77_Service::UnSubscribeNotifications (HANDLE *hApp*)

Brief

unsubscribe notifications

Parameters:

in	*hApp*	hApp - Handle to the Framework.

hApp HANDLE*

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK or e_CWORD33_StatusInvldParam

Preconditons

nopreconditions

Change of internal status

delete notifications from framework handler.

Classification

public

Type

sync only

See also:

_CWORD33_SubscribeNotificationWithCallback

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

24 _CWORD78___CWORD77__service_if.h

C_CWORD33__CWORD77_Session Class Reference

Public Member Functions

C_CWORD33__CWORD77_Session ()
virtual ~C_CWORD33__CWORD77_Session ()
E_CWORD33_Status HandleSessionMessage (UI_32 cmdID)
void SetResponseCallback (ResponseServiceTo_CWORD77_ fptr)
virtual E_CWORD33_Status OpenSession (HANDLE f_hApp, std::string f_sServiceName)=0
virtual E_CWORD33_Status CloseSession (HANDLE f_hApp)=0
virtual E_CWORD33_Status OpenSessionAcknowledge (HANDLE f_hApp)=0
UI_32 GetSessionType ()
void SetSessionType (const UI_32 &sessioType)
void AttachResponseCallbacks (HANDLE hApp)
void DetachResponseCallbacks (HANDLE hApp)
E_CWORD33_Status OnResponse (HANDLE hApp)
virtual UI_32 const * **GetResponseProtocolArray** ()=0
virtual UI_32 const **GetResponseProtocolArrayLength** ()=0
virtual UI_32 const * **GetEventArray** ()=0
virtual UI_32 const **GetEventArrayLength** ()=0

Protected Member Functions

void AddProtocolHandler (UI_32 f_uiCmdId, CbFuncPtr f_cbFuncPtr)
E_CWORD33_Status AttachProtocolwithDispatcher (HANDLE f_hApp)
E_CWORD33_Status DetachProtocolwithDispatcher (HANDLE f_hApp)
E_CWORD33_Status AttachEventswithDispatcher (HANDLE f_hApp)
E_CWORD33_Status RegisterEvents (HANDLE f_hApp)
E_CWORD33_Status AttachEventHandlerswithDispatcher (HANDLE f_hApp)

the naming conflict in registering the event

E_CWORD33_Status DetachEventHandlerswithDispatcher (HANDLE f_hApp)
void AddEventHandler (UI_32 f_uiCmdId, CbFuncPtr f_cbFuncPtr)

Protected Attributes

HANDLE m_hSession

Handle to session.

HANDLE m_hApp

Handle to Framework.

HANDLE m_hService

Handle to Service.

UI_32 m_uiSessionType

Session Type.

ResponseServiceTo_CWORD77_ m_ResponseTo_CWORD77_

Call back function pointer.

std::vector< _CWORD33_ProtocolCallbackHandler > **m_vSessionProtocolHanlder**
std::vector< UI_32 > **m_vEventVector**

std::vector< _CWORD33_ProtocolCallbackHandler > **m_vSessionEventHandler**

Constructor & Destructor Documentation

C_CWORD33__CWORD77_Session::C_CWORD33__CWORD77_Session ()

Brief

Constructor for C_CWORD33__CWORD77_Session

Return values:

none	

Preconditons

nopreconditions

Change of internal status

none

Classification

public

Type

none

See also:

~C_CWORD33__CWORD77_Session

virtual C_CWORD33__CWORD77_Session::~C_CWORD33__CWORD77_Session ()[virtual]

Brief

Destructor for C_CWORD33__CWORD77_Session

Return values:

none	

Preconditons

C_CWORD33__CWORD77_Session is decleared in constructor

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33__CWORD77_Session

Member Function Documentation

void C_CWORD33__CWORD77_Session::AddEventHandler (UI_32 *f_uiCmdId*, CbFuncPtr *f_cbFuncPtr*)[protected]

AddEventHandler Function to add event handler

Parameters:

in	*f_uiCmdId*	UI_32 - Command id
in	*f_cbFuncPtr*	CbFuncPtr - Callback function pointer

Returns:

None

void C_CWORD33__CWORD77_Session::AddProtocolHandler (UI_32 *f_uiCmdId*, CbFuncPtr *f_cbFuncPtr*)[protected]

AddProtocol Function to add notification

Parameters:

in	*f_uiCmdId*	UI_32 - Command id
in	*f_cbFuncPtr*	CbFuncPtr - Callback function pointer

Returns:

None

E_CWORD33_Status C_CWORD33__CWORD77_Session::AttachEventHandlerswithDispatcher (HANDLE *f_hApp*)[protected]

the naming conflict in registering the event

AttachEventHandlerswithDispatcher API to Register Session 1 of Service A protocols with Dispatcher

Parameters:

in	*f_hApp*	HANDLE - Handle to the Framework

Returns:

status E_CWORD33_Status - Success or Error

E_CWORD33_Status C_CWORD33__CWORD77_Session::AttachEventswithDispatcher (HANDLE *f_hApp*)[protected]

AttachEventswithDispatcher Function to register events with dispatcher

Parameters:

in	*f_hApp*	HANDLE - Handle to the Framework

Returns:

status E_CWORD33_Status - Success or Error

E_CWORD33_Status C_CWORD33__CWORD77_Session::AttachProtocolwithDispatcher (HANDLE *f_hApp*)[protected]

AttachProtocolwithDispatcher API to Register Session 1 of Service A protocols with Dispatcher

Parameters:

in	*f_hApp*	HANDLE - Handle to the Framework

Returns:

status E_CWORD33_Status - Success or Error

void C_CWORD33__CWORD77_Session::AttachResponseCallbacks (HANDLE *hApp*)

Brief

declearcallbackfuncionintoDispatcher

Parameters:

in	*hApp*	HANDLE - Handle to Application

Return values:

none	

Preconditons

noprecondition

Change of internal status

none

Classification

public

Type

none

See also:

_CWORD33_AttachCallbackToDispatcher, _CWORD33_RegisterEvent, GetResponseProtocolArray, GetEventArray, OnResponse

virtual E_CWORD33_Status C_CWORD33__CWORD77_Session::CloseSession (HANDLE *f_hApp*)[pure virtual]

Brief

function to close Session.(pure virtual function)

Parameters:

in	*f_hApp*	f_hApp - Handle of framework

f_hApp HANDLE*

Return values:

E_CWORD33_Status	Success or Error

Preconditons

nopreconditions

Change of internal status

none

Classification

public

Type

none

See also:

E_CWORD33_Status C_CWORD33__CWORD77_Session::DetachEventHandlerswithDispatcher (HANDLE *f_hApp*)[protected]

DetachEventHandlerswithDispatcher API to Un Register Session 1 of Service A protocols with Dispatcher

Parameters:

in	*f_hApp*	HANDLE - Handle to the Framework

Returns:

status E_CWORD33_Status - Success or Error

E_CWORD33_Status C_CWORD33__CWORD77_Session::DetachProtocolwithDispatcher (HANDLE *f_hApp*)[protected]

DetachProtocolwithDispatcher API to Un Register Session 1 of Service A protocols with Dispatcher

Parameters:

in	*f_hApp*	HANDLE - Handle to the Framework

Returns:

status E_CWORD33_Status - Success or Error

void C_CWORD33__CWORD77_Session::DetachResponseCallbacks (HANDLE *hApp*)

Brief

unscrible dispatcher callback function

Parameters:

in	*hApp*	HANDLE - Handle to Application

Return values:

none	

Preconditons

no preconditions

Change of internal status

none

Classification

public

Type

none

See also:

_CWORD33_DetachCallbackFromDispatcher, _CWORD33_UnRegisterEvent, GetResponseProtocolArray, GetEventArray

UI_32 C_CWORD33__CWORD77_Session::GetSessionType ()

GetSessionType API to get session type

Returns:

UI_32 - Session type E_CWORD33_Status - Success or Error

Brief

API to get session type

Parameters:

none	

Return values:

UI_32	- Session type

Preconditons

nopreconditions

Change of internal status

none

Classification

public

Type

sync only

See also:

E_CWORD33_Status C_CWORD33__CWORD77_Session::HandleSessionMessage (UI_32 *cmdID*)

Brief

send specific command by session

Parameters:

in	*cmdID*	cmdID - commond id

cmdID UI_32

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK or failed

Preconditons

no preconditions

Change of internal status

none

Classification

public

Type

sync only

See also:

GetReqDataFrom_CWORD77_DataPool

E_CWORD33_Status C_CWORD33__CWORD77_Session::OnResponse (HANDLE *hApp*)

OnResponse Callback function on receiving response protocol or event specified in the list, returned by either GetResponseProtocolArray or GetEventArray

Parameters:

in	*hApp*	HANDLE - Handle to framework

Returns:

None

Brief

Callback function on receiving response protocol or event specified in the list, returned by either GetResponseProtocolArray or GetEventArray

Parameters:

in	*f_hApp*	HANDLE - Handle of framework

Return values:

E_CWORD33_Status	Suce_CWORD33_StatusOK or e_CWORD33_StatusNullPointer

Preconditions

nopreconditions

Change of internal status

framework handler is invalid handler.[e_CWORD33_StatusNullPointer]

Classification

public

Type

none

See also:

_CWORD33_GetMsgDataOfSize

virtual E_CWORD33_Status C_CWORD33__CWORD77_Session::OpenSession (HANDLE *f_hApp*, std::string *f_sServiceName*)[pure virtual]

Brief

function to Open Session.(pure virtual function)

Parameters:

in	*f_hApp*	f_hApp - Handle of framework

f_hApp HANDLE*

Parameters:

in	*f_sServiceName*	f_sServiceName - Service Name

f_sServiceName std::string

Return values:

E_CWORD33_Status	Success or Error

Preconditons

nopreconditions

Change of internal status

none

Classification

public

Type

none

See also:

virtual E_CWORD33_Status C_CWORD33__CWORD77_Session::OpenSessionAcknowledge (HANDLE *f_hApp*)[pure virtual]

Brief

Open Session acknowledge.(pure virtual function)

Parameters:

in	*f_hApp*	f_hApp - Handle of framework

f_hApp HANDLE*

Return values:

E_CWORD33_Status	Success or Error

Preconditons

nopreconditions

Change of internal status

none

Classification

public

Type

none

See also:

E_CWORD33_Status C_CWORD33__CWORD77_Session::RegisterEvents (HANDLE *f_hApp*)[protected]

RegisterEvents Function to register events

Parameters:

in	*f_hApp*	HANDLE - Handle to the Framework

Returns:

status E_CWORD33_Status - Success or Error

void C_CWORD33__CWORD77_Session::SetResponseCallback (ResponseServiceTo_CWORD77_ *fptr*)

Brief

Set Call back function in Session

Parameters:

in	*fptr*	fptr - Pointer to call back function

fptr ResponseServiceTo_CWORD77_*

Return values:

none	

Preconditons

noppreconditions

Change of internal status

set callback into m_ResponseTo_CWORD77_.

Classification

public

Type

sync only

See also:

void C_CWORD33__CWORD77_Session::SetSessionType (const UI_32 & *sessioType*)

Brief

API to set session type into class C_CWORD33__CWORD77_Session object.

Parameters:

in	*sessioType*	const UI_32& - Session type

Return values:

none	

Preconditons

none

Change of internal status

none

Classification

public

Type

sync only

See also:

none

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

25 _CWORD78___CWORD77__session_if.h

C_CWORD33_Action Class Reference

Public Member Functions

C_CWORD33_Action (std::string f_strName)
virtual ~C_CWORD33_Action ()
virtual VOID _CWORD33_Action (C_CWORD33_State *f_pSourceState, C_CWORD33_State *f_pTargetState, CEventDataPtr f_pData)=0

Public Attributes

std::string **m_strName**

Constructor & Destructor Documentation

C_CWORD33_Action::C_CWORD33_Action (std::string *f_strName*)

Brief

Constructor for class C_CWORD33_Action.

Parameters:

in	*f_strName*	std::string -Name of Action object

Return values:

virtual C_CWORD33_Action::~C_CWORD33_Action ()[inline], [virtual]

Brief

Destructor for C_CWORD33_Action

Return values:

none	

Preconditons

none

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33_Action

Here is the call graph for this function:

IMAGE 

Member Function Documentation

virtual VOID C_CWORD33_Action::_CWORD33_Action (C_CWORD33_State * *f_pSourceState*, C_CWORD33_State * *f_pTargetState*, CEventDataPtr *f_pData*)[pure virtual]

Brief

Callback function of C_CWORD33_Action.(pure virtual function)

Parameters:

in	*f_pSourceState*	C_CWORD33_State - pointer of source state
in	*f_pTargetState*	C_CWORD33_State - pointer of target state
in	*f_pData*	CEventDataPtr - pointer of event data

Return values:

none	

Preconditons

none

Change of internal status

none

Classification

public

Type

none

See also:

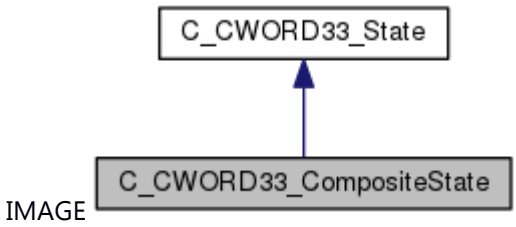
none

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

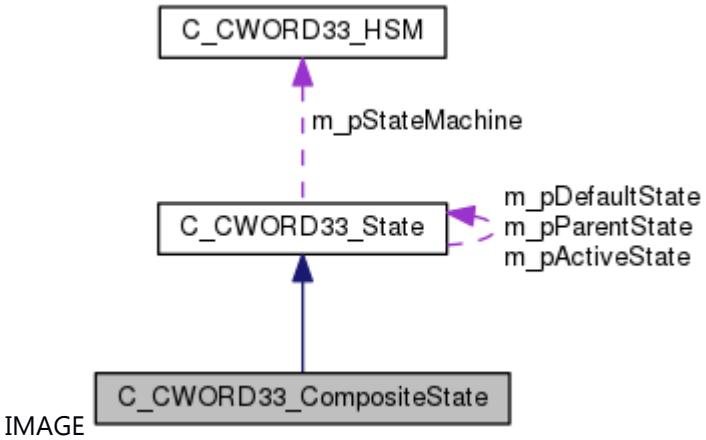
26 _CWORD78__sm_action.h

C_CWORD33_CompositeState Class Reference

#include <_CWORD78__sm_compositestate.h>
Inheritance diagram for C_CWORD33_CompositeState:



Collaboration diagram for C_CWORD33_CompositeState:



Public Types

enum **__CWORD33__STATE_TYPE** { **e_CWORD33_State** = ***, **e_CWORD33_DefaultState** }
typedef enum C_CWORD33_CompositeState::__CWORD33__STATE_TYPE
_CWORD33__STATE_TYPE

Public Member Functions

C_CWORD33_CompositeState (std::string f_pName)
virtual ~C_CWORD33_CompositeState ()
E_CWORD33_Status _CWORD33_AddState (C_CWORD33_State *f_pState,
_CWORD33__STATE_TYPE f_eStateType=e_CWORD33_State)
C_CWORD33_State * _CWORD33_GetDefaultState ()
virtual E_CWORD33_Status _CWORD33_PrintStates ()
virtual BOOL _CWORD33_HasSubStates ()
virtual C_CWORD33_State * _CWORD33_GetActiveState ()
E_CWORD33_Status UpdateHistory ()
virtual E_CWORD33_Status _CWORD33_PrintXML (std::ostringstream &f_strXMLString)

Protected Member Functions

virtual E_CWORD33_Status _CWORD33_OnEntry (CEventDataPtr f_pEventData)

virtual E_CWORD33_Status _CWORD33_OnExit (CEventDataPtr f_pEventData)
virtual E_CWORD33_Status _CWORD33_SetHSM (C_CWORD33_HSM *f_pStatemachine)

Protected Attributes

ChildStateList * m_pChildStates

stores the pointers to sub state, key is state name

Additional Inherited Members

Detailed Description

This class implements the additional functionality supported by HSM Composite state. It provides the standard interfaces for adding state.

Constructor & Destructor Documentation

C_CWORD33_CompositeState::C_CWORD33_CompositeState (std::string *f_pName*)

Brief

Constructor for class C_CWORD33_CompositeState.

Parameters:

in	*f_strName*	std::string -Name of state

Return values:

virtual C_CWORD33_CompositeState::~C_CWORD33_CompositeState ()[virtual]

Brief

Destructor for C_CWORD33_CompositeState

Return values:

none	

Preconditons

Self instance of C_CWORD33_CompositeState created.

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33_State, C_CWORD33_CompositeState::C_CWORD33_CompositeState

---

Member Function Documentation

E_CWORD33_Status C_CWORD33_CompositeState::_CWORD33_AddState (C_CWORD33_State * *f_pState*, _CWORD33__STATE_TYPE *f_eStateType* = e_CWORD33_State)

Brief

sets the given state as a substate of the current state. If the f_eStateType is e_CWORD33_DefaultState then substate is default state for current state.

Parameters:

in	*f_pState*	C_CWORD33_State* - Pointer to the substate object to be added in the current state
in	*f_eStateType*	_CWORD33__STATE_TYPE - state type indicating if it is default state

Return values:

E_CWORD33_Status	success or fail

Preconditons

Self instance of C_CWORD33_CompositeState created.

Change of internal status

none

Classification

public

Type

none

See also:

none

virtual C_CWORD33_State* C_CWORD33_CompositeState::_CWORD33_GetActiveState ()[virtual]

Brief

This interface returns the Active state of the current composite state. In case of non-composite state current state is active state

Parameters:

none	

Return values:

*C_CWORD33_State**	pointer of active state

Preconditons

Self instance of C_CWORD33_CompositeState created.

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33_State::_CWORD33_GetActiveState

Implements C_CWORD33_State.

C_CWORD33_State* C_CWORD33_CompositeState::_CWORD33_GetDefaultState ()

Brief

Returns the default state of the current composite state.

Parameters:

none	

Return values:

*C_CWORD33_State**	- pointer default state

Preconditons

Self instance of C_CWORD33_CompositeState created.

Change of internal status

none

Classification

public

Type

none

See also:

none

virtual BOOL C_CWORD33_CompositeState::_CWORD33_HasSubStates ()[virtual]

Brief

This indicates if the state has sub states. It returns TRUE only in the CompositeState where this function is overridden

Parameters:

none	

Return values:

BOOL	TRUE - has substate FALSE - has not substatue

Preconditons

Self instance of C_CWORD33_CompositeState created.

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33_State::_CWORD33_HasSubStates

Reimplemented from C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_CompositeState::_CWORD33_OnEntry (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnEntry This is pure virtual function implemented by the derived classes of the application. state initialization can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_CompositeState::_CWORD33_OnExit (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnExit This is pure virtual function implemented by the derived classes of the application. state cleanup can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_CompositeState::_CWORD33_PrintStates ()[virtual]

Brief

Logs the state name and events associated with the state.

Parameters:

none	

Return values:

e_CWORD33_StatusOK	succ
e_CWORD33_StatusNullPointer	failed

Preconditons

Self instance of C_CWORD33_CompositeState created.

Change of internal status

none

Classification

public

Type

none

See also:

none

Reimplemented from C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_CompositeState::_CWORD33_PrintXML (std::ostringstream & *f_strXMLString*)[virtual]

Brief

Update the State information in the given stream in the form of XML tags

Parameters:

f_strXMLString	std::ostringstream & - reference to the XML stream

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK e_CWORD33_StatusNullPointer

Preconditons

Self instance of C_CWORD33_CompositeState created.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal status

Self instance of C_CWORD33_CompositeState created failed.
[e_CWORD33_StatusNullPointer].

Classification

public

Type

none

See also:

C_CWORD33_State::_CWORD33_PrintXML

Reimplemented from C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_CompositeState::_CWORD33_SetHSM (C_CWORD33_HSM * *f_pStatemachine*)[protected], [virtual]

Brief

This Interface associates the gives statemachine with the current state

Parameters:

in	*f_pStatemachine*	C_CWORD33_HSM* - Statemachine associated with the state.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusNullPointer	NULL pointer exception

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

C_CWORD33_HSM

_CWORD33_SetHSM This Interface associates the gives statemachine with the current state

Parameters:

in	*f_pStatemachine*	C_CWORD33_HSM* - Statemachine associated with the state.

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Reimplemented from C_CWORD33_State.

E_CWORD33_Status C_CWORD33_CompositeState::UpdateHistory ()[virtual]

Brief

This function stores the last active state

Parameters:

none	

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK

Preconditons

Self instance of C_CWORD33_CompositeState created.

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33_State::UpdateHistory

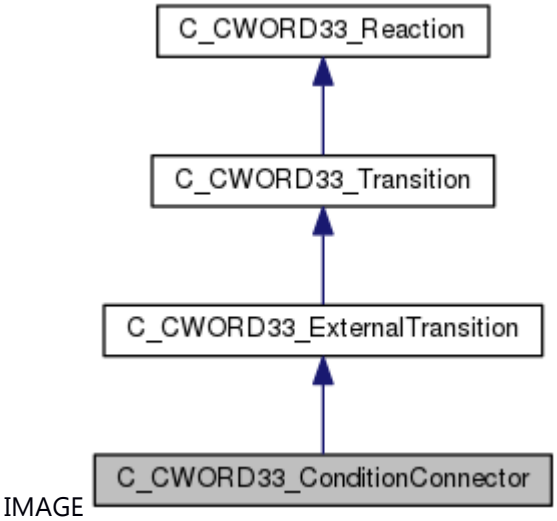
Implements C_CWORD33_State.

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

27 _CWORD78__sm_compositestate.h

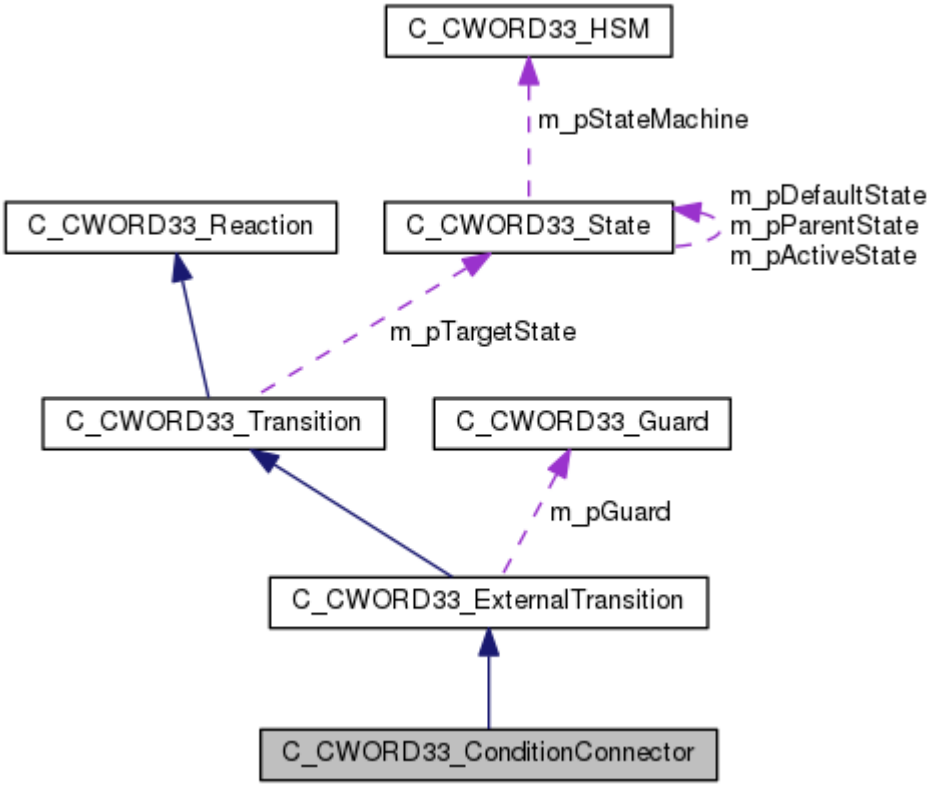
C_CWORD33_ConditionConnector Class Reference

Inheritance diagram for C_CWORD33_ConditionConnector:



IMAGE

Collaboration diagram for C_CWORD33_ConditionConnector:



IMAGE

Public Member Functions

C_CWORD33_ConditionConnector (std::string f_strName)
virtual ~C_CWORD33_ConditionConnector ()

virtual C_CWORD33_State * _CWORD33_Reaction (C_CWORD33_State *f_pSourceState,
CEventDataPtr f_pData)
virtual E_CWORD33_Status _CWORD33_AddCondition (C_CWORD33_Guard *f_pGuard,
C_CWORD33_State *f_pTargetState)

Additional Inherited Members

Constructor & Destructor Documentation

C_CWORD33_ConditionConnector::C_CWORD33_ConditionConnector (std::string *f_strName*)

Brief

Constructor for class C_CWORD33_ConditionConnector.

Parameters:

in	*f_strName*	std::string - Name of the condition connector

Return values:

virtual C_CWORD33_ConditionConnector::~C_CWORD33_ConditionConnector ()[virtual]

Brief

Destructor for C_CWORD33_CompositeState

Return values:

none	

Preconditons

Self instance of C_CWORD33_ConditionConnector created.

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33_ConditionConnector::C_CWORD33_ConditionConnector

Member Function Documentation

virtual E_CWORD33_Status C_CWORD33_ConditionConnector::_CWORD33_AddCondition (C_CWORD33_Guard * *f_pGuard*, C_CWORD33_State * *f_pTargetState*)[virtual]

Brief

Adds connect condition into condition list of current connection

Parameters:

in	*f_pGuard*	C_CWORD33_State* - Pointer to the guard object to be added in the currrent connection
in	*f_pTargetState*	C_CWORD33_State* - Target state for given guard condition

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK e_CWORD33_StatusNullPointer

Preconditons

Self instance of C_CWORD33_ConditionConnector created.

Change of internal status

none

Classification

public

Type

none

See also:

none

virtual C_CWORD33_State* C_CWORD33_ConditionConnector::_CWORD33_Reaction (C_CWORD33_State * *f_pSourceState*, CEventDataPtr *f_pData*)[virtual]

Brief

This API evaluates the guards added in the condition list. If the guard is evaluated as true then statemachine transitions to target state associated with guard.

Parameters:

in	*f_pSourceState*	C_CWORD33_State* - Source state in which reaction is being executed
in	*f_pData*	CEventDataPtr - event data

Return values:

*C_CWORD33_State**	C_CWORD33_State* - Returns Active state

Preconditons

Self instance of C_CWORD33_ConditionConnector created.

Change of internal status

none

Classification

public

Type

none

See also:

none

Reimplemented from C_CWORD33_ExternalTransition.

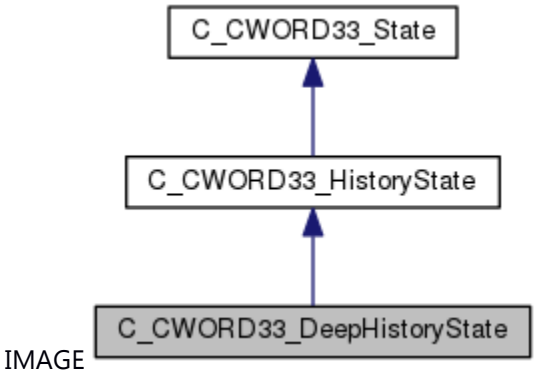
The documentation for this class was generated from the following file:

28 _CWORD78__sm_conditionconnector.h

C_CWORD33_DeepHistoryState Class Reference

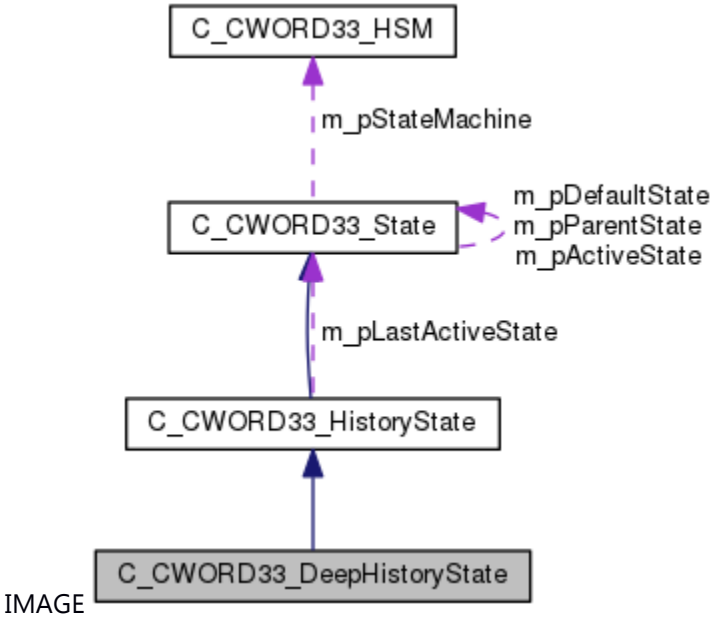
This class implements the additional functionality supported by HSM Shallow History state.

#include <_CWORD78__sm_deephistorystate.h>
Inheritance diagram for C_CWORD33_DeepHistoryState:



IMAGE

Collaboration diagram for C_CWORD33_DeepHistoryState:



IMAGE

Public Member Functions

C_CWORD33_DeepHistoryState (std::string f_pName)
~C_CWORD33_DeepHistoryState ()
E_CWORD33_Status UpdateHistory ()

Additional Inherited Members

Detailed Description

This class implements the additional functionality supported by HSM Shallow History state.

Constructor & Destructor Documentation

C_CWORD33_DeepHistoryState::C_CWORD33_DeepHistoryState (std::string *f_pName*)

Brief

Constructor for class C_CWORD33_DeepHistoryState.

Parameters:

in	*f_pName*	std::string - Name of the state

Return values:

C_CWORD33_DeepHistoryState::~C_CWORD33_DeepHistoryState ()

Brief

Destructor for C_CWORD33_DeepHistoryState

Return values:

none	

Preconditons

Self instance of C_CWORD33_DeepHistoryState created.

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33_DeepHistoryState::C_CWORD33_DeepHistoryState

Member Function Documentation

E_CWORD33_Status C_CWORD33_DeepHistoryState::UpdateHistory ()[virtual]

Brief

This function stores the last active state

Parameters:

Implements C_CWORD33_HistoryState.

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

29 _CWORD78__sm_deephistorystate.h

C_CWORD33_EventFactory Class Reference

This class defines the events used by the statemachine framework.

#include <_CWORD78__sm_eventfactory.h>

Public Member Functions

C_CWORD33_EventFactory ()
virtual ~C_CWORD33_EventFactory ()

Static Public Attributes

static const UI_32 ev_CWORD33_Start = ***

Defines the events for starting an application.

static const UI_32 ev_CWORD33_EventLimit = ***

Defines the Higher limit for event id reserved only for framework.

Detailed Description

This class defines the events used by the statemachine framework.

Constructor & Destructor Documentation

C_CWORD33_EventFactory::C_CWORD33_EventFactory ()

Brief

Constructor for class C_CWORD33_EventFactory.

Parameters:

none	

Return values:

virtual C_CWORD33_EventFactory::~C_CWORD33_EventFactory ()[virtual]

Brief

Destructor for C_CWORD33_EventFactory

Return values:

none	

Preconditons

Self instance of C_CWORD33_EventFactory created.

Change of internal status

none

Classification

public

Type

none

See also:

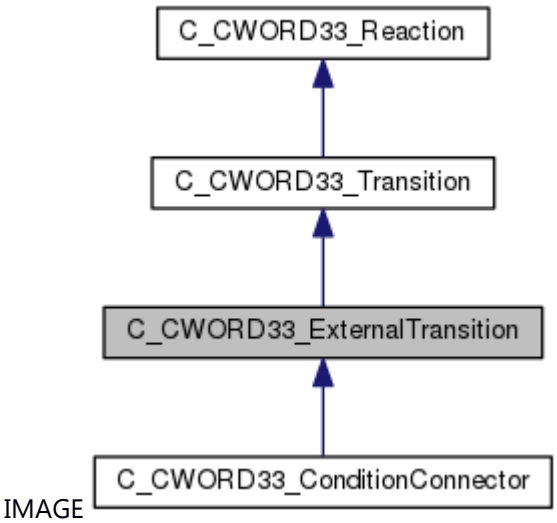
C_CWORD33_EventFactory::C_CWORD33_EventFactory

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

30 _CWORD78__sm_eventfactory.h

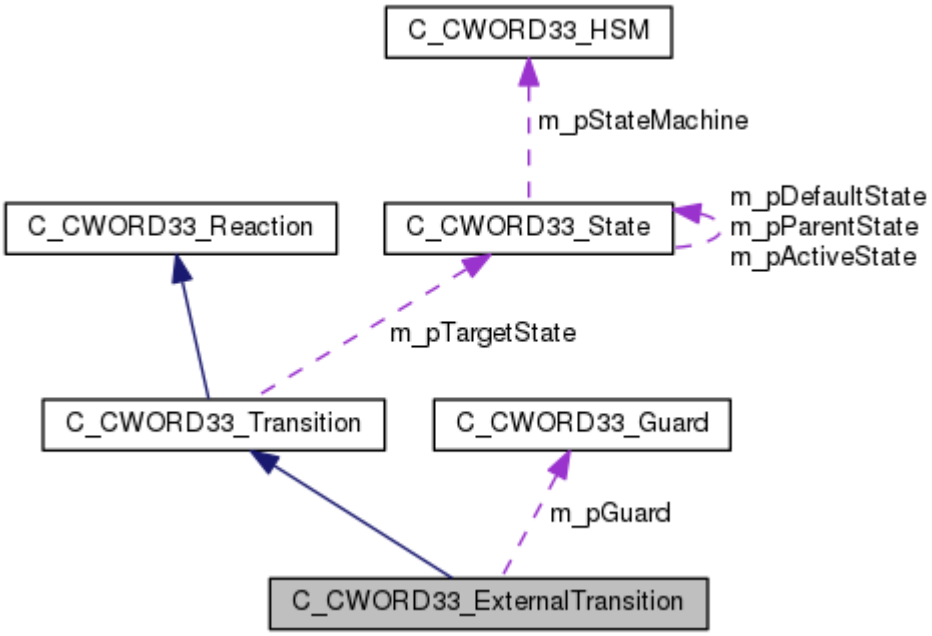
C_CWORD33_ExternalTransition Class Reference

#include <_CWORD78__sm_externaltransition.h>
Inheritance diagram for C_CWORD33_ExternalTransition:



IMAGE

Collaboration diagram for C_CWORD33_ExternalTransition:



IMAGE

Public Member Functions

C_CWORD33_ExternalTransition (C_CWORD33_State *f_pTargetState)
virtual ~C_CWORD33_ExternalTransition ()
virtual C_CWORD33_State * _CWORD33_Reaction (C_CWORD33_State *f_pSourceState,
CEventDataPtr f_pData)

virtual E_CWORD33_Status _CWORD33_SetGuard (C_CWORD33_Guard *f_pGuard)
virtual E_CWORD33_Status _CWORD33_AddAction (C_CWORD33_Action *f_pAction)

Protected Types

typedef std::vector< C_CWORD33_Action * > **TActionList**
typedef TActionList::iterator **TActionListIterator**

Protected Member Functions

C_CWORD33_State * ExecuteTransition (C_CWORD33_State *f_pSourceState, CEventDataPtr f_pData)

Protected Attributes

C_CWORD33_Guard * m_pGuard

pointer to the guard condition object

TActionList * **m_pActionList**

Additional Inherited Members

Detailed Description

This class implements the _CWORD33_Reaction interface to support transition from one state to another state.

Constructor & Destructor Documentation

C_CWORD33_ExternalTransition::C_CWORD33_ExternalTransition (C_CWORD33_State * *f_pTargetState*)

Brief

Constructor for class C_CWORD33_ExternalTransition.

Parameters:

in	*f_pTargetState*	C_CWORD33_State* - pointer of target C_CWORD33_State object

Return values:

none	

Prerequisite

none

Change of internal state

none

Classification

public

Type

none

See also:

C_CWORD33_ExternalTransition, _CWORD33_Reaction, _CWORD33_AddAction

C_CWORD33_ExternalTransition Parameterized constructor

Parameters:

in	*f_pTargetState*	C_CWORD33_State* - Target state object

Returns:

none

virtual C_CWORD33_ExternalTransition::~C_CWORD33_ExternalTransition ()[virtual]

Brief

Destructor for C_CWORD33_ExternalTransition

Return values:

none	

Preconditons

Self instance of C_CWORD33_ExternalTransition created.

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33_ExternalTransition::C_CWORD33_ExternalTransition

~C_CWORD33_ExternalTransition Class destructor

Returns:

none

Member Function Documentation

virtual E_CWORD33_Status C_CWORD33_ExternalTransition::_CWORD33_AddAction (C_CWORD33_Action * *f_pAction*)[virtual]

Brief

Adds C_CWORD33_Action object to the action list

Parameters:

f_pAction	C_CWORD33_Action* - C_CWORD33_Action object pointer

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusNullPointer	set null pointer

Preconditons

Self instance of C_CWORD33_ExternalTransition created.

Change of internal status

none

Classification

public

Type

none

See also:

_CWORD33_Reaction, _CWORD33_SetGuard

virtual C_CWORD33_State* C_CWORD33_ExternalTransition::_CWORD33_Reaction (C_CWORD33_State * *f_pSourceState*, CEventDataPtr *f_pData*)[virtual]

Brief

C_CWORD33_ExternalTransition instance event process.

Parameters:

f_pSourceState	C_CWORD33_State* - source state object pointer
f_pData	CEventDataPtr* - CEventData event pointer

Return values:

*C_CWORD33_State**	active state pointer
NULL	

Preconditons

Self instance of C_CWORD33_ExternalTransition created.

Change of internal status

If input parameter f_pSourceState is NULL. [NULL]
If member variable m_pTargetState is NULL. [NULL]

Classification

public

Type

none

See also:

_CWORD33_SetGuard, _CWORD33_AddAction

Implements C_CWORD33_Transition.

Reimplemented in C_CWORD33_ConditionConnector.

virtual E_CWORD33_Status C_CWORD33_ExternalTransition::_CWORD33_SetGuard (C_CWORD33_Guard * *f_pGuard*)[virtual]

Brief

Sets the Guard object in External Transition

Parameters:

f_pGuard	C_CWORD33_Guard* - C_CWORD33_Guard object pointer

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusNullPointer	set null pointer

Preconditons

Self instance of C_CWORD33_ExternalTransition created.

Change of internal status

If input parameter C_CWORD33_Guard is NULL. [e_CWORD33_StatusNullPointer]

Classification

public

Type

none

See also:

_CWORD33_Reaction, _CWORD33_AddAction

_CWORD33_SetGuard Sets the Guard object in External Transition

Parameters:

in	*f_pGuard*	C_CWORD33_Guard* - guard object

Returns:

status E_CWORD33_Status -

C_CWORD33_State* C_CWORD33_ExternalTransition::ExecuteTransition (C_CWORD33_State * *f_pSourceState*, CEventDataPtr *f_pData*)[protected]

ExecuteTransition Execute the transition actions, calls OnExit of Source State and OnEntry of target state

Parameters:

in	*f_pSourceState*	CEventDataPtr - source state
in	*f_pData*	C_CWORD33_State* - Event data

Returns:

status E_CWORD33_Status -

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

31 _CWORD78__sm_externaltransition.h

C_CWORD33_Guard Class Reference

Public Member Functions

C_CWORD33_Guard (std::string f_strName)
virtual ~C_CWORD33_Guard ()
virtual BOOL _CWORD33_Evaluate ()=0

Constructor & Destructor Documentation

C_CWORD33_Guard::C_CWORD33_Guard (std::string *f_strName*)

Brief

C_CWORD33_Guard's constructor

Parameters:

in	*f_strName*	string - Name of the guard

Return values:

none	

Preconditons

None

Change of internal status

None

Classification

Public

Type

Sync only

See also:

~C_CWORD33_Guard

virtual C_CWORD33_Guard::~C_CWORD33_Guard ()[virtual]

~

Brief

Destructor of C_CWORD33_Guard class

Parameters:

None	

Return values:

none	

Preconditons

Calls constructor to successfully create object.

Change of internal status

None

Classification

Public

Type

Sync only

See also:

C_CWORD33_Guard

Member Function Documentation

virtual BOOL C_CWORD33_Guard::_CWORD33_Evaluate ()[pure virtual]

Brief

This is a pure virtual function that defines the logic for evaluating the guard condition

Parameters:

None	

Return values:

TRUE	- if test passes
FALSE	- otherwise returns false

Preconditons

None

Change of internal status

None

Classification

Public

Type

Sync only

See also:

None

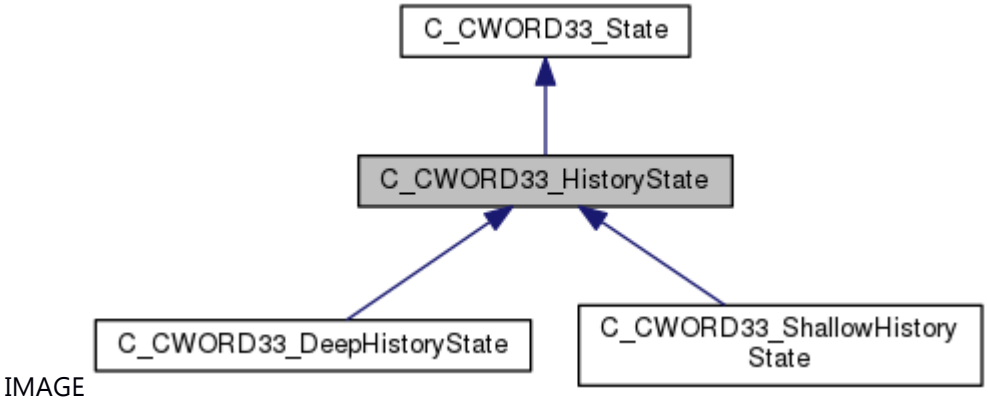
The documentation for this class was generated from the following file:

32 _CWORD78__sm_guard.h

C_CWORD33_HistoryState Class Reference

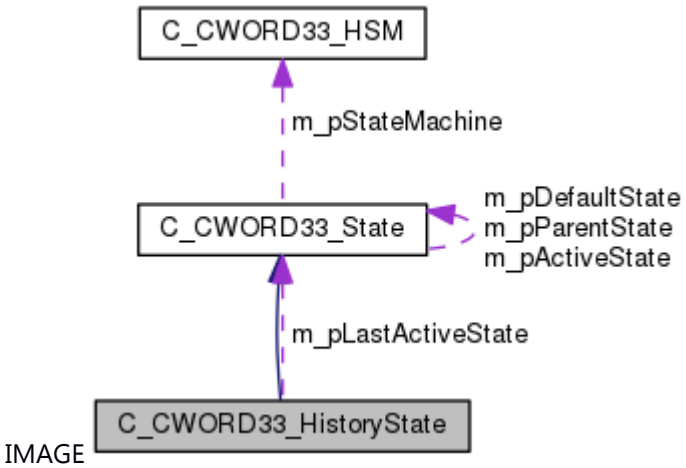
this file has the C_CWORD33_HistoryState class definitions

#include <_CWORD78__sm_historystate.h>
Inheritance diagram for C_CWORD33_HistoryState:



IMAGE

Collaboration diagram for C_CWORD33_HistoryState:



IMAGE

Public Member Functions

C_CWORD33_HistoryState (std::string f_pName)
virtual ~C_CWORD33_HistoryState ()
virtual C_CWORD33_State * _CWORD33_GetActiveState ()
virtual E_CWORD33_Status UpdateHistory ()=0
E_CWORD33_Status SetDefaultHistory ()

Protected Member Functions

E_CWORD33_Status _CWORD33_OnEntry (CEventDataPtr f_pEventData)
E_CWORD33_Status _CWORD33_OnExit (CEventDataPtr f_pEventData)
virtual C_CWORD33_State * _CWORD33_OnHSMStart (CEventDataPtr f_pEventData)
virtual C_CWORD33_State * _CWORD33_OnHSMStop (CEventDataPtr f_pEventData)

Protected Attributes

C_CWORD33_State * **m_pLastActiveState**
UI_32 **m_uiEventId**
std::string **m_cEventName**

Additional Inherited Members

Detailed Description

this file has the C_CWORD33_HistoryState class definitions

Brief Introduction

This class implements the additional functionality supported by HSM History state.

Constructor & Destructor Documentation

C_CWORD33_HistoryState::C_CWORD33_HistoryState (std::string *f_pName*)

Brief

-Constructor of C_CWORD33_HistoryState.

Parameters:

in	*f_pName*	std::string - Name of the state

Return values:

None	

Preconditons

None

Change of internal status

None

Classification

Public

See also:

~C_CWORD33_HistoryState

C_CWORD33_HistoryState Parameterized constructor

Parameters:

in	*f_pName*	string - Name of the state

Returns:

none

virtual C_CWORD33_HistoryState::~C_CWORD33_HistoryState ()[virtual]

Brief

-Destructor of C_CWORD33_HistoryState.

Parameters:

None	

Return values:

None	

Preconditons

-Calls constructor to successfully create object.

Change of internal status

None

Classification

Public

See also:

C_CWORD33_HistoryState::C_CWORD33_HistoryState

~C_CWORD33_HistoryState Class destructor

Returns:

none

Member Function Documentation

virtual C_CWORD33_State* C_CWORD33_HistoryState::_CWORD33_GetActiveState ()[virtual]

Brief

Gets the pointor of this class.

Parameters:

None	

Return values:

None	

Preconditons

Calls constructor to successfully create object.

Change of internal status

None

Classification

Public

See also:

None

Implements C_CWORD33_State.

E_CWORD33_Status C_CWORD33_HistoryState::_CWORD33_OnEntry (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnEntry state initialization can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

E_CWORD33_Status C_CWORD33_HistoryState::_CWORD33_OnExit (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnExit state cleanup can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

virtual C_CWORD33_State* C_CWORD33_HistoryState::_CWORD33_OnHSMStart (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnHSMStart This function is called recursively till the leaf state is reached. This internally calls the Entry function of the current state.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

CurrentState C_CWORD33_State* - Returns current state after operation

Reimplemented from C_CWORD33_State.

virtual C_CWORD33_State* C_CWORD33_HistoryState::_CWORD33_OnHSMStop (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnHSMStop This function is called recursively till the required parent state is reached. This internally calls the Exit function of the current state.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

CurrentState C_CWORD33_State* - Returns current state after operation

Reimplemented from C_CWORD33_State.

E_CWORD33_Status C_CWORD33_HistoryState::SetDefaultHistory ()

Brief

This function sets the default active state in history state.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	-Success
e_CWORD33_StatusNullPointer	- Null Pointer

Preconditons

-Calls constructor to successfully create object.

Change of internal status

None

Classification

Public

See also:

None

SetDefaultHistory This function sets the default active state in history state

Parameters:

virtual E_CWORD33_Status C_CWORD33_HistoryState::UpdateHistory ()[pure virtual]

Brief

This function stores the last active state.

Parameters:

None	

Return values:

E_CWORD33_Status	- Returns status of operation

Preconditons

-None

Change of internal status

None

Classification

Public

See also:

None

Implements C_CWORD33_State.

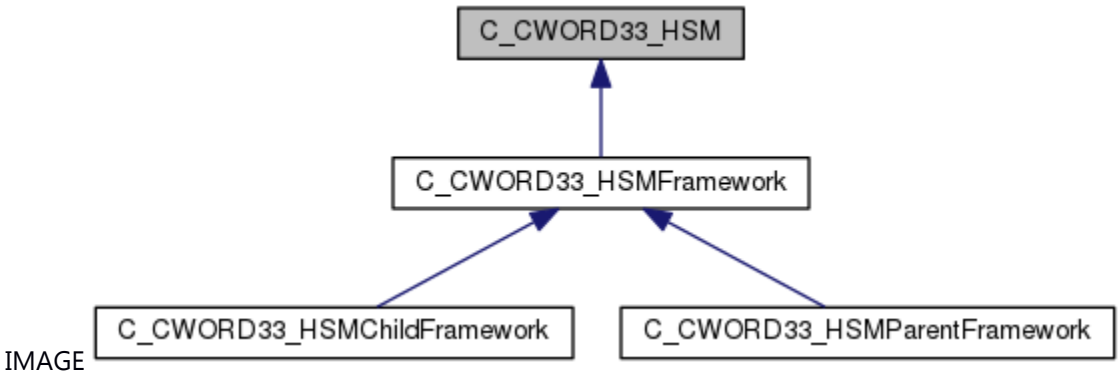
Implemented in C_CWORD33_ShallowHistoryState, and C_CWORD33_DeepHistoryState.

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

33 _CWORD78__sm_historystate.h

C_CWORD33_HSM Class Reference

#include <_CWORD78__sm_hsm.h>
Inheritance diagram for C_CWORD33_HSM:



IMAGE

Public Member Functions

virtual ~C_CWORD33_HSM ()
C_CWORD33_HSM ()
C_CWORD33_HSM (PVOID f_pHApp)
C_CWORD33_State * _CWORD33_GetActiveState ()
E_CWORD33_Status _CWORD33_PostEvent (UI_32 f_uiEventId)
E_CWORD33_Status _CWORD33_PostEvent (CEventDataPtr f_pEventData)
virtual E_CWORD33_Status _CWORD33_Create (PVOID f_pEventData=NULL)=0
E_CWORD33_Status _CWORD33_Start (CEventDataPtr f_pEventData=boost::make_shared< CEventData >(0))
E_CWORD33_Status _CWORD33_Close (CEventDataPtr f_pEventData=boost::make_shared< CEventData >(0))
E_CWORD33_Status _CWORD33_PrintAllStates ()
E_CWORD33_Status _CWORD33_Connect (C_CWORD33_State *f_pParentState, C_CWORD33_State *f_pChildState, UI_32 f_uiEventId, C_CWORD33_Reaction *f_pReaction, BOOL f_eIsDefaultState=FALSE, BOOL f_eIsDeferredEventType=FALSE, std::string f_strEventName="")
E_CWORD33_Status _CWORD33_Connect (C_CWORD33_State *f_pParentState, C_CWORD33_State *f_pChildState, BOOL f_eIsDefaultState=FALSE)
E_CWORD33_Status _CWORD33_Connect (C_CWORD33_State *f_pState, UI_32 f_uiEventId, C_CWORD33_Reaction *f_pReaction, std::string f_strEventName="", BOOL f_eIsDeferredEventType=FALSE)
E_CWORD33_Status _CWORD33_Connect (C_CWORD33_State *f_pRootState)
E_CWORD33_Status _CWORD33_ConnectOrthogonal (C_CWORD33_OrthogonalState *f_pOrthogonalState, C_CWORD33_CompositeState *f_pOrthogonalRegion)
HANDLE _CWORD33_GetAppHandle ()
E_CWORD33_Status _CWORD33_PrintXML ()
E_CWORD33_Status _CWORD33_QueueEvent (CEventDataPtr f_pEventData)
E_CWORD33_Status RemoveEventFromPostedEventQueue (const UI_32 f_uiEventId)

Public Attributes

UI_32 **m_uiCurrentEvent**

HANDLE m_pHApp

Application handle.

Detailed Description

This class implements interfaces for connecting child states to parent states, connecting events to state.

Constructor & Destructor Documentation

virtual C_CWORD33_HSM::~C_CWORD33_HSM ()[virtual]

Brief

Destructor for C_CWORD33_HSM

Return values:

none	

Preconditons

Self instance of C_CWORD33_HSM created.

Change of internal status

none

Classification

public

Type

none

See also:

none

~C_CWORD33_HSM Class destructor

Returns:

none

C_CWORD33_HSM::C_CWORD33_HSM ()

Brief

Constructor for class C_CWORD33_HSM.

Parameters:

C_CWORD33_HSM::C_CWORD33_HSM (PVOID *f_pHApp*)

Brief

Constructor for class C_CWORD33_HSM.

Parameters:

f_pHApp	PVOID - handle of application

Return values:

none	

Prerequisite

none

Change of internal state

none

Classification

public

Type

none

See also:

~C_CWORD33_HSM

C_CWORD33_HSM(PVOID f_pHApp) Class constructor

Returns:

none

Member Function Documentation

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_Close (CEventDataPtr *f_pEventData* = boost::make_shared< CEventData >(0))

Brief

Stop the state machine.

Parameters:

f_pEventData	CEventDataPtr - Event data

Return values:

E_CWORD33_Status	
e_CWORD33_StatusNullPointer	null pointer

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

_CWORD33_PostEvent(UI_32), CEventData, _CWORD33_QueueEvent, ProcessEvent

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_Connect (C_CWORD33_State * *f_pParentState*, C_CWORD33_State * *f_pChildState*, UI_32 *f_uiEventId*, C_CWORD33_Reaction * *f_pReaction*, BOOL *f_eIsDefaultState* = FALSE, BOOL *f_eIsDeferredEventType* = FALSE, std::string *f_strEventName* = "")

34 (f_pChildState)

35 **Brief**

This connects the reaction to event and add event to child states then add child state to parent state.

36 **Parameters:**

f_pParentState	C_CWORD33_State* - Parent state pointer
f_pChildState	C_CWORD33_State* - Child state pointer
f_uiEventId	UI_32 - eventi id
f_pReaction	C_CWORD33_Reaction* - Reaction object for attaching to event
f_eIsDefaultState	BOOL - Does has default state.(default: FALSE)

37

TRUE : has default state
FALSE: not has default state

38 **Parameters:**

f_eIsDeferredEventType	BOOL - Does has the deferred event(default: FALSE)

39

TRUE : has the deferred event
FALSE: not has the deferred event

40 **Parameters:**

f_strEventName	std::string - name of event(default:"")

41 **Return values:**

e_CWORD33_StatusOK	succ
e_CWORD33_StatusNullPointer	null pointer

42 **Prerequisite**

Self instance of C_CWORD33_HSM created.

43 **Change of internal state**

none

44 **Classification**

public

45 **Type**

none

46 **See also:**

_CWORD33_Connect(C_CWORD33_State*,UI_32,C_CWORD33_Reaction*,std::string,BOOL), _CWORD33_Connect(C_CWORD33_State*,C_CWORD33_State*,BOOL)

47 _CWORD33_Connect This connects the reaction to event and add event to child states then add child state to parent state.

Parameters:

in	*f_pParentState*	C_CWORD33_State* - Parent state
in	*f_pChildState*	C_CWORD33_State* - Child state
in	*f_uiEventId*	UI_32 - Event id
in	*f_pReaction*	C_CWORD33_Reaction* - Reaction object for attaching to event
in	*f_eIsDefaultState*	BOOL - Defines if the state is default state
in	*f_eIsDeferredEventType*	BOOL - defines if the event is deferred event
in	*f_strEventName*	BOOL - Event name

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_Connect (C_CWORD33_State * *f_pParentState*, C_CWORD33_State * *f_pChildState*, BOOL *f_eIsDefaultState* = FALSE)

Brief

This add child state to parent state.

Parameters:

f_pParentState	C_CWORD33_State* - Parent state pointer
f_pChildState	C_CWORD33_State* - Child state pointer
f_eIsDefaultState	BOOL - Does has default state.(default: FALSE)

TRUE : has default state
FALSE: not has default state

Return values:

e_CWORD33_StatusOK	succ
e_CWORD33_StatusNullPointer	null pointer

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

_CWORD33_Connect(C_CWORD33_State*,UI_32,C_CWORD33_Reaction*,std::string,BOOL),

_CWORD33_Connect This add child state to parent state.

Parameters:

in	*f_pParentState*	C_CWORD33_State* - Parent state
in	*f_pChildState*	C_CWORD33_State* - Child state
in	*f_eIsDefaultState*	BOOL - Defines if the state is default state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_Connect (C_CWORD33_State * *f_pState*, UI_32 *f_uiEventId*, C_CWORD33_Reaction * *f_pReaction*, std::string *f_strEventName* = "", BOOL *f_eIsDeferredEventType* = FALSE)

Brief

This connects the reaction to event and add event to states.

Parameters:

f_pState	C_CWORD33_State* - pointer of state class
f_uiEventId	UI_32 - eventi id
f_pReaction	C_CWORD33_Reaction* - Reaction object for attaching to event
f_eIsDefaultState	BOOL - Does has default state.(default: FALSE)

TRUE : has default state
FALSE: not has default state

Parameters:

f_eIsDeferredEventType	BOOL - Does has the deferred event(default: FALSE)

TRUE : has the deferred event
FALSE: not has the deferred event

Parameters:

f_strEventName	std::string - name of event(default:"")

Return values:

e_CWORD33_StatusOK	succ

e_CWORD33_StatusNullPointer	null pointer

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

_CWORD33_Connect(C_CWORD33_State*,C_CWORD33_State*,UI_32,C_CWORD33_Reaction*,BOOL,BOOL,std::string)

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_Connect (C_CWORD33_State * *f_pRootState*)

Brief

This sets the givens state as root state in the state machine

Parameters:

f_pRootState	C_CWORD33_State* - Root state

Return values:

e_CWORD33_StatusOK	succ
e_CWORD33_StatusNullPointer	null pointer

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

_CWORD33_Connect(C_CWORD33_State*,C_CWORD33_State*,UI_32,C_CWORD33_Reaction*,BOOL,BOOL,std::string)

_CWORD33_Connect This sets the givens state as root state in the state machine

Parameters:

in	*f_pRootState*	C_CWORD33_State* - Root state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_ConnectOrthogonal (C_CWORD33_OrthogonalState * *f_pOrthogonalState*, C_CWORD33_CompositeState * *f_pOrthogonalRegion*)

Brief

Set Composite state as HSM state OrthogonalState.

Parameters:

f_pOrthogonalState	C_CWORD33_OrthogonalState* - Orthogonal state pointer
f_pOrthogonalRegion	C_CWORD33_CompositeState* - Composite state pointer

Return values:

e_CWORD33_StatusOK	succ
e_CWORD33_StatusNullPointer	null pointer(process failed)

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

_CWORD33_Connect This sets the givens state as root state in the state machine

Parameters:

in	*f_pOrthogonalState*	C_CWORD33_OrthogonalState* - Orthogonal state
in	*f_pOrthogonalRegion*	C_CWORD33_CompositeState* - Orthogonal region

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

virtual E_CWORD33_Status C_CWORD33_HSM::_CWORD33_Create (PVOID *f_pEventData* = NULL)[pure virtual]

Brief

This is pure virtual function to be implemented by the derived classes. Statemachine states and events created and connected in this interface.

Parameters:

f_pEventData	PVOID - Event data

Return values:

E_CWORD33_Status	Returns status of operation

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

none

Implemented in C_CWORD33_HSMChildFramework, C_CWORD33_HSMParentFramework, and C_CWORD33_HSMFramework.

C_CWORD33_State* C_CWORD33_HSM::_CWORD33_GetActiveState ()

Brief

Returns the active state of the statemachine

Parameters:

none	

Return values:

*C_CWORD33_State**	- Returns ActiveState
NULL	- failed

Prerequisite

Self instance of C_CWORD33_HSM created and active state exist.

Change of internal state

none

Classification

public

Type

none

See also:

C_CWORD33_State:: _CWORD33_GetActiveState

HANDLE C_CWORD33_HSM::_CWORD33_GetAppHandle ()

Brief

This interface returns the application handle

Parameters:

none	

Return values:

HANDLE	application handle

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

none

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_PostEvent (UI_32 *f_uiEventId*)

Brief

This creates the default event data and sends the event to the active HSM state.

Parameters:

f_uiEventId	UI_32 - Event Id

Return values:

e_CWORD33_StatusOK	add succ
e_CWORD33_StatusNullPointer	set null pointer

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

_CWORD33_PostEvent(CEventDataPtr), CEventData, _CWORD33_QueueEvent, ProcessEvent

_CWORD33_PostEvent This creates the default event data and sends the event to the active HSM state.

Parameters:

in	*f_uiEventId*	UI_32 - Event Id

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_PostEvent (CEventDataPtr *f_pEventData*)

Brief

Add event data into event queue of state machine.

Parameters:

f_pEventData	CEventDataPtr - Event data

Return values:

e_CWORD33_StatusOK	add succ
e_CWORD33_StatusNullPointer	set null pointer

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

_CWORD33_PostEvent(UI_32), CEventData, _CWORD33_QueueEvent, ProcessEvent

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_PrintAllStates ()

Brief

Output log of all states and events name associated with every state.

Parameters:

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_PrintXML ()

Brief

Output statemachine information into XML file.

Parameters:

none	

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK succ

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

none

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_QueueEvent (CEventDataPtr *f_pEventData*)

Brief

Push the event in the post event list.

Parameters:

f_pEventData	CEventDataPtr - pointer of event data

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK succ e_CWORD33_StatusNullPointer NULL pointer

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

none

E_CWORD33_Status C_CWORD33_HSM::_CWORD33_Start (CEventDataPtr *f_pEventData* = boost::make_shared< CEventData >(0))

Brief

Start the state machine.

Parameters:

f_pEventData	CEventDataPtr - Event data

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK succ
e_CWORD33_StatusNullPointer	set null pointer

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

_CWORD33_PostEvent(UI_32), CEventData, _CWORD33_QueueEvent, ProcessEvent

E_CWORD33_Status C_CWORD33_HSM::RemoveEventFromPostedEventQueue (const UI_32 *f_uiEventId*)

Brief

Remove the all events of eventId f_uiEventId from event queue of statemachine.

Parameters:

f_pEventData	const UI_32 - event id

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK succ e_CWORD33_StatusInvldID Not found event id e_CWORD33_StatusNullPointer NULL pointer

Prerequisite

Self instance of C_CWORD33_HSM created.

Change of internal state

none

Classification

public

Type

none

See also:

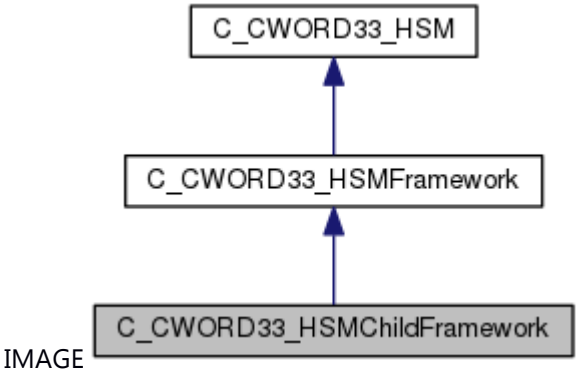
none

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

48 _CWORD78__sm_hsm.h

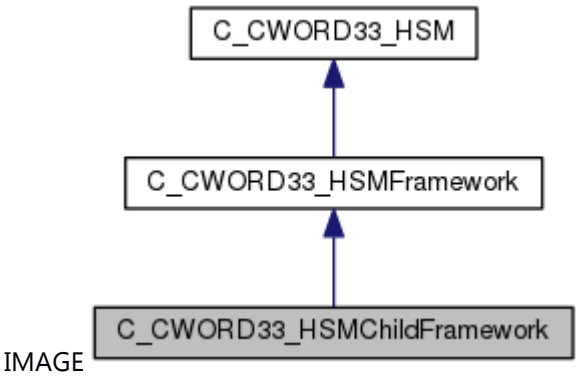
C_CWORD33_HSMChildFramework Class Reference

Inheritance diagram for C_CWORD33_HSMChildFramework:



IMAGE

Collaboration diagram for C_CWORD33_HSMChildFramework:



IMAGE

Public Member Functions

C_CWORD33_HSMChildFramework ()

Define s_CWORD33_ThreadIdle state.

C_CWORD33_HSMChildFramework (PVOID f_pHApp)

virtual ~C_CWORD33_HSMChildFramework ()

virtual E_CWORD33_Status _CWORD33_Create (PVOID f_pEventData=NULL)

E_CWORD33_Status _CWORD33_FrameworkConnect (C_CWORD33_State *f_pAppState, BOOL f_bIsDefaultState=FALSE)

E_CWORD33_Status _CWORD33_FrameworkConnect (_CWORD33__HSM_STATES f_e_CWORD33_State, UI_32 f_uiEventId, C_CWORD33_Reaction *f_pReaction, std::string f_strEventName="", BOOL f_bIsDeferredEvent=FALSE)

Public Attributes

CbFuncPtr m_fpStartThread

function pointer of the Thread start routine

CbFuncPtr m_fpStopThread

function pointer of the Thread stop routine

Additional Inherited Members

Constructor & Destructor Documentation

C_CWORD33_HSMChildFramework::C_CWORD33_HSMChildFramework ()

Define s_CWORD33_ThreadIdle state.

Define s_CWORD33_ThreadIdle state Define s_CWORD33_ThreadStart state Define s_CWORD33_ThreadReady state Define s_CWORD33_ThreadStop state

Brief

C_CWORD33_HSMChildFramework constructor

Parameters:

None	

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

C_CWORD33_HSMChildFramework default constructor

Returns:

none

C_CWORD33_HSMChildFramework::C_CWORD33_HSMChildFramework (PVOID *f_pHApp*)

Brief

C_CWORD33_HSMChildFramework constructor with param

Parameters:

in	*f_pHApp*	PVOID f_pHApp - application handle

Return values:

None	

Prerequisite

None

Change of internal state

m_p_CWORD33_StateList = new std::map<UI_32, C_CWORD33_State *>();

Classification

Public

Type

sync only

See also:

C_CWORD33_HSMChildFramework default constructor

Parameters:

in	*f_pHApp*	PVOID - application handle

Returns:

none

virtual C_CWORD33_HSMChildFramework::~C_CWORD33_HSMChildFramework ()[virtual]

Brief

C_CWORD33_HSMChildFramework destructor

Parameters:

None	

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

~C_CWORD33_HSMChildFramework Class destructor

Returns:

none

Member Function Documentation

virtual E_CWORD33_Status C_CWORD33_HSMChildFramework::_CWORD33_Create (PVOID *f_pEventData* = NULL)[virtual]

Brief

create state and event

Parameters:

in	*f_pEventData*	PVOID f_pEventData - event data

Return values:

e_CWORD33_StatusOK	create state and event success
e_CWORD33_StatusNullPointer	NULL point exception occur when create state and event

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_Create Statemachine states and events created and connected in this interface.

Parameters:

in	*f_pEventData*	PVOID - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_HSMFramework.

E_CWORD33_Status C_CWORD33_HSMChildFramework::_CWORD33_FrameworkConnect (C_CWORD33_State * *f_pAppState*, BOOL *f_bIsDefaultState* = FALSE)[virtual]

Brief

add sub state

Parameters:

in	*f_pAppState*	C_CWORD33_State *f_pAppState - state object to be added in the ready state.
in	*f_bIsDefaultState*	BOOL f_bIsDefaultState - defines if the state is default state.

Return values:

e_CWORD33_StatusOK	add sub state success
e_CWORD33_StatusNullPointer	NULL point exception occur when add sub state

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_FrameworkConnect This adds the given state as a sub state of sReady state of the Framework

Parameters:

in	*f_pAppState*	C_CWORD33_State* - state object to be added in the ready state.
in	*f_bIsDefaultState*	BOOL - defines if the state is default state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_HSMFramework.

E_CWORD33_Status C_CWORD33_HSMChildFramework::_CWORD33_FrameworkConnect (_CWORD33__HSM_STATES *f_e_CWORD33_State*, UI_32 *f_uiEventId*, C_CWORD33_Reaction * *f_pReaction*, std::string *f_strEventName* = "", BOOL *f_bIsDeferredEvent* = FALSE)[virtual]

Brief

connects the reaction to event and add event to states

Parameters:

in	*f_e_CWORD33_State*	_CWORD33__HSM_STATES f_e_CWORD33_State - framework state
in	*f_uiEventId*	UI_32 f_uiEventId - Event id
in	*f_pReaction*	C_CWORD33_Reaction *f_pReaction - Reaction object
in	*f_strEventName*	std::string f_strEventName - Event name
in	*f_bIsDeferredEvent*	BOOL f_bIsDeferredEvent - defines if the event is deferred event

Return values:

e_CWORD33_StatusOK	add sub state success
e_CWORD33_StatusNullPointer	NULL point exception occur when add sub state

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_FrameworkConnect This connects the reaction to event and add event to states

Parameters:

in	*f_e_CWORD33_State*	_CWORD33__HSM_STATES - enum maping to framework state
in	*f_uiEventId*	UI_32 - Event id
in	*f_pReaction*	C_CWORD33_Reaction* - Reaction object for attaching to event
in	*f_strEventName*	std::string - Event name
in	*f_bIsDeferredEvent*	BOOL - defines if the event is deferred event

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

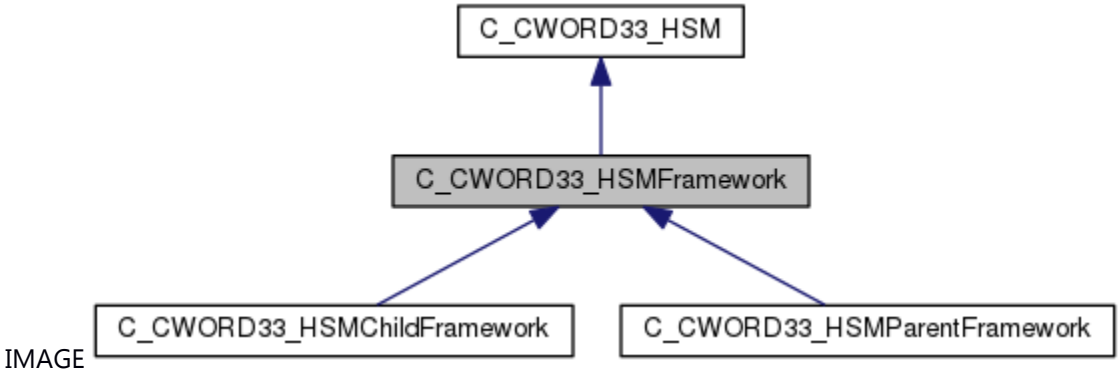
Implements C_CWORD33_HSMFramework.

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

49 _CWORD78__sm_hsmframework.h

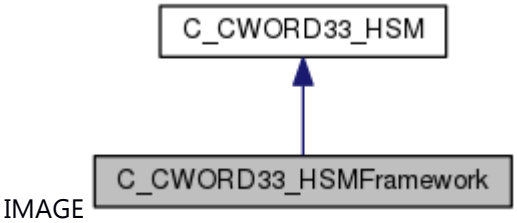
C_CWORD33_HSMFramework Class Reference

Inheritance diagram for C_CWORD33_HSMFramework:



IMAGE

Collaboration diagram for C_CWORD33_HSMFramework:



IMAGE

Public Types

enum __CWORD33__HSM_STATES { es_CWORD33_Root = ***, es_CWORD33_Initialization, es_CWORD33_LoadData, es_CWORD33_Pre, es_CWORD33_Background, es_CWORD33_Run, es_CWORD33_LoadSessions, es_CWORD33_Ready, es_CWORD33_Stop, es_CWORD33_App, es_CWORD33_UserChange, es_CWORD33_ThreadRoot, es_CWORD33_ThreadIdle, es_CWORD33_ThreadStart, es_CWORD33_ThreadReady, es_CWORD33_ThreadStop, es_CWORD33_Stopping }

typedef enum C_CWORD33_HSMFramework::__CWORD33__HSM_STATES __CWORD33__HSM_STATES

Public Member Functions

C_CWORD33_HSMFramework ()

Define ev_CWORD33_Start event.

C_CWORD33_HSMFramework (PVOID f_pHApp)

virtual E_CWORD33_Status _CWORD33_Create (PVOID f_pEventData=NULL)=0

virtual E_CWORD33_Status _CWORD33_FrameworkConnect (C_CWORD33_State *f_pAppState, BOOL f_bIsDefaultState=FALSE)=0

virtual E_CWORD33_Status _CWORD33_FrameworkConnect (_CWORD33__HSM_STATES f_e_CWORD33_State, UI_32 f_uiEventId, C_CWORD33_Reaction *f_pReaction, std::string f_strEventName="", BOOL f_bIsDeferredEvent=FALSE)=0

Protected Attributes

std::map< UI_32, C_CWORD33_State * > * m_p_CWORD33_StateList

Stores the state objects against _CWORD33__HSM_STATES enumerations.

Additional Inherited Members

Member Typedef Documentation

typedef enum C_CWORD33_HSMFramework::_CWORD33__HSM_STATES C_CWORD33_HSMFramework::_CWORD33__HSM_STATES

state machine state

Member Enumeration Documentation

enum C_CWORD33_HSMFramework::_CWORD33__HSM_STATES

state machine state

Enumerator

es_CWORD33_Root Root state.

es_CWORD33_Initialization Initialization state.

es_CWORD33_LoadData LoadData state.

es_CWORD33_Pre Pre state.

es_CWORD33_Background Background state.

es_CWORD33_Run Run state.

es_CWORD33_LoadSessions LoadSessions state.

es_CWORD33_Ready Ready state.

es_CWORD33_Stop Stop state.

es_CWORD33_App App state.

es_CWORD33_UserChange UserChange state.

es_CWORD33_ThreadRoot ThreadRoot state.

es_CWORD33_ThreadIdle ThreadIdle state.

es_CWORD33_ThreadStart ThreadStart state.

es_CWORD33_ThreadReady ThreadReady state.

es_CWORD33_ThreadStop ThreadStop state.

es_CWORD33_Stopping Stopping state.

Constructor & Destructor Documentation

C_CWORD33_HSMFramework::C_CWORD33_HSMFramework ()

Define ev_CWORD33_Start event.

Define ev_CWORD33_Stop event Define ev_CWORD33_Destroy event Define ev_CWORD33_Ready event Define ev_CWORD33_Error event Define ev_CWORD33_PreStart event Define ev_CWORD33_PreStop event Define ev_CWORD33_BackgroundStart event Define ev_CWORD33_BackgroundStop event

Brief

C_CWORD33_HSMFramework default constructor

Parameters:

None	

Return values:

None	

Prerequisite

None

Change of internal state

C_CWORD33_HSM()

Classification

Public

Type

sync only

See also:

C_CWORD33_HSMFramework::C_CWORD33_HSMFramework (PVOID *f_pHApp*)

Brief

C_CWORD33_HSMFramework constructor with param

Parameters:

in	*f_pHApp*	PVOID f_pHApp - for parent class constructor

Return values:

None	

Prerequisite

None

Change of internal state

C_CWORD33_HSM(f_pHApp)

Classification

Public

Type

sync only

See also:

Member Function Documentation

virtual E_CWORD33_Status C_CWORD33_HSMFramework::_CWORD33_Create (PVOID *f_pEventData* = NULL)[pure virtual]

Summary

pure virtual fuction

Parameters:

in	*f_pEventData*	PVOID f_pEventData - event data

Return values:

None	

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_Create Statemachine states and events created and connected in this interface.

Parameters:

in	*f_pEventData*	PVOID - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_HSM.

Implemented in C_CWORD33_HSMChildFramework, and C_CWORD33_HSMParentFramework.

virtual E_CWORD33_Status C_CWORD33_HSMFramework::_CWORD33_FrameworkConnect (C_CWORD33_State * *f_pAppState*, BOOL *f_bIsDefaultState* = FALSE)[pure virtual]

Summary

pure virtual fuction

Parameters:

in	*f_pAppState*	C_CWORD33_State *f_pAppState - state object to be added in the ready state.
in	*f_bIsDefaultState*	BOOL f_bIsDefaultState - if the state is default state

Return values:

None	

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_FrameworkConnect This adds the given state as a sub state of sReady state of the Framework

Parameters:

in	*f_pAppState*	C_CWORD33_State* - state object to be added in the ready state.
in	*f_bIsDefaultState*	BOOL - defines if the state is default state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implemented in C_CWORD33_HSMChildFramework, and C_CWORD33_HSMParentFramework.

virtual E_CWORD33_Status C_CWORD33_HSMFramework::_CWORD33_FrameworkConnect (CWORD33__HSM_STATES *f_e_CWORD33_State*, UI_32 *f_uiEventId*, C_CWORD33_Reaction * *f_pReaction*, std::string *f_strEventName* = "", BOOL *f_bIsDeferredEvent* = FALSE)[pure virtual]

Summary

pure virtual fuction

Parameters:

in	*f_e_CWORD33_State*	_CWORD33__HSM_STATES f_e_CWORD33_State - enum maping to framework state
in	*f_uiEventId*	UI_32 f_uiEventId - Event id
in	*f_pReaction*	C_CWORD33_Reaction *f_pReaction - Reaction object for attaching to event
in	*f_strEventName*	std::string f_strEventName - Event name
in	*f_bIsDeferredEvent*	BOOL f_bIsDeferredEvent - defines if the event is deferred event

Return values:

E_CWORD33_Status	depend on implement class

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_FrameworkConnect This connects the reaction to event and add event to states

Parameters:

in	*f_e_CWORD33_State*	_CWORD33__HSM_STATES - enum maping to framework state
in	*f_uiEventId*	UI_32 - Event id
in	*f_pReaction*	C_CWORD33_Reaction* - Reaction object for attaching to event
in	*f_strEventName*	std::string - Event name
in	*f_bIsDeferredEvent*	BOOL - defines if the event is deferred event

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

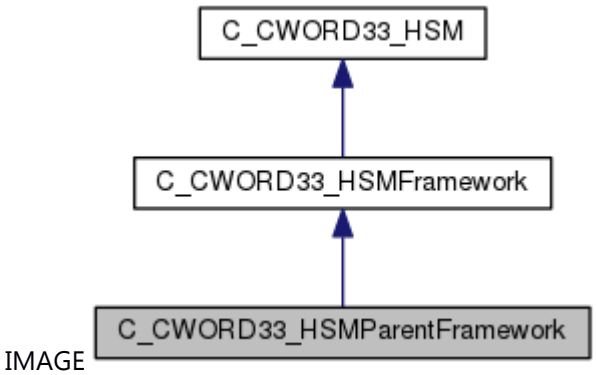
Implemented in C_CWORD33_HSMChildFramework, and C_CWORD33_HSMParentFramework.

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

50 _CWORD78__sm_hsmframework.h

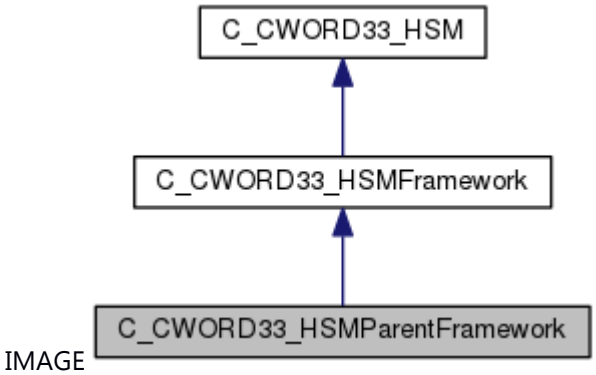
C_CWORD33_HSMParentFramework Class Reference

#include <_CWORD78__sm_hsmframework.h>
Inheritance diagram for C_CWORD33_HSMParentFramework:



IMAGE

Collaboration diagram for C_CWORD33_HSMParentFramework:



IMAGE

Public Member Functions

C_CWORD33_HSMParentFramework ()
Define ev_CWORD33_LoadPersistenceAck event.
C_CWORD33_HSMParentFramework (PVOID f_pHApp)
virtual ~C_CWORD33_HSMParentFramework ()
virtual E_CWORD33_Status _CWORD33_Create (PVOID f_pEventData=NULL)
E_CWORD33_Status _CWORD33_FrameworkConnect (C_CWORD33_State *f_pAppState, BOOL f_bIsDefaultState=FALSE)
E_CWORD33_Status _CWORD33_FrameworkConnect (_CWORD33__HSM_STATES f_e_CWORD33_State, UI_32 f_uiEventId, C_CWORD33_Reaction *f_pReaction, std::string f_strEventName="", BOOL f_bIsDeferredEvent=FALSE)

Additional Inherited Members

Detailed Description

It defines the states required for NS statemachine framework, it creates the NS statemachine and implements the reaction associated with the events

---

Constructor & Destructor Documentation

C_CWORD33_HSMParentFramework::C_CWORD33_HSMParentFramework ()

Define ev_CWORD33_LoadPersistenceAck event.

Define ev_CWORD33_LoadComplete event Define ev_CWORD33_UserChange event Define ev_CWORD33_ReInit event Define ev_CWORD33_LoadUserData event Define ev_CWORD33_CloseSessionAck event Define ev_CWORD33_CloseSessionAck event Define ev_CWORD33_StopComplete event Define ev_CWORD33_OpenSessionReq event Define ev_CWORD33_CloseSessionReq event Define ev_CWORD33_Init event Define ev_CWORD33_UserChangeComplete event Define ev_CWORD33_LoadSessions event Define s_CWORD33_Root state Define s_CWORD33_Root state Define s_CWORD33_Initialization state Define s_CWORD33_Pre state Define s_CWORD33_Background state Define s_CWORD33_Run state Define s_CWORD33_Stop state Define s_CWORD33_Ready state Define s_CWORD33_LoadData state Define s_CWORD33_LoadSessions state Define s_CWORD33_Stopping state Define s_CWORD33_UserChange state

Brief

C_CWORD33_HSMFramework default constructor

Parameters:

None	

Return values:

None	

Prerequisite

None

Change of internal state

m_p_CWORD33_StateList = new std::map<UI_32, C_CWORD33_State *>();

Classification

Public

Type

sync only

See also:

C_CWORD33_HSMParentFramework::C_CWORD33_HSMParentFramework (PVOID *f_pHApp*)

Brief

C_CWORD33_HSMFramework constructor with param

Parameters:

in	*f_pHApp*	PVOID f_pHApp - Name of the state

Return values:

None	

Prerequisite

None

Change of internal state

m_p_CWORD33_StateList = new std::map<UI_32, C_CWORD33_State *>();

Classification

Public

Type

sync only

See also:

C_CWORD33_HSMParentFramework default constructor

Parameters:

in	*f_pName*	string - Name of the state

Returns:

none

virtual C_CWORD33_HSMParentFramework::~C_CWORD33_HSMParentFramework ()[virtual]

Brief

C_CWORD33_HSMFramework destructor

Parameters:

None	

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

~C_CWORD33_HSMParentFramework Class destructor

Returns:

none

Member Function Documentation

virtual E_CWORD33_Status C_CWORD33_HSMParentFramework::_CWORD33_Create (PVOID *f_pEventData* = NULL)[virtual]

Brief

create state and event

Parameters:

in	*f_pEventData*	PVOID f_pEventData - event data

Return values:

e_CWORD33_StatusOK	create state and event success
e_CWORD33_StatusNullPointer	NULL point exception occur when create state and event

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_Create Statemachine states and events created and connected in this interface.

Parameters:

in	*f_pHApp*	PVOID - application handle

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_HSMFramework.

E_CWORD33_Status C_CWORD33_HSMParentFramework::_CWORD33_FrameworkConnect (C_CWORD33_State * *f_pAppState*, BOOL *f_bIsDefaultState* = FALSE)[virtual]

Brief

add sub state

Parameters:

in	*f_pAppState*	C_CWORD33_State *f_pAppState - state object to be added in the ready state.
in	*f_bIsDefaultState*	BOOL f_bIsDefaultState - defines if the state is default state.

Return values:

e_CWORD33_StatusOK	add sub state success
e_CWORD33_StatusNullPointer	NULL point exception occur when add sub state

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_FrameworkConnect This adds the given state as a sub state of sReady state of the Framework

Parameters:

in	*f_pAppState*	C_CWORD33_State* - state object to be added in the ready state.
in	*f_bIsDefaultState*	BOOL - defines if the state is default state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_HSMFramework.

E_CWORD33_Status C_CWORD33_HSMParentFramework::_CWORD33_FrameworkConnect (C_CWORD33_HSM_STATES *f_e_CWORD33_State*, UI_32 *f_uiEventId*, C_CWORD33_Reaction * *f_pReaction*, std::string *f_strEventName* = "", BOOL *f_bIsDeferredEvent* = FALSE)[virtual]

Brief

connects the reaction to event and add event to states

Parameters:

in	*f_e_CWORD33_State*	_CWORD33__HSM_STATES f_e_CWORD33_State - framework state
in	*f_uiEventId*	UI_32 f_uiEventId - Event id
in	*f_pReaction*	C_CWORD33_Reaction *f_pReaction - Reaction object
in	*f_strEventName*	std::string f_strEventName - Event name
in	*f_bIsDeferredEvent*	BOOL f_bIsDeferredEvent - defines if the event is deferred event

Return values:

e_CWORD33_StatusOK	add sub state success
e_CWORD33_StatusNullPointer	NULL point exception occur when add sub state

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_FrameworkConnect This connects the reaction to event and add event to states

Parameters:

in	*f_e_CWORD33_State*	_CWORD33__HSM_STATES - enum maping to framework state
in	*f_uiEventId*	UI_32 - Event id
in	*f_pReaction*	C_CWORD33_Reaction* - Reaction object for attaching to event
in	*f_strEventName*	std::string - Event name
in	*f_bIsDeferredEvent*	BOOL - defines if the event is deferred event

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_HSMFramework.

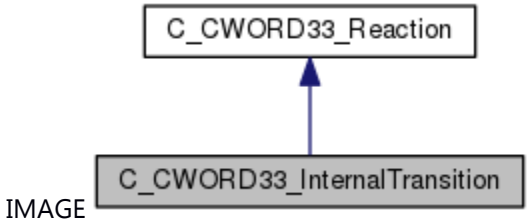
The documentation for this class was generated from the following file:

51 _CWORD78__sm_hsmframework.h

C_CWORD33_InternalTransition Class Reference

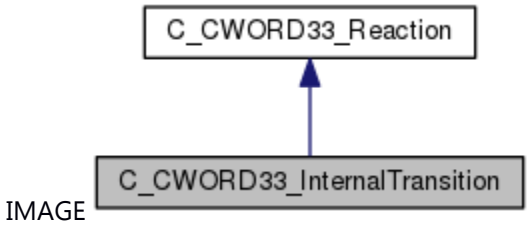
This class provides the interface for defining a reaction for an event.

#include <_CWORD78__sm_internaltransition.h>
Inheritance diagram for C_CWORD33_InternalTransition:



IMAGE

Collaboration diagram for C_CWORD33_InternalTransition:



IMAGE

Public Member Functions

virtual C_CWORD33_State * _CWORD33_Reaction (C_CWORD33_State *f_pSourceState, CEventDataPtr f_pData)=0

Additional Inherited Members

Detailed Description

This class provides the interface for defining a reaction for an event.

Member Function Documentation

virtual C_CWORD33_State* C_CWORD33_InternalTransition::_CWORD33_Reaction (C_CWORD33_State * *f_pSourceState*, CEventDataPtr *f_pData*)[pure virtual]

Summary

pure virtual fuction

Parameters:

in	*f_pSourceState*	C_CWORD33_State* f_pSourceState - source state
in	*f_pData*	CEventDataPtr f_pData - event data

Return values:

None	

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

synconly

See also:

_CWORD33_Reaction The reaction for the event has to be implemented in this function

Parameters:

in	*f_pSourceState*	C_CWORD33_State* - Event id of the event to be added in the state
in	*f_pData*	CEventDataPtr - Reaction to be associated with the with event id in the state

Returns:

ActiveState C_CWORD33_State* - Returns Active state

Implements C_CWORD33_Reaction.

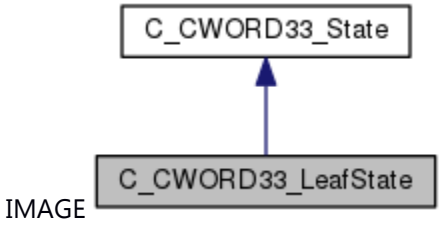
The documentation for this class was generated from the following file:

52 _CWORD78__sm_internaltransition.h

C_CWORD33_LeafState Class Reference

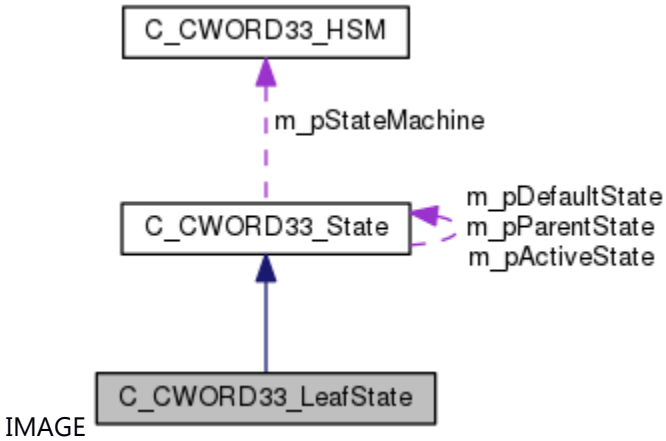
This class implements the additional functionality supported by HSM Leafstate.

#include <_CWORD78__sm_leafstate.h>
Inheritance diagram for C_CWORD33_LeafState:



IMAGE

Collaboration diagram for C_CWORD33_LeafState:



IMAGE

Public Member Functions

C_CWORD33_LeafState (std::string f_pName)
virtual C_CWORD33_State * _CWORD33_GetActiveState ()
E_CWORD33_Status UpdateHistory ()

Protected Member Functions

virtual E_CWORD33_Status _CWORD33_OnEntry (CEventDataPtr f_pEventData)
virtual E_CWORD33_Status _CWORD33_OnExit (CEventDataPtr f_pEventData)

Additional Inherited Members

Detailed Description

This class implements the additional functionality supported by HSM Leafstate.

Member Function Documentation

virtual C_CWORD33_State* C_CWORD33_LeafState::_CWORD33_GetActiveState ()[virtual]

Summary

pure virtual fuction

Parameters:

None	

Return values:

*C_CWORD33_State**	depend on implement class

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_GetActiveState This interface returns the Active state of the current composite state. In case of non-composite state current state is active state

Returns:

Active state C_CWORD33_State* - Active state

Implements C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_LeafState::_CWORD33_OnEntry (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnEntry This is pure virtual function implemented by the derived classes of the application. state initialization can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_LeafState::_CWORD33_OnExit (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnExit This is pure virtual function implemented by the derived classes of the application. state cleanup can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

E_CWORD33_Status C_CWORD33_LeafState::UpdateHistory ()[virtual]

Summary

pure virtual fuction

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

UpdateHistory This function stores the last active state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

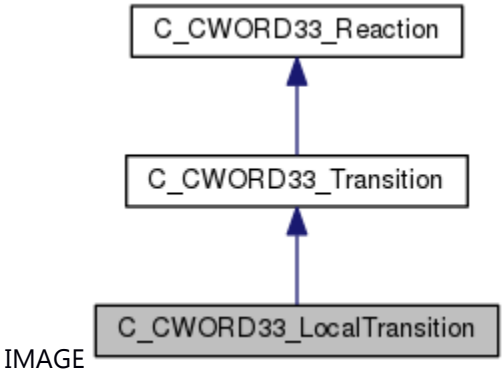
The documentation for this class was generated from the following file:

53 _CWORD78__sm_leafstate.h

C_CWORD33_LocalTransition Class Reference

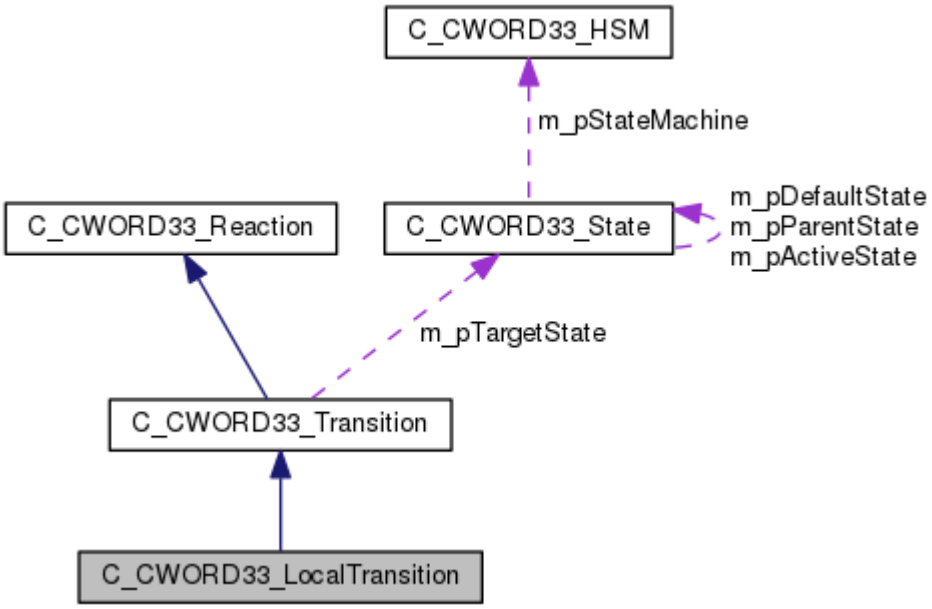
This class provides the interface for local transition over event.

#include <_CWORD78__sm_localtransition.h>
Inheritance diagram for C_CWORD33_LocalTransition:



IMAGE

Collaboration diagram for C_CWORD33_LocalTransition:



IMAGE

Public Member Functions

C_CWORD33_LocalTransition (C_CWORD33_State *f_pTargetState)
virtual ~C_CWORD33_LocalTransition ()
virtual C_CWORD33_State * _CWORD33_Reaction (C_CWORD33_State *f_pSourceState,
CEventDataPtr f_pData)

Additional Inherited Members

Detailed Description

This class provides the interface for local transition over event.

Constructor & Destructor Documentation

C_CWORD33_LocalTransition::C_CWORD33_LocalTransition (C_CWORD33_State * *f_pTargetState*)

Brief

C_CWORD33_LocalTransition constructor

Parameters:

in	*f_pTargetState*	C_CWORD33_State *f_pTargetState - target state

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

virtual C_CWORD33_LocalTransition::~C_CWORD33_LocalTransition ()[virtual]

Brief

C_CWORD33_LocalTransition destructor

Parameters:

None	

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

Member Function Documentation

virtual C_CWORD33_State* C_CWORD33_LocalTransition::_CWORD33_Reaction (C_CWORD33_State * *f_pSourceState*, CEventDataPtr *f_pData*)[virtual]

Brief

reaction for location transition

Parameters:

None	

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_Reaction The reaction for an event is implemented in this function. For local transition from inner state to outer state exit of the inner state is invoked, but entry of outer state is not invoked and from outer state to inner state entry of the inner state is invoked but exit of outer state is not invoked

Parameters:

in	*f_pSourceState*	C_CWORD33_State* - Source state in which reaction is being executed
in	*f_pData*	CEventDataPtr - event data

Returns:

ActiveState C_CWORD33_State* - Returns Active state

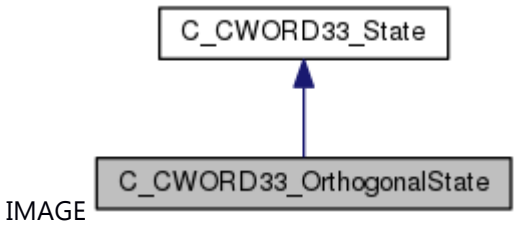
Implements C_CWORD33_Transition.

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

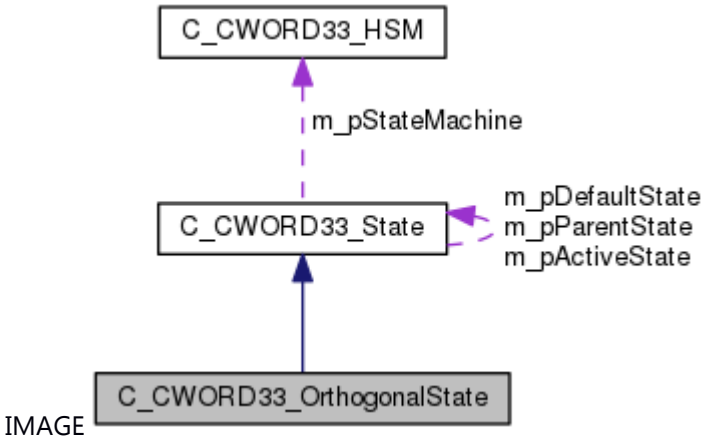
54 _CWORD78__sm_localtransition.h

C_CWORD33_OrthogonalState Class Reference

#include <_CWORD78__sm_orthogonalstate.h>
Inheritance diagram for C_CWORD33_OrthogonalState:



Collaboration diagram for C_CWORD33_OrthogonalState:



Public Member Functions

C_CWORD33_OrthogonalState (std::string f_pName)
C_CWORD33_State * _CWORD33_OnHSMStart (CEventDataPtr f_pEventData)
C_CWORD33_State * _CWORD33_OnHSMStop (CEventDataPtr f_pEventData)
E_CWORD33_Status _CWORD33_AddOrthogonalRegion (C_CWORD33_CompositeState *f_pOrthogonalRegion)
virtual BOOL _CWORD33_HasOrthogoanlRegions ()
virtual C_CWORD33_State * _CWORD33_OnEvent (CEventDataPtr f_pEventData)
virtual E_CWORD33_Status _CWORD33_PrintStates ()
virtual C_CWORD33_State * _CWORD33_GetActiveState ()
virtual E_CWORD33_Status _CWORD33_SetHSM (C_CWORD33_HSM *f_pStatemachine)
E_CWORD33_Status _CWORD33_PrintXML (std::ostringstream &f_strXMLString)
E_CWORD33_Status UpdateHistory ()

Protected Member Functions

virtual E_CWORD33_Status _CWORD33_OnEntry (CEventDataPtr f_pEventData)
virtual E_CWORD33_Status _CWORD33_OnExit (CEventDataPtr f_pEventData)

Additional Inherited Members

Detailed Description

This class implements the additional functionality supported by HSM Orthogonal state. It provides the standard interfaces for adding orthogonal state machines.

Member Function Documentation

E_CWORD33_Status C_CWORD33_OrthogonalState::_CWORD33_AddOrthogonalRegion (C_CWORD33_CompositeState * *f_pOrthogonalRegion*)

_CWORD33_AddOrthogonalRegion This Interface adds orthogonal region in the orthogonal state.

Parameters:

in	*f_pOrthogonalRegion*	C_CWORD33_CompositeState - Orthogonal region object to be associated with state.

Returns:

CurrentState C_CWORD33_State* - Returns current state after operation

virtual C_CWORD33_State* C_CWORD33_OrthogonalState::_CWORD33_GetActiveState ()[virtual]

Summary

pure virtual fuction

Parameters:

None	

Return values:

*C_CWORD33_State**	depend on implement class

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_GetActiveState This interface returns the Active state of the current composite state. In case of non-composite state current state is active state

Returns:

Active state C_CWORD33_State* - Active state

Implements C_CWORD33_State.

virtual BOOL C_CWORD33_OrthogonalState::_CWORD33_HasOrthogoanlRegions ()[virtual]

Brief

This function indicates if the state has sub states.

Parameters:

None	

Return values:

BOOL	if the state has sub states

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

C_CWORD33_OrthogonalState

_CWORD33_HasOrthogoanlRegions This indicates if the state has sub states. It returns TRUE only in the OrthogonalState where this function is overridden

Returns:

TRUE/FASLE BOOL - returns TRUE if it has sub states otherwise returns false.

Reimplemented from C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_OrthogonalState::_CWORD33_OnEntry (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnEntry state initialization can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

virtual C_CWORD33_State* C_CWORD33_OrthogonalState::_CWORD33_OnEvent (CEventDataPtr *f_pEventData*)[virtual]

Brief

This function processes the event.

Parameters:

in	*f_pEventData*	CEventDataPtr - Pointer to event data class

Return values:

*C_CWORD33_State**	Pointer to current C_CWORD33_State object after processing event

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

CEventData, _CWORD33_IsReactionAvailable

_CWORD33_OnEvent This function processes the event. If the reaction for event is available in the current state within eventlist and deferred eventlist then it is consumed in the current state otherwise forwarded to the parent state. Event forwarding is done recursively till either event is consumed or the root state has encountered. This also process the events posted in the reactions recursively till all posted events are cleared.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

state C_CWORD33_State* - Returns current state after the event is processed

Reimplemented from C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_OrthogonalState::_CWORD33_OnExit (CEventDataPtr *f_pEventData*)[protected], [virtual]

_CWORD33_OnExit state cleanup can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

C_CWORD33_State* C_CWORD33_OrthogonalState::_CWORD33_OnHSMStart (CEventDataPtr *f_pEventData*)[virtual]

Brief

Initialize state.

Parameters:

in	*f_pEventData*	CEventDataPtr - Pointer to event data class

Return values:

*C_CWORD33_State**	Pointer to current C_CWORD33_State object after processing event
NULL	error occurs in internal process

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

CEventData, _CWORD33_OnEntry, _CWORD33_OnHSMStop

_CWORD33_OnHSMStart This function is called recursively till the leaf state is reached. This internally calls the Entry function of the current state.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

CurrentState C_CWORD33_State* - Returns current state after operation

Reimplemented from C_CWORD33_State.

C_CWORD33_State* C_CWORD33_OrthogonalState::_CWORD33_OnHSMStop (CEventDataPtr *f_pEventData*)[virtual]

Brief

Clearup state.

Parameters:

in	*f_pEventData*	CEventDataPtr - Pointer to event data class

Return values:

*C_CWORD33_State**	Pointer to current C_CWORD33_State object after processing event
NULL	error occurs in internal process

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

CEventData, _CWORD33_OnExit, _CWORD33_OnHSMStart

_CWORD33_OnHSMStop This function is called recursively till the required parent state is reached. This internally calls the Exit function of the current state.

Parameters:

in	*f_pEventData*	CECEventDataPtr Event data

Returns:

CurrentState C_CWORD33_State* - Returns current state after operation

Reimplemented from C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_OrthogonalState::_CWORD33_PrintStates ()[virtual]

Brief

This logs the state name and event name associated with the state

Parameters:

None	

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusNullPointer	null pointer exception

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_PrintStates This logs the state name and events associated with the state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Reimplemented from C_CWORD33_State.

E_CWORD33_Status C_CWORD33_OrthogonalState::_CWORD33_PrintXML (std::ostringstream & *f_strXMLString*)[virtual]

Brief

Update the State information in the given stream in the form of XML tags

Parameters:

out	*f_strXMLString*	std::ostringstream & - reference to the XML stream

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusNullPointer	NULL pointer exception

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

CEventData, _CWORD33_IsReactionAvailable, C_CWORD33_HSM::_CWORD33_QueueEvent, C_CWORD33_HSM::ProcessEvent

_CWORD33_PrintXML Update the State information in the given stream in the form of XML tags

Parameters:

in	*f_strXMLString*	std::ostringstream & - reference to the XML stream

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Reimplemented from C_CWORD33_State.

virtual E_CWORD33_Status C_CWORD33_OrthogonalState::_CWORD33_SetHSM (C_CWORD33_HSM * *f_pStatemachine*)[virtual]

Brief

This Interface associates the gives statemachine with the current state

Parameters:

in	*f_pStatemachine*	C_CWORD33_HSM* - Statemachine associated with the state.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusNullPointer	NULL pointer exception

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

C_CWORD33_HSM

_CWORD33_SetHSM This Interface associates the gives statemachine with the current state

Parameters:

in	*f_pStatemachine*	C_CWORD33_HSM* - Statemachine associated with the state.

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Reimplemented from C_CWORD33_State.

E_CWORD33_Status C_CWORD33_OrthogonalState::UpdateHistory ()[virtual]

Summary

pure virtual fuction

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

UpdateHistory This function stores the last active state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implements C_CWORD33_State.

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

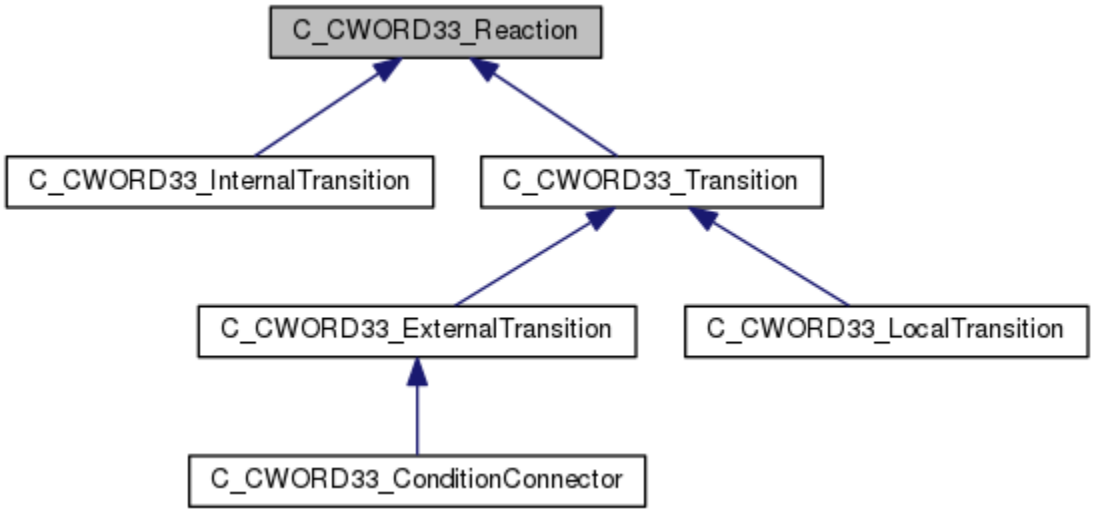
55 _CWORD78__sm_orthogonalstate.h

C_CWORD33_Reaction Class Reference

This class provides the interface for defining a reaction for an event.

#include <_CWORD78__sm_reaction.h>
Inheritance diagram for C_CWORD33_Reaction:

IMAGE



Public Member Functions

virtual C_CWORD33_State * _CWORD33_Reaction (C_CWORD33_State *f_pSourceState,
CEventDataPtr f_pData)=0
C_CWORD33_Reaction ()
virtual ~C_CWORD33_Reaction ()

Public Attributes

UI_8 **m_ucRefCount**

Detailed Description

This class provides the interface for defining a reaction for an event.

Constructor & Destructor Documentation

C_CWORD33_Reaction::C_CWORD33_Reaction ()[inline]

Summary

C_CWORD33_Reaction constructor

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

m_ucRefCount = ***

Classification

Public

Type

sync only

See also:

C_CWORD33_Reaction Constructor

Parameters:

virtual C_CWORD33_Reaction::~C_CWORD33_Reaction ()[inline], [virtual]

Summary

C_CWORD33_Reaction destructor

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

~C_CWORD33_Reaction Destructor

Parameters:

Member Function Documentation

virtual C_CWORD33_State* C_CWORD33_Reaction::_CWORD33_Reaction (C_CWORD33_State * *f_pSourceState*, CEventDataPtr *f_pData*)[pure virtual]

Summary

pure virtual fuction

Parameters:

in	*f_pSourceState*	C_CWORD33_State* f_pSourceState - source state
in	*f_pData*	CEventDataPtr f_pData - event data

Return values:

None	

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_Reaction The reaction for the event has to be implemented in this function

Parameters:

in	*f_pSourceState*	C_CWORD33_State* - Source state in which reaction is being executed
in	*f_pData*	CEventDataPtr - event data

Returns:

ActiveState C_CWORD33_State* - Returns Active state

Implemented in C_CWORD33_ExternalTransition, C_CWORD33_LocalTransition, C_CWORD33_Transition, C_CWORD33_ConditionConnector, and C_CWORD33_InternalTransition.

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

56 _CWORD78__sm_reaction.h

C_CWORD33_ServiceInterface Class Reference

#include <_CWORD78__service_if.h>

Public Member Functions

C_CWORD33_ServiceInterface ()
~C_CWORD33_ServiceInterface ()
HANDLE OpenService (const HANDLE f_hApp, std::string f_sServiceName)
E_CWORD33_Status CloseService (const HANDLE f_hApp, const HANDLE f_hService)
E_CWORD33_Status OpenSession (const HANDLE f_hApp, UI_32 f_uisessionType)
virtual E_CWORD33_Status OpenSessionAcknowledge (const HANDLE f_hApp, HANDLE &hSession)
E_CWORD33_Status CloseSession (const HANDLE f_hService, const HANDLE f_hSession)
E_CWORD33_Status CloseSessionAcknowledge (const HANDLE hClient)

Detailed Description

class: CServiceSession Description: This is base class for service interface classes

---

Constructor & Destructor Documentation

C_CWORD33_ServiceInterface::C_CWORD33_ServiceInterface ()

Brief

Constructor for C_CWORD33_ServiceInterface

Return values:

none	

Preconditions

nopreconditions

Change of internal status

none

Classification

public

Type

none

See also:

~C_CWORD33_ServiceInterface

C_CWORD33_ServiceInterface::~C_CWORD33_ServiceInterface ()

Brief

Destructor for C_CWORD33_ServiceInterface

Return values:

none	

Preconditons

C_CWORD33_ServiceInterface is decleared in constructor

Change of internal status

none

Classification

public

Type

none

See also:

C_CWORD33_ServiceInterface

Member Function Documentation

E_CWORD33_Status C_CWORD33_ServiceInterface::CloseService (const HANDLE *f_hApp*, const HANDLE *f_hService*)

Brief

API to close service.

Parameters:

in	*f_hApp*	HANDLE* - Handle of framework
in	*f_hService*	HANDLE - Handle of the service

Return values:

E_CWORD33_Status	Success or Error

Preconditons

none

Change of internal status

none

Classification

public

Type

sync only

See also:

CWORD33_CloseService

E_CWORD33_Status C_CWORD33_ServiceInterface::CloseSession (const HANDLE *f_hService*, const HANDLE *f_hSession*)

Brief

API to close session.

Parameters:

in	*f_hService*	HANDLE* - Handle of service
in	*hSession*	HANDLE - Session Handle

Return values:

E_CWORD33_Status	Success or Error

Preconditons

none

Change of internal status

none

Classification

public

Type

sync only

See also:

CWORD33_CloseSession

E_CWORD33_Status C_CWORD33_ServiceInterface::CloseSessionAcknowledge (const HANDLE *hClient*)

Brief

API to close session.

Parameters:

in	*f_hService*	HANDLE* - Handle of service
in	*hSession*	HANDLE - Session Handle

Return values:

E_CWORD33_Status	Success or Error

Preconditons

none

Change of internal status

none

Classification

public

Type

sync only

See also:

CWORD33_GetMsgDataOfSize

HANDLE C_CWORD33_ServiceInterface::OpenService (const HANDLE *f_hApp*, std::string *f_sServiceName*)

Brief

API to open Service.

Parameters:

in	*f_hApp*	HANDLE* - Handle of framework
in	*f_sServiceName*	std::string - Service Name

Return values:

HANDLE	

Preconditions

nopreconditions

Change of internal status

none

Classification

public

Type

sync only

See also:

CWORD33_OpenService

E_CWORD33_Status C_CWORD33_ServiceInterface::OpenSession (const HANDLE *f_hApp*, UI_32 *f_uisessionType*)

Brief

APItoopensession

Parameters:

in	*f_hApp*	HANDLE* -frameworkhandle

in	*f_uisessionType*	UI_32 - sessiontype

Return values:

E_CWORD33_Status	Success or Error

Preconditons

none

Change of internal status

none

Classification

public

Type

sync only

See also:

_CWORD33_OpenSessionWithData

virtual E_CWORD33_Status C_CWORD33_ServiceInterface::OpenSessionAcknowledge (const HANDLE *f_hApp*, HANDLE & *hSession*)[virtual]

Brief

API to open session Acknowledge

Parameters:

in	*f_hApp*	HANDLE* - Handle of framework
in	*hSession*	HANDLE - Session Handle

Return values:

E_CWORD33_Status	Success or Error

Preconditons

none

Change of internal status

none

Classification

public

Type

sync only

See also:

none

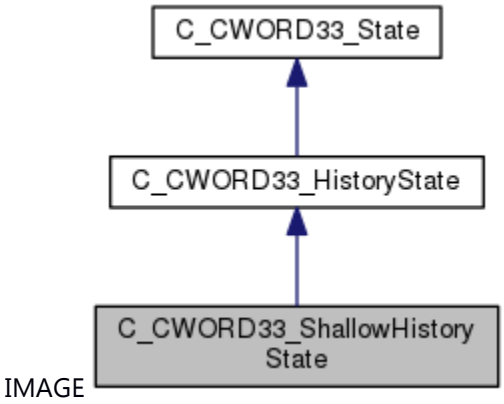
The documentation for this class was generated from the following file:

57 _CWORD78__service_if.h

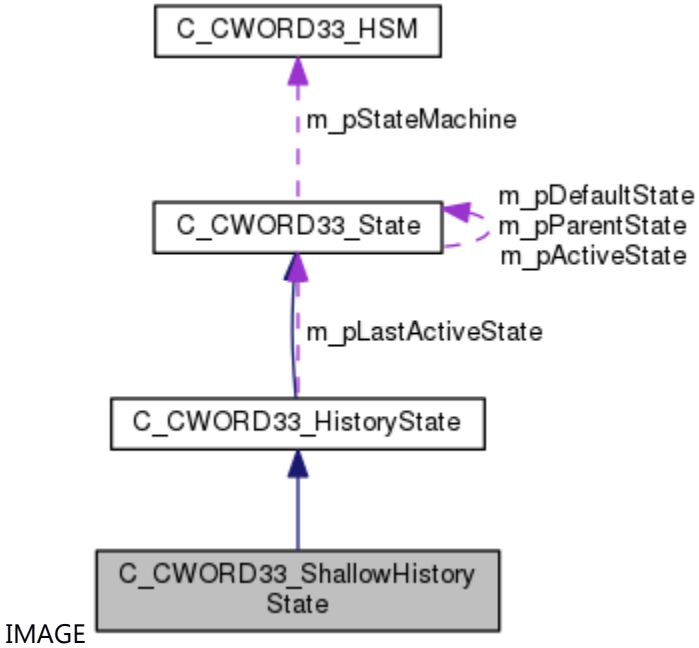
C_CWORD33_ShallowHistoryState Class Reference

This class implements the additional functionality supported by HSM Shallow History state.

#include <_CWORD78__sm_shallowhistorystate.h>
Inheritance diagram for C_CWORD33_ShallowHistoryState:



Collaboration diagram for C_CWORD33_ShallowHistoryState:



Public Member Functions

C_CWORD33_ShallowHistoryState (std::string f_pName)
~C_CWORD33_ShallowHistoryState ()
E_CWORD33_Status UpdateHistory ()

Additional Inherited Members

Detailed Description

This class implements the additional functionality supported by HSM Shallow History state.

Constructor & Destructor Documentation

C_CWORD33_ShallowHistoryState::C_CWORD33_ShallowHistoryState (std::string *f_pName*)

Brief

C_CWORD33_ShallowHistoryState constructor

Parameters:

in	*f_pName*	std::string - Name of the state

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

~C_CWORD33_ShallowHistoryState

C_CWORD33_ShallowHistoryState Parameterized constructor

Parameters:

in	*f_pName*	string - Name of the state

Returns:

none

C_CWORD33_ShallowHistoryState::~C_CWORD33_ShallowHistoryState ()

Brief

C_CWORD33_ShallowHistoryState destructor

Parameters:

None	

Return values:

None	

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

C_CWORD33_ShallowHistoryState::C_CWORD33_ShallowHistoryState

~C_CWORD33_ShallowHistoryState Class destructor

Returns:

none

Member Function Documentation

E_CWORD33_Status C_CWORD33_ShallowHistoryState::UpdateHistory ()[virtual]

Brief

update history infomation

Parameters:

None	

Return values:

e_CWORD33_StatusOK	update history infomation success
e_CWORD33_StatusNullPointer	exception occur when update history infomation

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

UpdateHistory This function stores the last active state

Parameters:

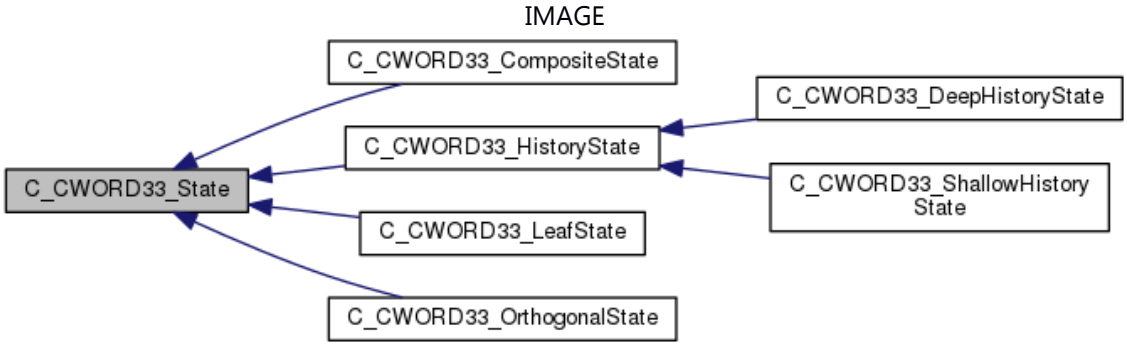
Implements C_CWORD33_HistoryState.

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

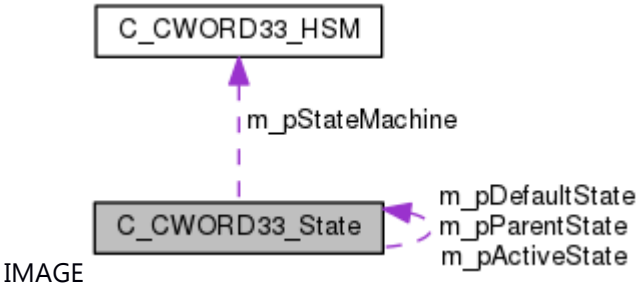
58 _CWORD78__sm_shallowhistorystate.h

C_CWORD33_State Class Reference

#include <_CWORD78__sm_state.h>
Inheritance diagram for C_CWORD33_State:



Collaboration diagram for C_CWORD33_State:



Public Member Functions

C_CWORD33_State (std::string f_pName)
virtual ~C_CWORD33_State ()
E_CWORD33_Status _CWORD33_AddEvent (UI_32 f_uiEventId, C_CWORD33_Reaction *f_pReaction, std::string f_strEventName="")
virtual C_CWORD33_State * _CWORD33_OnEvent (CEventDataPtr f_pEventData)
E_CWORD33_Status _CWORD33_AddDeferredEvent (UI_32 f_uiEventId, std::string f_strEventName="")
E_CWORD33_Status _CWORD33_PostEvent (UI_32 f_uiEventId)
E_CWORD33_Status _CWORD33_PostEvent (CEventDataPtr f_pEventData)
E_CWORD33_Status _CWORD33_RemoveEventFromDeferredEventList (UI_32 f_uiEventId)
virtual BOOL _CWORD33_HasSubStates ()
virtual BOOL _CWORD33_HasOrthogoanlRegions ()
virtual E_CWORD33_Status _CWORD33_PrintStates ()
virtual C_CWORD33_State * _CWORD33_OnHSMStart (CEventDataPtr f_pEventData)
virtual C_CWORD33_State * _CWORD33_OnHSMStop (CEventDataPtr f_pEventData)
virtual BOOL _CWORD33_IsReactionAvailable (UI_32 f_uiEventId)
virtual C_CWORD33_State * _CWORD33_GetActiveState ()=0
virtual E_CWORD33_Status _CWORD33_SetHSM (C_CWORD33_HSM *f_pStatemachine)
virtual HANDLE _CWORD33_GetAppHandle ()

virtual E_CWORD33_Status _CWORD33_PrintXML (std::ostringstream &f_strXMLString)
virtual E_CWORD33_Status UpdateHistory ()=0

Public Attributes

std::string m_strStateName

state name

C_CWORD33_State * m_pParentState

Pointer to the parent state.

C_CWORD33_State * m_pDefaultState

pointer to the default state

C_CWORD33_State * m_pActiveState

pointer to the active state

EventNameList * m_pEventName

pointer to the map that stores the event name against event id

Protected Member Functions

virtual E_CWORD33_Status _CWORD33_OnEntry (CEventDataPtr f_pEventData)=0
virtual E_CWORD33_Status _CWORD33_OnExit (CEventDataPtr f_pEventData)=0

Protected Attributes

EventReactionList * m_pEventList

pointer to the map that stores the reaction against event id

DeferredEventList * m_pDeferredEventList

pointer to the map that stores deferred event ids

EventInfoList * m_pDeferredPostEventList

pointer to the vector that stores the eventinfo of the posted deferred event

C_CWORD33_HSM * m_pStateMachine

pointer to the application statemachine

Detailed Description

This class implements the basic functionality required for HSM state. It provides the standard interfaces for entering, exiting and reacting in a state.

Constructor & Destructor Documentation

C_CWORD33_State::C_CWORD33_State (std::string *f_pName*)

Brief

C_CWORD33_State constructor

Parameters:

in	*f_pName*	std::string - Name of the state

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

public

Type

sync only

See also:

~C_CWORD33_State

C_CWORD33_State Parameterized constructor

Parameters:

in	*f_pName*	string - Name of the state

Returns:

none

virtual C_CWORD33_State::~C_CWORD33_State ()[virtual]

Brief

C_CWORD33_State destructor

Parameters:

None	

Return values:

None	

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

public

Type

sync only

See also:

C_CWORD33_State

~C_CWORD33_State Class destructor

Returns:

none

Member Function Documentation

E_CWORD33_Status C_CWORD33_State::_CWORD33_AddDeferredEvent (UI_32 *f_uiEventId*, std::string *f_strEventName* = "")

Brief

add deferred event information

Parameters:

in	*f_uiEventId*	UI_32 f_uiEventId - event ID
in	*f_strEventName*	std::string f_strEventName - event name

Return values:

e_CWORD33_StatusOK	add deferred event success
e_CWORD33_StatusNullPointer	exception occur when add deferred event

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_AddDeferredEvent Associates the deferred event id with the reaction in the state. When the event is posted to the state the event is deferred and stored in the state. In case of implicit recall of the deferred events, events are processed before exiting the state. The deferred events can also be recalled explicitly in the state This also adds the event name to the map which is used for debugging.

Parameters:

in	*f_uiEventId*	UI_32 - Event id of the event to be added in the state
in	*f_strEventName*	string - Name of the event (used for debugging purpose only)

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

E_CWORD33_Status C_CWORD33_State::_CWORD33_AddEvent (UI_32 *f_uiEventId*, C_CWORD33_Reaction * *f_pReaction*, std::string *f_strEventName* = "")

Brief

add event information

Parameters:

in	*f_uiEventId*	UI_32 f_uiEventId - event ID
in	*f_pReaction*	C_CWORD33_Reaction *f_pReaction - event reaction
in	*f_strEventName*	std::string f_strEventName - event name

Return values:

e_CWORD33_StatusOK	add event success
e_CWORD33_StatusNullPointer	exception occur when add event

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_AddEvent Associates the event id with the reaction in the state. When the event is posted to the state the associated reaction is executed. This also adds the event name to the map which is used for debugging.

Parameters:

in	*f_uiEventId*	UI_32 - Event id of the event to be added in the state
in	*f_pReaction*	C_CWORD33_Reaction* - Reaction to be associated with the with event id in the state
in	*f_strEventName*	string - Name of the event (used for debugging purpose only)

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

virtual C_CWORD33_State* C_CWORD33_State::_CWORD33_GetActiveState ()[pure virtual]

Summary

pure virtual fuction

Parameters:

None	

Return values:

*C_CWORD33_State**	depend on implement class

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_GetActiveState This interface returns the Active state of the current composite state. In case of non-composite state current state is active state

Returns:

Active state C_CWORD33_State* - Active state

Implemented in C_CWORD33_CompositeState, C_CWORD33_OrthogonalState, C_CWORD33_HistoryState, and C_CWORD33_LeafState.

virtual HANDLE C_CWORD33_State::_CWORD33_GetAppHandle ()[virtual]

Brief

Get application handle.

Parameters:

None	

Return values:

HANDLE	

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

C_CWORD33_HSM::_CWORD33_GetAppHandle

_CWORD33_GetAppHandle This interface returns the application handle

Returns:

HANDLE HANDLE - Application Handle

virtual BOOL C_CWORD33_State::_CWORD33_HasOrthogoanlRegions ()[virtual]

Brief

This function indicates if the state has sub states.

Parameters:

None	

Return values:

BOOL	if the state has sub states

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

C_CWORD33_OrthogonalState

_CWORD33_HasOrthogoanlRegions This indicates if the state has sub states. It returns TRUE only in the OrthogonalState where this function is overridden

Returns:

TRUE/FASLE BOOL - returns TRUE if it has sub states otherwise returns false.

Reimplemented in C_CWORD33_OrthogonalState.

virtual BOOL C_CWORD33_State::_CWORD33_HasSubStates ()[virtual]

Brief

This function indicates if the state has sub states.

Parameters:

None	

Return values:

BOOL	if the state has sub states

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

C_CWORD33_CompositeState:: _CWORD33_HasSubStates

_CWORD33_HasSubStates This indicates if the state has sub states. It returns TRUE only in the CompositeState where this function is overridden

Returns:

TRUE/FASLE BOOL - returns TRUE if it has sub states otherwise returns false.

Reimplemented in C_CWORD33_CompositeState.

virtual BOOL C_CWORD33_State::_CWORD33_IsReactionAvailable (UI_32 *f_uiEventId*)[virtual]

Brief

Check the availability of event.

Parameters:

in	*f_uiEventId*	UI_32 - event ID

Return values:

TRUE	reaction is available for this event
FALSE	reaction is not available for this event

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_PostEvent

_CWORD33_IsReactionAvailable

Parameters:

in	*f_uiEventId*	UI_32 - Event identifier

Returns:

TRUE/FASLE BOOL - returns TRUE if it reaction is available for this event

virtual E_CWORD33_Status C_CWORD33_State::_CWORD33_OnEntry (CEventDataPtr *f_pEventData*)[protected], [pure virtual]

_CWORD33_OnEntry This is pure virtual function implemented by the derived classes of the application. state initialization can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implemented in C_CWORD33_CompositeState, C_CWORD33_OrthogonalState, C_CWORD33_HistoryState, and C_CWORD33_LeafState.

virtual C_CWORD33_State* C_CWORD33_State::_CWORD33_OnEvent (CEventDataPtr *f_pEventData*)[virtual]

Brief

This function processes the event.

Parameters:

in	*f_pEventData*	CEventDataPtr - Pointer to event data class

Return values:

*C_CWORD33_State**	Pointer to current C_CWORD33_State object after processing event

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

CEventData, _CWORD33_IsReactionAvailable

_CWORD33_OnEvent This function processes the event. If the reaction for event is available in the current state within eventlist and deferred eventlist then it is consumed in the current state otherwise forwarded to the parent state. Event forwarding is done recursively till either event is consumed or the root state has encountered. This also process the events posted in the reactions recursively till all posted events are cleared.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

state C_CWORD33_State* - Returns current state after the event is processed

Reimplemented in C_CWORD33_OrthogonalState.

virtual E_CWORD33_Status C_CWORD33_State::_CWORD33_OnExit (CEventDataPtr *f_pEventData*)[protected], [pure virtual]

_CWORD33_OnExit This is pure virtual function implemented by the derived classes of the application. state cleanup can be performed in this function.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implemented in C_CWORD33_CompositeState, C_CWORD33_OrthogonalState, C_CWORD33_HistoryState, and C_CWORD33_LeafState.

virtual C_CWORD33_State* C_CWORD33_State::_CWORD33_OnHSMStart (CEventDataPtr *f_pEventData*)[virtual]

Brief

Initialize state.

Parameters:

in	*f_pEventData*	CEventDataPtr - Pointer to event data class

Return values:

*C_CWORD33_State**	Pointer to current C_CWORD33_State object after processing event
NULL	error occurs in internal process

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

CEventData, _CWORD33_OnEntry, _CWORD33_OnHSMStop

_CWORD33_OnHSMStart This function is called recursively till the leaf state is reached. This internally calls the Entry function of the current state.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

CurrentState C_CWORD33_State* - Returns current state after operation

Reimplemented in C_CWORD33_HistoryState, and C_CWORD33_OrthogonalState.

virtual C_CWORD33_State* C_CWORD33_State::_CWORD33_OnHSMStop (CEventDataPtr *f_pEventData*)[virtual]

Brief

Clearup state.

Parameters:

in	*f_pEventData*	CEventDataPtr - Pointer to event data class

Return values:

*C_CWORD33_State**	Pointer to current C_CWORD33_State object after processing event
NULL	error occurs in internal process

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

CEventData, _CWORD33_OnExit, _CWORD33_OnHSMStart

_CWORD33_OnHSMStop This function is called recursively till the required parent state is reached. This internally calls the Exit function of the current state.

Parameters:

in	*f_pEventData*	CECEventDataPtr Event data

Returns:

CurrentState C_CWORD33_State* - Returns current state after operation

Reimplemented in C_CWORD33_HistoryState, and C_CWORD33_OrthogonalState.

E_CWORD33_Status C_CWORD33_State::_CWORD33_PostEvent (UI_32 *f_uiEventId*)

Brief

This function creates new eventdata object and add it to event queue of the state machine.

Parameters:

in	*f_uiEventId*	UI_32 f_uiEventId - event ID

Return values:

e_CWORD33_StatusOK	post event success
e_CWORD33_StatusNullPointer	exception occur when post event

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_PostEvent(CEventDataPtr), CEventData, C_CWORD33_HSM::_CWORD33_QueueEvent, C_CWORD33_HSM::ProcessEvent

_CWORD33_PostEvent This function creates new eventdata object and add the to event queue of the state. The events are posted in the reaction which are executed in the state. The event queue is processed once the execution of the reaction is completed.

Parameters:

in	*f_uiEventId*	UI_32 - Event Id

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

E_CWORD33_Status C_CWORD33_State::_CWORD33_PostEvent (CEventDataPtr *f_pEventData*)

Brief

This function adds event data to event queue of the state machine.

Parameters:

in	*f_pEventData*	CEventDataPtr - Pointer to event data class

Return values:

e_CWORD33_StatusOK	post event success
e_CWORD33_StatusNullPointer	exception occur when post event

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

CEventData, _CWORD33_IsReactionAvailable,
C_CWORD33_HSM::_CWORD33_QueueEvent, C_CWORD33_HSM::ProcessEvent

_CWORD33_PostEvent This function adds the event queue of the state. The events are posted in the reaction which are executed in the state. The event queue is processed once the execution of the reaction is completed.

Parameters:

in	*f_pEventData*	CEventDataPtr - Event data

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

virtual E_CWORD33_Status C_CWORD33_State::_CWORD33_PrintStates ()[virtual]

Brief

This logs the state name and event name associated with the state

Parameters:

None	

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusNullPointer	null pointer exception

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_PrintStates This logs the state name and events associated with the state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Reimplemented in C_CWORD33_OrthogonalState, and C_CWORD33_CompositeState.

virtual E_CWORD33_Status C_CWORD33_State::_CWORD33_PrintXML (std::ostringstream & *f_strXMLString*)[virtual]

Brief

Update the State information in the given stream in the form of XML tags

Parameters:

out	*f_strXMLString*	std::ostringstream & - reference to the XML stream

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusNullPointer	NULL pointer exception

Prerequisite

Calls constructor to successfully create object.
State machine of current state in the applicaton is setted in _CWORD33_SetHSM.

Change of internal state

None

Classification

Public

Type

sync only

See also:

CEventData, _CWORD33_IsReactionAvailable,
C_CWORD33_HSM::_CWORD33_QueueEvent, C_CWORD33_HSM::ProcessEvent

_CWORD33_PrintXML Update the State information in the given stream in the form of XML tags

Parameters:

in	*f_strXMLString*	std::ostringstream & - reference to the XML stream

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Reimplemented in C_CWORD33_CompositeState, and C_CWORD33_OrthogonalState.

E_CWORD33_Status C_CWORD33_State::_CWORD33_RemoveEventFromDeferredEventList (UI_32 *f_uiEventId*)

Brief

delete event from deferred event list

Parameters:

in	*f_uiEventId*	UI_32 f_uiEventId - event ID

Return values:

e_CWORD33_StatusOK	delete deferred event success
e_CWORD33_StatusInvldID	invalid event ID
e_CWORD33_StatusNullPointer	exception occur when delete deferred event

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

_CWORD33_RemoveEventFromDeferredEventList This function removes the event from the posted deferred queue list of the state.

Parameters:

in	*f_uiEventId*	UI_32 - Event id

Returns:

E_CWORD33_Status E_CWORD33_Status - e_CWORD33_StatusOK if event removed
e_CWORD33_StatusInvldID if event not found in list

virtual E_CWORD33_Status C_CWORD33_State::_CWORD33_SetHSM (C_CWORD33_HSM * *f_pStatemachine*)[virtual]

Brief

This Interface associates the gives statemachine with the current state

Parameters:

in	*f_pStatemachine*	C_CWORD33_HSM* - Statemachine associated with the state.

Return values:

e_CWORD33_StatusOK	success
e_CWORD33_StatusNullPointer	NULL pointer exception

Prerequisite

Calls constructor to successfully create object.

Change of internal state

None

Classification

Public

Type

sync only

See also:

C_CWORD33_HSM

_CWORD33_SetHSM This Interface associates the gives statemachine with the current state

Parameters:

in	*f_pStatemachine*	C_CWORD33_HSM* - Statemachine associated with the state.

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Reimplemented in C_CWORD33_CompositeState, and C_CWORD33_OrthogonalState.

virtual E_CWORD33_Status C_CWORD33_State::UpdateHistory ()[pure virtual]

Summary

pure virtual fuction

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

UpdateHistory This function stores the last active state

Returns:

E_CWORD33_Status E_CWORD33_Status - Returns status of operation

Implemented in C_CWORD33_OrthogonalState, C_CWORD33_CompositeState, C_CWORD33_HistoryState, C_CWORD33_ShallowHistoryState, C_CWORD33_DeepHistoryState, and C_CWORD33_LeafState.

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

59 _CWORD78__sm_state.h

C_CWORD33_SyncData Class Reference

this file has the C_CWORD33_SyncData class definitions

#include <_CWORD78__framework_sync.h>

Public Member Functions

E_CWORD33_Status _CWORD33_StartNotificationSync (HANDLE hApp)
E_CWORD33_Status _CWORD33_StopNotificationSync (HANDLE hApp)
E_CWORD33_Status _CWORD33_SubscribeNotificationWithDataSync (const std::string &f_cNotification)
E_CWORD33_Status _CWORD33_UnSubscribeNotificationWithDataSync (const std::string &f_cNotification)
E_CWORD33_Status _CWORD33_GetSyncNotificationData (const std::string &f_cNotification, PVOID f_pBuffer, UI_16 f_nBufferSize)
UI_32 _CWORD33_GetSyncDataSize (const std::string &f_cNotification)

Static Public Member Functions

static C_CWORD33_SyncData * _CWORD33_GetSyncDataInstance ()
static E_CWORD33_Status _CWORD33_ReleaseSyncDataInstance ()

Detailed Description

this file has the C_CWORD33_SyncData class definitions

Brief Introduction

60 This class provids some functions to handle the synchronization data as per the notification.

Member Function Documentation

static C_CWORD33_SyncData* C_CWORD33_SyncData::_CWORD33_GetSyncDataInstance ()[static]

Brief

This function is used to get the singleton instance of class.

Parameters:

None	

Return values:

C_CWORD33_SyncData	- singleton instance of class

Preconditons

-None

Change of internal status

None

Classification

Public

See also:

C_CWORD33_SyncData::C_CWORD33_SyncData

UI_32 C_CWORD33_SyncData::_CWORD33_GetSyncDataSize (const std::string & *f_cNotification*)

Brief

API to get the size of synchronization notification data.

Parameters:

in	*f_cNotification*	const std::string - Notification to be subscribed.

Return values:

UI_32	- size of notification data

Preconditons

-Instance is created successfully.

Change of internal status

-None

Classification

Public

See also:

None

E_CWORD33_Status C_CWORD33_SyncData::_CWORD33_GetSyncNotificationData (const std::string & *f_cNotification*, PVOID *f_pBuffer*, UI_16 *f_nBufferSize*)

Brief

This function is used to get the synchronization notification data for a particular notification.

Parameters:

in	*f_cNotification*	std::string - Notification for which data is required
in	*f_pBuffer*	PVOID - Buffer for the synchronization data
in	*f_nBufferSize*	UI_16 - size of notification data buffer

Return values:

e_CWORD33_StatusOK	- Success
e_CWORD33_StatusNullPointer	- Null pointor
e_CWORD33_StatusFail	- failure

Preconditons

none

Change of internal status

none

Classification

Public

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

None

static E_CWORD33_Status C_CWORD33_SyncData::_CWORD33_ReleaseSyncDataInstance ()[static]

Brief

This function is used to release the instance of class.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	- success
except	e_CWORD33_StatusOK - failure

Preconditons

-None

Change of internal status

None

Classification

Public

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

C_CWORD33_SyncData::C_CWORD33_SyncData

E_CWORD33_Status C_CWORD33_SyncData::_CWORD33_StartNotificationSync (HANDLE *hApp*)

Brief

This method is used by the application to start the notification synchronization thread.

Parameters:

in	*hApp*	HANDLE - Application Handle.

Return values:

e_CWORD33_StatusOK	- On Success
e_CWORD33_StatusThreadAlreadyRunning	- If thread already running
e_CWORD33_StatusNullPointer	- Null Handle
e_CWORD33_StatusInvldHandle	- Invalid Handle

Preconditons

The function _CWORD33_ReleaseSyncDataInstance is successed, object is created successfully.

Change of internal status

-m_hSyncThreadMsgQHandle saves sync thread handle.

Classification

Public

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

None

E_CWORD33_Status C_CWORD33_SyncData::_CWORD33_StopNotificationSync (HANDLE *hApp*)

Brief

This method is used to stop the synchronization notification thread.

Parameters:

in	*hApp*	HANDLE - Application Handle.

Return values:

e_CWORD33_StatusOK	- On Success
e_CWORD33_StatusThreadNotExist	- If thread already running
e_CWORD33_StatusNullPointer	- Null Handle
e_CWORD33_StatusInvldHandle	- Invalid Handle

Preconditons

-The function _CWORD33_StartNotificationSync is successed.

Change of internal status

m_hSyncThreadMsgQHandle will be delete.

Classification

Public

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

None

E_CWORD33_Status

C_CWORD33_SyncData::_CWORD33_SubscribeNotificationWithDataSync (const std::string & *f_cNotification*)

Brief

API to subscribe to a synchronization notification data.

Parameters:

in	*f_cNotification*	std::string - Notification to be subscribed.

Return values:

e_CWORD33_StatusOK	- On Success
e_CWORD33_StatusFail	- On failure

Preconditons

none

Change of internal status

none

Classification

Public

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
e_CWORD33_StatusFail:Failure

See also:

None

E_CWORD33_Status

C_CWORD33_SyncData::_CWORD33_UnSubscribeNotificationWithDataSync (const std::string & *f_cNotification*)

Brief

API to unsubscribe to a synchronization notification data.

Parameters:

in	*f_cNotification*	std::string - Notification to be subscribed.

Return values:

e_CWORD33_StatusOK	- On Success
e_CWORD33_StatusFail	- On failure

Preconditons

the _CWORD33_SubscribeNotificationWithDataSync result is success.

Change of internal status

The synchronization data's pointor of the special notification will be remove and delete.
The synchronization data of the special notification will delete from m_mSyncDataMap.

Classification

Public

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
e_CWORD33_StatusFail:Failure

See also:

None

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

61 _CWORD78__framework_sync.h

CWORD33::framework::C_CWORD33_ThreadPriorities Class Reference

This class parses and stores thread priorities.

#include <_CWORD78__thread_priority.h>

Public Member Functions

C_CWORD33_ThreadPriorities ()
virtual ~C_CWORD33_ThreadPriorities ()

Static Public Member Functions

static SI_32 GetPriority (const std::string &f_cThreadName)
static E_CWORD33_Status AddPriority (const std::string &f_cThreadName, SI_32 f_si32Priority)
static VOID PrintPriorites ()
static E_CWORD33_Status ParseThreadArguments (PCHAR f_cArgumentValue)

Detailed Description

This class parses and stores thread priorities.

Constructor & Destructor Documentation

CWORD33::framework::C_CWORD33_ThreadPriorities::C_CWORD33_ThreadPriorities ()

Brief

C_CWORD33_ThreadPriorities constructor

Parameters:

None	

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

virtual _CWORD33_::framework::C_CWORD33_ThreadPriorities::~C_CWORD33_ThreadPriorities ()[virtual]

Brief

C_CWORD33_ThreadPriorities destructor

Parameters:

None	

Return values:

None	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

sync only

See also:

Member Function Documentation

static E_CWORD33_Status _CWORD33_::framework::C_CWORD33_ThreadPriorities::AddPriority (const std::string & *f_cThreadName*, SI_32 *f_si32Priority*)[static]

Summary

set thread Priority

Parameters:

in	*f_cThreadName*	const std::string &f_cThreadName - thread name
in	*f_si32Priority*	SI_32 f_si32Priority - Priority

Return values:

e_CWORD33_StatusOK	set Priority success

Preconditions

None

Change of the internal state

ms_mapThreadPritories[f_cThreadName] = f_si32Priority

Classification

public

Type

sync only

See also:

static SI_32 _CWORD33_::framework::C_CWORD33_ThreadPriorities::GetPriority (const std::string & *f_cThreadName*)[static]

Brief

Get the priority of the thread. Priority of the thread was added by AddPriority(), when parsing command line arguments.

Parameters:

in	*f_cThreadName*	const std::string& - Name of thread

Return values:

SI_32	Returns the priority of a thread.
_CWORD33__PRIORITY_NOT_FOUND	if specified thread name not found

_CWORD33__PRIORITY_NOT_FOUND
#define _CWORD33__PRIORITY_NOT_FOUND -1 // if specified thread name not found

Prerequisite

Call AddPriority() and set priority.

Change of internal state

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

Classification

Public

Type

sync only

See also:

AddPriority

static E_CWORD33_Status _CWORD33_::framework::C_CWORD33_ThreadPriorities::ParseThreadArguments (PCHAR *f_cArgumentValue*)[static]

Summary

analyse thread param

Parameters:

in	*f_cArgumentValue*	PCHAR f_cArgumentValue - thread param string

Return values:

e_CWORD33_StatusOK	parse thread param success
e_CWORD33_StatusFail	parse thread param failed

Preconditions

None

Change of the internal state

None

Classification

public

Type

sync only

See also:

getsubopt, AddPriority

static VOID _CWORD33_::framework::C_CWORD33_ThreadPriorities::PrintPriorites ()[static]

Summary

print thread Priority

Parameters:

None	

Return values:

None	

Preconditions

None

Change of the internal state

None

Classification

public

Type

sync only

See also:

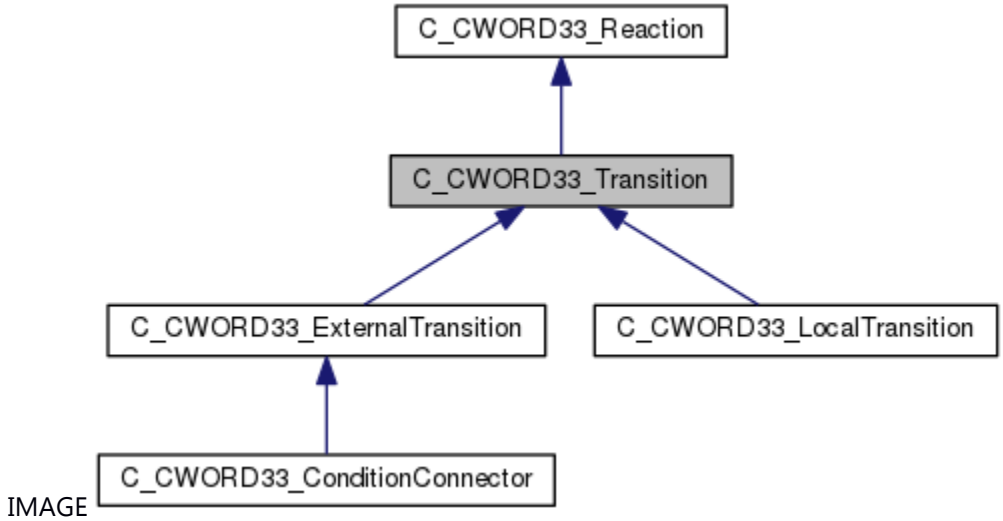
The documentation for this class was generated from the following file:

62 _CWORD78__thread_priority.h

C_CWORD33_Transition Class Reference

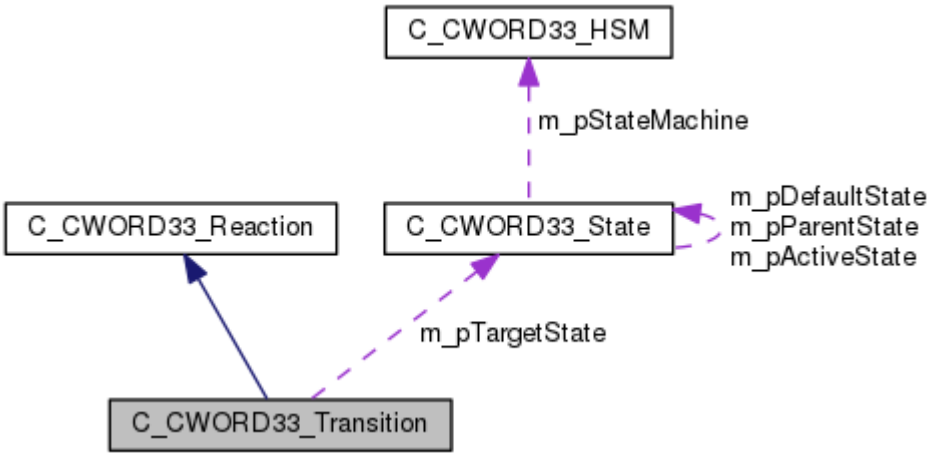
This class provides the interface for reacting local transition and external transition.

#include <_CWORD78__sm_transition.h>
Inheritance diagram for C_CWORD33_Transition:



IMAGE

Collaboration diagram for C_CWORD33_Transition:



IMAGE

Public Member Functions

C_CWORD33_Transition (C_CWORD33_State *f_pTargetState)
virtual ~C_CWORD33_Transition ()
virtual C_CWORD33_State * _CWORD33_Reaction (C_CWORD33_State *f_pSourceState,
CEventDataPtr f_pData)=0

Protected Attributes

C_CWORD33_State * m_pTargetState

Pointer to the target state of transition.

Additional Inherited Members

Detailed Description

This class provides the interface for reacting local transition and external transition.

Constructor & Destructor Documentation

C_CWORD33_Transition::C_CWORD33_Transition (C_CWORD33_State * *f_pTargetState*)

Summary

C_CWORD33_Transition constructor

Parameters:

in	*f_pTargetState*	C_CWORD33_State *f_pTargetState - set m_pTargetState value

Return values:

None	

Preconditions

None.

Change of the internal state

m_pTargetState = f_pTargetState

Classification

Public

Type

sync only

See also:

virtual C_CWORD33_Transition::~C_CWORD33_Transition ()[virtual]

Summary

C_CWORD33_Transition destructor

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

Member Function Documentation

virtual C_CWORD33_State* C_CWORD33_Transition::_CWORD33_Reaction (C_CWORD33_State * *f_pSourceState*, CEventDataPtr *f_pData*)[pure virtual]

Summary

virtual fuction

Parameters:

in	*f_pSourceState*	C_CWORD33_State *f_pSourceState - source state
in	*f_pData*	CEventDataPtr f_pData - event data

Return values:

*C_CWORD33_State**	depend on implement class

Preconditions

None.

Change of the internal state

None

Classification

Public

Type

sync only

See also:

Implements C_CWORD33_Reaction.

Implemented in C_CWORD33_ExternalTransition, C_CWORD33_LocalTransition, and C_CWORD33_ConditionConnector.

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

63 _CWORD78__sm_transition.h

C_CWORD33_Version Class Reference

Version Info Class.

#include <ns_version_if.h>

Public Member Functions

C_CWORD33_Version ()
C_CWORD33_Version (UI_16 major, UI_16 minor, UI_16 revision)
virtual ~C_CWORD33_Version ()
BOOL operator== (const C_CWORD33_Version &f_test_i)
PCSTR VersionStr ()
PCSTR DateStr ()
UI_32 Signature () const
UI_32 StrucVersion () const
UI_32 ProductVersion () const
UI_32 Date () const
UI_16 Major () const
UI_16 Minor () const
PCSTR Product () const
PCSTR Build () const
UI_16 Revision () const

Friends

BOOL operator== (C_CWORD33_Version &a, C_CWORD33_Version &b)

Detailed Description

Version Info Class.

Brief Introduction

This class provides functions to get various information of the version.

Constructor & Destructor Documentation

C_CWORD33_Version::C_CWORD33_Version ()

Brief

Construct a C_CWORD33_Version object.

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

~C_CWORD33_Version

C_CWORD33_Version::C_CWORD33_Version (UI_16 *major*, UI_16 *minor*, UI_16 *revision*)

Brief

Constructs a C_CWORD33_Version object with parameters.

Parameters:

in	*major*	UI_16 - Major number of the application
in	*minor*	UI_16 - Minor number of the application
in	*revision*	UI_16 - Revision number of the application

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

~C_CWORD33_Version

virtual C_CWORD33_Version::~C_CWORD33_Version ()[virtual]

Brief

Destruct the C_CWORD33_Version object.

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

C_CWORD33_Version

Member Function Documentation

PCSTR C_CWORD33_Version::Build () const

Brief

Get the building version string

Return values:

the	building version string

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

UI_32 C_CWORD33_Version::Date () const

Brief

Get the date as number

Return values:

the	date number

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

PCSTR C_CWORD33_Version::DateStr ()

Brief

Get the date as string

Return values:

the	date string

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

UI_16 C_CWORD33_Version::Major () const

Brief

Get the major verion number

Return values:

the	major version number

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

UI_16 C_CWORD33_Version::Minor () const

Brief

Get the minor version number

Return values:

the	minor version number

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

BOOL C_CWORD33_Version::operator== (const C_CWORD33_Version & *f_test_i*)

Brief

Get the data number

Parameters:

in	*f_test_i*	C_CWORD33_Version - the C_CWORD33_Version object that will be compared against this obj.

Return values:

TRUE	Match, FALSE: Does not Match

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

PCSTR C_CWORD33_Version::Product () const

Brief

Get the unique product identifier

Return values:

the	revision number

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

UI_32 C_CWORD33_Version::ProductVersion () const

Brief

Get the product version

Return values:

the	product version

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

UI_16 C_CWORD33_Version::Revision () const

Brief

Get the revision number

Return values:

the	revision number

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

C_CWORD33_Version

UI_32 C_CWORD33_Version::Signature () const

Brief

Get the signature number

Return values:

the	signature

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

UI_32 C_CWORD33_Version::StrucVersion () const

Brief

Get the structure version number

Return values:

the	structure version number

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

PCSTR C_CWORD33_Version::VersionStr ()

Brief

Get the version as string

Return values:

the	vision string

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

Friends And Related Function Documentation

BOOL operator== (C_CWORD33_Version & *a*, C_CWORD33_Version & *b*)[friend]

Brief

Get the data number

Parameters:

in	*a*	C_CWORD33_Version - the C_CWORD33_Version object that will be compared against b.
in	*b*	C_CWORD33_Version - the C_CWORD33_Version object that will be compared against a.

Return values:

TRUE	Match, FALSE: Does not Match

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

operator ==

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

64 ns_version_if.h

C_CWORD77_Data Class Reference

Public Member Functions

C_CWORD77_Data ()
instance to C_CWORD77_Data
~C_CWORD77_Data ()
VOID SetRequData (const _CWORD77_DataPoolKey &f_DataPoolKey, UI_32 f_uiSize, PVOID f_pData)
VOID SetRequData (UI_32 f_uiMsgId, UI_32 f_uiSize, PVOID f_pData)
VOID SetRespoData (const _CWORD77_DataPoolKey &f_DataPoolKey, UI_32 f_uiSize, const PVOID f_pData)
VOID SetRespoData (UI_32 f_uiMsgId, UI_32 f_uiSize, const PVOID f_pData)
E_CWORD33_Status GetRequData (const _CWORD77_DataPoolKey &f_DataPoolKey, UI_32 &f_uiSize, PVOID &f_pData)
E_CWORD33_Status GetRequData (UI_32 f_uiMsgId, UI_32 &f_uiSize, PVOID &f_pData)
E_CWORD33_Status GetRespoData (const _CWORD77_DataPoolKey &f_DataPoolKey, UI_32 &f_uiSize, PVOID &f_pData)
E_CWORD33_Status GetRespoData (UI_32 f_uiMsgId, UI_32 &f_uiSize, PVOID &f_pData)

Static Public Member Functions

static C_CWORD77_Data * GetInstance ()
static void DeleteInstance ()

Constructor & Destructor Documentation

C_CWORD77_Data::C_CWORD77_Data ()

instance to C_CWORD77_Data

C_CWORD77_Data Constructor for C_CWORD77_Data

Parameters:

None	

Returns:

None

C_CWORD77_Data::~C_CWORD77_Data ()

C_CWORD77_Data Destructor for C_CWORD77_Data

Parameters:

None	

Returns:

None

Member Function Documentation

static void C_CWORD77_Data::DeleteInstance ()[static]

DeleteInstance API to delete single instance of C_CWORD77_DataPool

Parameters:

None	

Returns:

None

static C_CWORD77_Data* C_CWORD77_Data::GetInstance ()[static]

GetInstance API to get single instance of C_CWORD77_Data

Parameters:

None	

Returns:

Pointer to single instance of C_CWORD77_Data

E_CWORD33_Status C_CWORD77_Data::GetRequData (const _CWORD77_DataPoolKey & *f_DataPoolKey*, UI_32 & *f_uiSize*, PVOID & *f_pData*)

GetReqDataIn_CWORD77_DataPool API to get data associated with request from *CWORD77* Data Pool

Parameters:

in	*f_DataPoolKey*	_CWORD77_DataPoolKey - Data pool key
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	PVOID - void pointer to data

Returns:

E_CWORD33_Status - Succes or Error

E_CWORD33_Status C_CWORD77_Data::GetRequData (UI_32 *f_uiMsgId*, UI_32 & *f_uiSize*, PVOID & *f_pData*)

GetReqDataIn_CWORD77_DataPool API to get data associated with request from *CWORD77* Data Pool Depricated. Do not use.

Parameters:

in	*f_uiMsgId*	UI_32 - MsgId
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	PVOID - void pointer to data

Returns:

E_CWORD33_Status - Succes or Error

E_CWORD33_Status C_CWORD77_Data::GetRespoData (const _CWORD77_DataPoolKey & *f_DataPoolKey*, UI_32 & *f_uiSize*, PVOID & *f_pData*)

GetRespDataFrom_CWORD77_DataPool API to get data associated with response from *CWORD77* Data Pool

Parameters:

in	*f_DataPoolKey*	_CWORD77_DataPoolKey - Data pool key
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	PVOID - void pointer to data

Returns:

E_CWORD33_Status - Succes or Error

E_CWORD33_Status C_CWORD77_Data::GetRespoData (UI_32 *f_uiMsgId*, UI_32 & *f_uiSize*, PVOID & *f_pData*)

GetRespDataFrom_CWORD77_DataPool API to get data associated with response from *CWORD77* Data Pool Depricated. Do not use.

Parameters:

in	*f_uiMsgId*	UI_32 - MsgId
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	PVOID - void pointer to data

Returns:

E_CWORD33_Status - Succes or Error

VOID C_CWORD77_Data::SetRequData (const _CWORD77_DataPoolKey & *f_DataPoolKey*, UI_32 *f_uiSize*, PVOID *f_pData*)

SetReqDataIn_CWORD77_DataPool API to set data associated with request into *CWORD77* Data Pool

Parameters:

in	*f_DataPoolKey*	_CWORD77_DataPoolKey - Data pool key
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	PVOID - void pointer to data

Returns:

VOID - None

VOID C_CWORD77_Data::SetRequData (UI_32 *f_uiMsgId*, UI_32 *f_uiSize*, PVOID *f_pData*)

SetReqDataIn_CWORD77_DataPool API to set data associated with request into *CWORD77* Data Pool Depricated. Do not use.

Parameters:

in	*f_uiMsgId*	UI_32 - MsgId
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	PVOID - void pointer to data

Returns:

VOID - None

VOID C_CWORD77_Data::SetRespoData (const _CWORD77_DataPoolKey & *f_DataPoolKey*, UI_32 *f_uiSize*, const PVOID *f_pData*)

SetRespDataIn_CWORD77_DataPool API to set data associated with response into *CWORD77* Data Pool

Parameters:

in	*f_DataPoolKey*	_CWORD77_DataPoolKey - Data pool key
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	PVOID - void pointer to data

Returns:

VOID - None

VOID C_CWORD77_Data::SetRespoData (UI_32 *f_uiMsgId*, UI_32 *f_uiSize*, const PVOID *f_pData*)

SetRespDataIn_CWORD77_DataPool API to set data associated with response into *CWORD77* Data Pool Depricated. Do not use.

Parameters:

in	*f_uiMsgId*	UI_32 - MsgId
in	*f_uiSize*	UI_32 - size of the data
in	*f_pData*	PVOID - void pointer to data

Returns:

VOID - None

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

65 ns_CWORD77_data_pool_table.h

C_CWORD77_DataPool Class Reference

#include <ns__CWORD77__data_pool_table.h>

Public Member Functions

C_CWORD77_DataPool ()
~C_CWORD77_DataPool ()
template<class T > void SetReqArrayData (UI_32 VarName, EDataType DataType, T *Array, UI_32 ArraySize)
template<class T > void SetRespArrayData (UI_32 VarName, EDataType DataType, T *Array, UI_32 ArraySize)
template<class T > void GetReqArrayData (UI_32 VarName, T *Array, UI_32 &ArraySize)
template<class T > void GetRespArrayData (UI_32 VarName, T *Array, UI_32 &ArraySize)
template<class T > void SetReqData (UI_32 VarName, EDataType DataType, T Array)
template<class T > void SetRespData (UI_32 VarName, EDataType DataType, T Array)
template<class T > T GetReqData (UI_32 VarName)
template<class T > T GetRespData (UI_32 VarName)
void SetReqArrayStringData (UI_32 VarName, EDataType VarType, std::string DataValue[], UI_32 size)
void SetRespArrayStringData (UI_32 VarName, EDataType VarType, std::string DataValue[], UI_32 size)
void GetReqArrayStringData (UI_32 VarName, std::string DataValue[], UI_32 &size)
void GetRespArrayStringData (UI_32 VarName, std::string DataValue[], UI_32 &size)
void SetReqStringData (UI_32 VarName, EDataType VarType, std::string DataValue)
void SetRespStringData (UI_32 VarName, EDataType VarType, std::string DataValue)
std::string GetReqStringData (UI_32 VarName)
UI_32 GetReqDataSize (UI_32 Key)
UI_32 GetRespDataSize (UI_32 Key)
EDataType GetReqDataType (UI_32 Key)
EDataType GetRespDataType (UI_32 Key)

Static Public Member Functions

static C_CWORD77_DataPool * GetInstance ()
static void DeleteInstance ()

Detailed Description

class: C_CWORD77_DataPool Description: This is interface class to *CWORD77* Data base. It provides APIs to set and get data from *CWORD77* Data base

---

Constructor & Destructor Documentation

C_CWORD77_DataPool::C_CWORD77_DataPool ()

C_CWORD77_DataPool Constructor for C_CWORD77_DataPool

Parameters:

None	

Returns:

None

C_CWORD77_DataPool::~C_CWORD77_DataPool ()

~C_CWORD77_DataPool Destructor for C_CWORD77_DataPool

Parameters:

None	

Returns:

None

Member Function Documentation

static void C_CWORD77_DataPool::DeleteInstance ()[static]

DeleteInstance API to delete single instance of C_CWORD77_DataPool

Parameters:

None	

Returns:

None

static C_CWORD77_DataPool* C_CWORD77_DataPool::GetInstance ()[static]

GetInstance API to get single instance of C_CWORD77_DataPool

Parameters:

None	

Returns:

Pointer to single instance of C_CWORD77_Controller

template<class T > void C_CWORD77_DataPool::GetReqArrayData (UI_32 *VarName*, T * *Array*, UI_32 & *ArraySize*)[inline]

GetRequestArrayData Templatized Function to get array data from Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key

out	*Array*	T * - Pointer to an array of given type
out	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

void C_CWORD77_DataPool::GetReqArrayStringData (UI_32 *VarName*, std::string *DataValue*[], UI_32 & *size*)[inline]

GetRequestArrayStringData API to get array of string data from Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
out	*DataValue*	string [] - array of strings
out	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

template<class T > T C_CWORD77_DataPool::GetReqData (UI_32 *VarName*)[inline]

GetRequestData Templatized Function to get data from Resquest *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

T - value belong to type T

UI_32 C_CWORD77_DataPool::GetReqDataSize (UI_32 *Key*)[inline]

GetNoofElementsFromRequestTable API to get No of elements in a key from request data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

UI_32 - No of elements

EDataType C_CWORD77_DataPool::GetReqDataType (UI_32 *Key*)[inline]

GetDataTypeFromRequestTable API to get data type of key from request data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

EDataType - Data type of key

std::string C_CWORD77_DataPool::GetReqStringData (UI_32 *VarName*)[inline]

GetRequestStringData API to get string data from Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

string - value of data

template<class T > void C_CWORD77_DataPool::GetRespArrayData (UI_32 *VarName*, T * *Array*, UI_32 & *ArraySize*)[inline]

GetRequestArrayData Templatized Function to get array data from Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
out	*Array*	T * - Pointer to an array of given type
out	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

void C_CWORD77_DataPool::GetRespArrayStringData (UI_32 *VarName*, std::string *DataValue*[], UI_32 & *size*)[inline]

GetResponseArrayStringData API to get array of string data from Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
out	*DataValue*	string [] - array of strings
out	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

template<class T > T C_CWORD77_DataPool::GetRespData (UI_32 *VarName*)[inline]

GetResponseData Templatized Function to get data from Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

T - value belong to type T

UI_32 C_CWORD77_DataPool::GetRespDataSize (UI_32 *Key*)[inline]

GetNoofElementsFromResponseTable API to get No of elements in a key from response data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

UI_32 - No of elements

EDataType C_CWORD77_DataPool::GetRespDataType (UI_32 *Key*)[inline]

GetDataTypeFromResponseTable API to get data type of key from response data base

Parameters:

in	*VarName*	UI_32 - key

Returns:

EDataType - Data type of key

template<class T > void C_CWORD77_DataPool::SetReqArrayData (UI_32 *VarName*, EDataType *DataType*, T * *Array*, UI_32 *ArraySize*)[inline]

SetRequestArrayData Templatized Function to set array data into Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*Array*	T * - Pointer to an array of given type
in	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

void C_CWORD77_DataPool::SetReqArrayStringData (UI_32 *VarName*, EDataType *VarType*, std::string *DataValue*[], UI_32 *size*)[inline]

SetRequestArrayStringData API to set array of string data into Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*DataValue*	string [] - array of strings
in	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

template<class T > void C_CWORD77_DataPool::SetReqData (UI_32 *VarName*, EDataType *DataType*, T *Array*)[inline]

SetRequestData Templatized Function to set data into Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*Array*	T - Data type of key

Returns:

None

void C_CWORD77_DataPool::SetReqStringData (UI_32 *VarName*, EDataType *VarType*, std::string *DataValue*)[inline]

SetRequestStringData API to set string data into Request *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*DataValue*	string [] - array of strings

Returns:

None

template<class T > void C_CWORD77_DataPool::SetRespArrayData (UI_32 *VarName*, EDataType *DataType*, T * *Array*, UI_32 *ArraySize*)[inline]

SetRequestArrayData Templatized Function to set array data into Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*Array*	T * - Pointer to an array of given type
in	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

void C_CWORD77_DataPool::SetRespArrayStringData (UI_32 *VarName*, EDataType *VarType*, std::string *DataValue*[], UI_32 *size*)[inline]

SetRequestArrayStringData API to set array of string data into Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*DataValue*	string [] - array of strings

in	*ArraySize*	UI_32 - No of elements of Array

Returns:

None

template<class T > void C_CWORD77_DataPool::SetRespData (UI_32 *VarName*, EDataType *DataType*, T *Array*)[inline]

SetRequestData Templatized Function to set data into Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*Array*	T - Data type of key

Returns:

None

void C_CWORD77_DataPool::SetRespStringData (UI_32 *VarName*, EDataType *VarType*, std::string *DataValue*)[inline]

SetResponseStringData API to set string data into Response *CWORD77* Data base

Parameters:

in	*VarName*	UI_32 - key
in	*DataType*	EDataType - Data Type of key
in	*DataValue*	string [] - array of strings

Returns:

None

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

66 ns_CWORD77_data_pool_table.h

CCallback< C, M > Class Template Reference

Static Public Member Functions

static CallbackFunctionPtr **set** (void *pInst)
static E_CWORD33_Status **call** (HANDLE y)

The documentation for this class was generated from the following file:
67 ns__CWORD77__common.h

CEventData Class Reference

Public Member Functions

CEventData (UI_32 f_uiEventId)

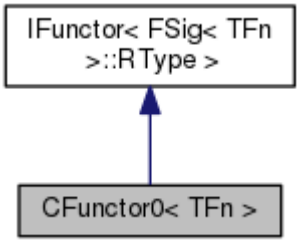
Public Attributes

UI_32 **m_uiEventId**

The documentation for this class was generated from the following file:
68 _CWORD78__sm_eventdata.h

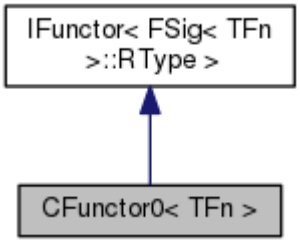
CFunctor0< TFn > Class Template Reference

Inheritance diagram for CFunctor0< TFn >:



IMAGE

Collaboration diagram for CFunctor0< TFn >:



IMAGE

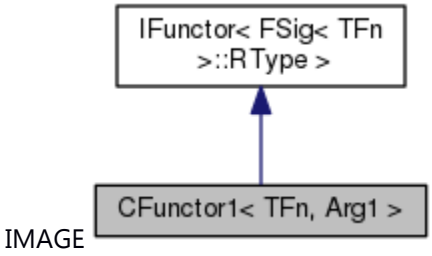
Public Member Functions

CFunctor0 (TFn fn)
FSig< TFn >::RType **operator()** () const
UI_32 **size** () const

The documentation for this class was generated from the following file:
69 ns_utility.hpp

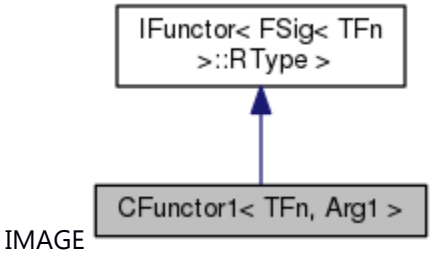
CFunctor1< TFn, Arg1 > Class Template Reference

Inheritance diagram for CFunctor1< TFn, Arg1 >:



IMAGE

Collaboration diagram for CFunctor1< TFn, Arg1 >:



IMAGE

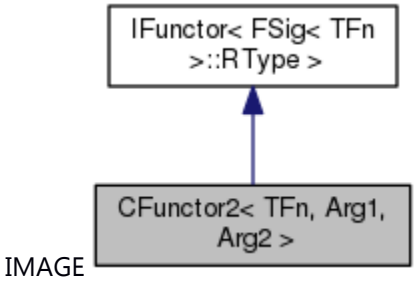
Public Member Functions

CFunctor1 (TFn fn, Arg1 arg1)
FSig< TFn >::RType **operator()** () const
UI_32 **size** () const

The documentation for this class was generated from the following file:
70 ns_utility.hpp

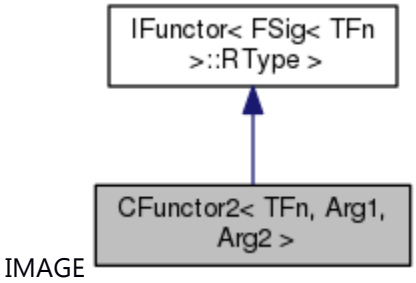
CFunctor2< TFn, Arg1, Arg2 > Class Template Reference

Inheritance diagram for CFunctor2< TFn, Arg1, Arg2 >:



IMAGE

Collaboration diagram for CFunctor2< TFn, Arg1, Arg2 >:



IMAGE

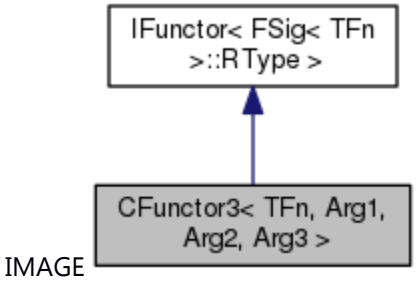
Public Member Functions

CFunctor2 (TFn fn, Arg1 arg1, Arg2 arg2)
FSig< TFn >::RType **operator()** () const
UI_32 **size** () const

The documentation for this class was generated from the following file:
71 ns_utility.hpp

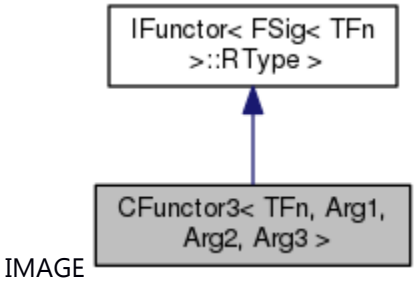
CFunctor3< TFn, Arg1, Arg2, Arg3 > Class Template Reference

Inheritance diagram for CFunctor3< TFn, Arg1, Arg2, Arg3 >:



IMAGE

Collaboration diagram for CFunctor3< TFn, Arg1, Arg2, Arg3 >:



IMAGE

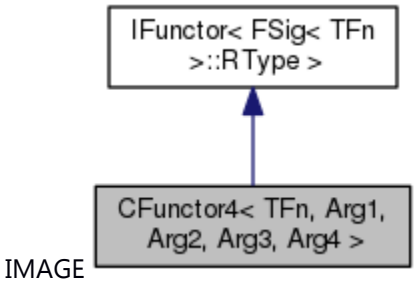
Public Member Functions

CFunctor3 (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3)
FSig< TFn >::RType **operator()** () const
UI_32 **size** () const

The documentation for this class was generated from the following file:
72 ns_utility.hpp

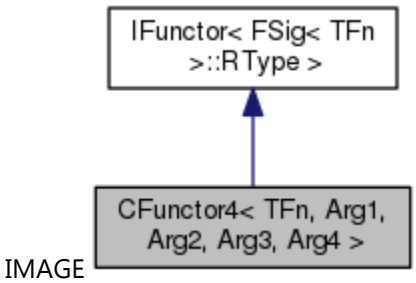
CFunctor4< TFn, Arg1, Arg2, Arg3, Arg4 > Class Template Reference

Inheritance diagram for CFunctor4< TFn, Arg1, Arg2, Arg3, Arg4 >:



IMAGE

Collaboration diagram for CFunctor4< TFn, Arg1, Arg2, Arg3, Arg4 >:



IMAGE

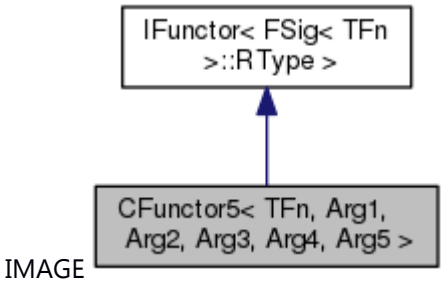
Public Member Functions

CFunctor4 (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4)
FSig< TFn >::RType **operator()** () const
UI_32 **size** () const

The documentation for this class was generated from the following file:
73 ns_utility.hpp

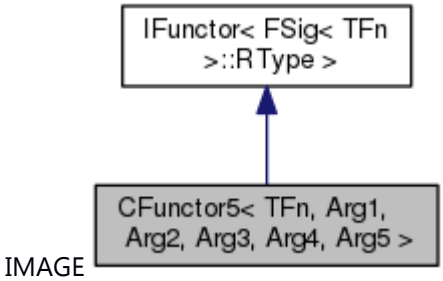
CFunctor5< TFn, Arg1, Arg2, Arg3, Arg4, Arg5 > Class Template Reference

Inheritance diagram for CFunctor5< TFn, Arg1, Arg2, Arg3, Arg4, Arg5 >:



IMAGE

Collaboration diagram for CFunctor5< TFn, Arg1, Arg2, Arg3, Arg4, Arg5 >:



IMAGE

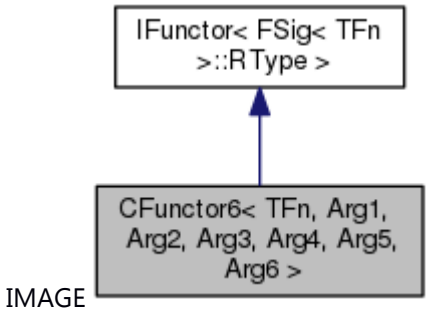
Public Member Functions

CFunctor5 (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4, Arg5 arg5)
FSig< TFn >::RType **operator()** () const
UI_32 **size** () const

The documentation for this class was generated from the following file:
74 ns_utility.hpp

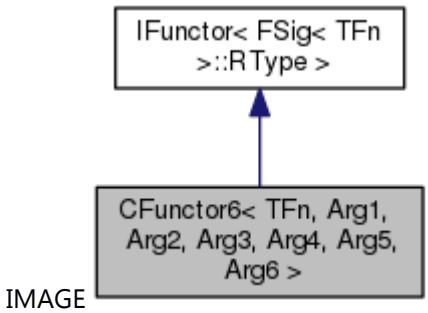
CFunctor6< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6 > Class Template Reference

Inheritance diagram for CFunctor6< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6 >:



IMAGE

Collaboration diagram for CFunctor6< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6 >:



IMAGE

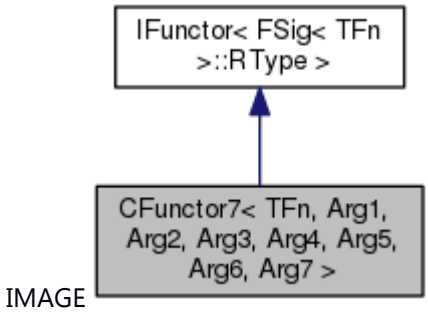
Public Member Functions

CFunctor6 (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4, Arg5 arg5, Arg6 arg6)
FSig< TFn >::RType **operator()** () const
UI_32 **size** () const

The documentation for this class was generated from the following file:
75 ns_utility.hpp

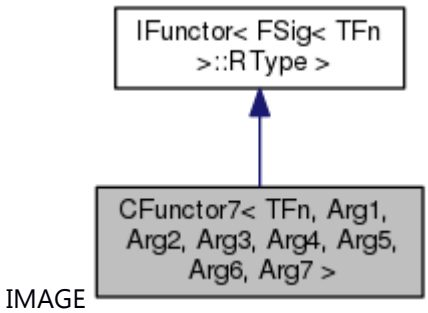
CFunctor7< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7 > Class Template Reference

Inheritance diagram for CFunctor7< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7 >:



IMAGE

Collaboration diagram for CFunctor7< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7 >:



IMAGE

Public Member Functions

CFunctor7 (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4, Arg5 arg5, Arg6 arg6, Arg7 arg7)
FSig< TFn >::RType **operator()** () const
UI_32 **size** () const

The documentation for this class was generated from the following file:
76 ns_utility.hpp

CheckForErrorPolicy< RsrcTraits > Class Template Reference

Public Types

typedef RsrcTraits::Type **Type**

Static Public Member Functions

static Type **check** (Type t)
static bool **isValid** (Type t)

The documentation for this class was generated from the following file:
77 ns_utility.hpp

CMutex Class Reference

Mutex Lock Class.

#include <ns_utility_sys.hpp>

Public Member Functions

CMutex ()
~CMutex ()
void ReadLock ()
void WriteLock ()
void Unlock ()

Detailed Description

Mutex Lock Class.

Brief Introduction

This class defines locking policy for Mutexes.

Constructor & Destructor Documentation

CMutex::CMutex ()

Brief

Construct a CMutex object.

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CMutex::~CMutex ()

Brief

Destruct the CMutex object.

Prerequisite

Construct a CMutex object

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CMutex()

Member Function Documentation

void CMutex::ReadLock ()

Brief

Lock the CMutex object when reading.

Return values:

void	

Prerequisite

Construct a CMutex object

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CMutex(), WriteLock(), UnLock()

void CMutex::Unlock ()

Brief

Open the lock of the CMutex object.

Return values:

void	

Prerequisite

Construct a CMutex object

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CMutex(), ReadLock(), WriteLock()

void CMutex::WriteLock ()

Brief

Lock the CMutex object when writing.

Return values:

void	

Prerequisite

Construct a CMutex object

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CMutex(), ReadLock(), UnLock()

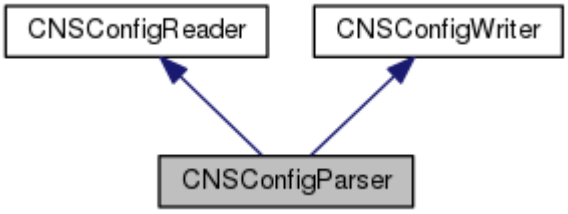
The documentation for this class was generated from the following file:

78 ns_utility_sys.hpp

CNSConfigParser Class Reference

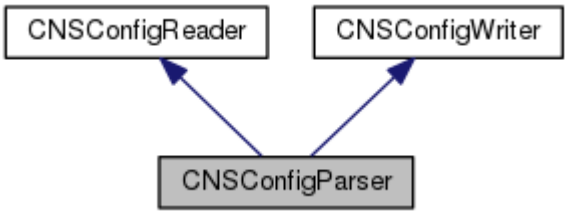
File both Reade and Write Service Class.

#include <ns_config_parser_if.h>
Inheritance diagram for CNSConfigParser:



IMAGE

Collaboration diagram for CNSConfigParser:



IMAGE

Public Member Functions

CNSConfigParser ()
CNSConfigParser (const std::string &f_c_filepath)
~CNSConfigParser ()
E_CWORD33_Status Parse (const std::string &f_c_filepath)

Additional Inherited Members

Detailed Description

File both Reade and Write Service Class.

Brief Introduction

This class provides API's to application for both reading and writing into configuration files.

Constructor & Destructor Documentation

CNSConfigParser::CNSConfigParser ()

Brief

Constructor of CNSConfigParser class

Return values:

none	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigParser::~CNSConfigParser

CNSConfigParser::CNSConfigParser (const std::string & *f_c_filepath*)[explicit]

Brief

Constructor of CNSConfigParser class This is depricated. Use parameterless constructor instead to create object and Parse(filepath) to parse the file using parser object.

Parameters:

in	*f_c_filepath*	const std::string& - Full path of the configuration file

Return values:

none	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigParser::~CNSConfigParser

CNSConfigParser::~CNSConfigParser ()

~

Brief

Destructor of CNSConfigParser class

Return values:

none	

Prerequisite

Call the construct of CNSConfigParser

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigParser::CNSConfigParser

Member Function Documentation

E_CWORD33_Status CNSConfigParser::Parse (const std::string & *f_c_filepath*)

Brief

This function is used to create config reader object and config writer object which then parses the configuration file.

Parameters:

in	*f_c_filepath*	const std::string& - Name of the configuration file

Return values:

e_CWORD33_StatusOK	on success
e_CWORD33_StatusNullPointer	- if memory allocation failed or document not proper
e_CWORD33_StatusFail	- invalid file type
e_CWORD33_StatusInvldParam	invalid parameter
e_CWORD33_StatusErrOther	file data error

Prerequisite

Call the construct of CNSConfigParser

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigReader::Parse, CNSConfigWriter::Parse

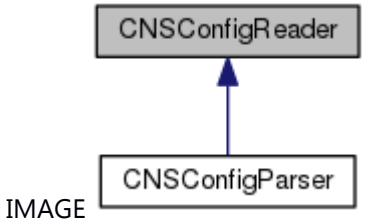
The documentation for this class was generated from the following file:

79 ns_config_parser_if.h

CNSConfigReader Class Reference

Configuration File Read Service Class.

#include <ns_config_parser_if.h>
Inheritance diagram for CNSConfigReader:



IMAGE

Public Member Functions

CNSConfigReader ()
CNSConfigReader (const std::string &f_c_filepath)
E_CWORD33_Status Parse (const std::string &f_c_filepath)
~CNSConfigReader ()
BOOL GetBool (const std::string &f_c_key)
F_64 GetDouble (const std::string &f_c_key)
F_32 GetFloat (const std::string &f_c_key)
SI_32 GetInt (const std::string &f_c_key)
std::string GetString (const std::string &f_c_key)
E_CWORD33_Status GetBool (const std::string &f_c_key, BOOL &f_b_value)
E_CWORD33_Status GetDouble (const std::string &f_c_key, F_64 &f_f_value)
E_CWORD33_Status GetFloat (const std::string &f_c_key, F_32 &f_f_value)
E_CWORD33_Status GetInt (const std::string &f_c_key, SI_32 &f_si_value)
E_CWORD33_Status GetString (const std::string &f_c_key, std::string &f_c_value)

Protected Member Functions

PVOID GetDataPtr ()

Detailed Description

Configuration File Read Service Class.

Brief Introduction

This class provides API's to application to read from configuration files.

Constructor & Destructor Documentation

CNSConfigReader::CNSConfigReader ()

Brief

Constructor of CNSConfigReader class This constructor can only be called by derived class. Application cannot use this constructor to create object of CNSConfigReader class.

Return values:

none	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigReader::~CNSConfigReader

CNSConfigReader::CNSConfigReader (const std::string & *f_c_filepath*)[explicit]

Brief

Constructor of CNSConfigReader class This is depricated. Use parameterless constructor instead to create object and Parse(filepath) to parse the file using reader object.

Parameters:

in	*f_c_filepath*	const std::string& - Name of the configuration file

Return values:

none	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigReader::~CNSConfigReader

CNSConfigReader::~CNSConfigReader ()

~

Brief

Destructor of CNSConfigReader class. It is used when release the instance.

Return values:

none	

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigReader::CNSConfigReader

Member Function Documentation

BOOL CNSConfigReader::GetBool (const std::string & *f_c_key*)

Brief

This function is used to get the bool value associated with key. If there are multiple values associated with the same key, then the first value is returned. This API has been depricated, use GetBool(key, ref_value)

Parameters:

in	*f_c_key*	const std::string& - key to search

Return values:

TRUE	- value associated with key
FALSE	- Failed to get value associated with key

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::SetBool

E_CWORD33_Status CNSConfigReader::GetBool (const std::string & *f_c_key*, BOOL & *f_b_value*)

Brief

This function is used to get the bool value associated with key If there are multiple values associated with the same key, then the first value is returned.

Parameters:

in	*f_c_key*	const std::string& - key string
in	*f_b_value*	BOOL& - value associated with key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key
e_CWORD33_StatusErrOther	- value is not bool type

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::SetBool

PVOID CNSConfigReader::GetDataPtr ()[protected]

GetDataPtr This function is used to get the data pointer. This pointer is then set in config writer. This is needed to avoid recreation of same data structure object in both reader and writer when we create object of NSConfigParser.

Returns:

PVOID - pointer of data structure

F_64 CNSConfigReader::GetDouble (const std::string & *f_c_key*)

Brief

This function is used to get the double value associated with key. If there are multiple values associated with the same key, then the first value is returned. This API has been depricated, use GetDouble(key, ref_value)

Parameters:

in	*f_c_key*	const std::string& - key to search
out	-	none

Return values:

F_64	- value associated with key
0	- Failed to get value associated with key(64bit)

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Please pay attention to the following points when you use this API.

We can take the value in specification only if it is in the range of [1.1754943508222875079e-38F, 3.4028234663852885981e+38F](min value of double, max value of double).

Type

Sync Only

See also:

CNSConfigWriter::Set

E_CWORD33_Status CNSConfigReader::GetDouble (const std::string & *f_c_key*, F_64 & *f_f_value*)

Brief

This function is used to get the double value associated with key. If there are multiple values associated with the same key, then the first value is returned.

Parameters:

in	*f_c_key*	const std::string& - key to search
in	*f_f_value*	F_64& - value associated with key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key
e_CWORD33_StatusInvldBuf	- change error

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Please pay attention to the following points when you use this API.

Return e_CWORD33_StatusOK if we didn't excute the changing operation
We can take the value in specification only if it is in the range of [1.17549435082228750797e-38F, 3.40282346638528859812e+38F](min value of double, max value of double).

Type

Sync Only

See also:

CNSConfigWriter::Set

F_32 CNSConfigReader::GetFloat (const std::string & *f_c_key*)

Brief

This function is used to get the float value associated with key If there are multiple values associated with the same key, then the first value is returned. This API has been depricated, use GetFloat(key, ref_value)

Parameters:

in	*f_c_key*	const std::string& - key to search

Return values:

F_32	value associated with key
0	- Failed to get value associated with key(32bit)

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::Set

E_CWORD33_Status CNSConfigReader::GetFloat (const std::string & *f_c_key*, F_32 & *f_f_value*)

Brief

This function is used to get the float value associated with key If there are multiple values associated with the same key, then the first value is returned.

Parameters:

in	*f_c_key*	const std::string& - key to search
in	*f_f_value*	F_32& - value associated with key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key
e_CWORD33_StatusInvldBuf	- change error

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::Set

SI_32 CNSConfigReader::GetInt (const std::string & *f_c_key*)

Brief

This function is used to get the integer value associated with key If there are multiple values associated with the same key, then the first value is returned. This API has been depricated, use GetInt(key, ref_value)

Parameters:

in	*f_c_key*	const std::string& - key to search

Return values:

SI_32	- value associated with key
0.0f	- Failed to get value associated with key(32bit)

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Type

Sync Only

Please pay attention to the following points when you use this API.

If data string getted from file is larger than INT_MAX(2147483648) return -1.

See also:

CNSConfigWriter::Set

E_CWORD33_Status CNSConfigReader::GetInt (const std::string & *f_c_key*, SI_32 & *f_si_value*)

Brief

This function is used to get the integer value associated with key If there are multiple values associated with the same key, then the first value is returned.

Parameters:

in	*f_c_key*	const std::string& - key to search

out	*f_si_value*	SI_32& - value associated with key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key
e_CWORD33_StatusInvldBuf	- change error

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::Set

std::string CNSConfigReader::GetString (const std::string & *f_c_key*)

Brief

This function is used to get the string value associated with key If there are multiple values associated with the same key, all the values separated by delimiter "\n" are returned. This API has been depricated, use GetString(key, ref_value)

Parameters:

in	*f_c_key*	const std::string& - key to search

Return values:

std::string	value(string) associated with key
nullbuffer	- Failed to get value associated with key

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::Set

E_CWORD33_Status CNSConfigReader::GetString (const std::string & *f_c_key*, std::string & *f_c_value*)

Brief

This function is used to get the string value associated with key If there are multiple values associated with the same key, all the values separated by delimiter "\n" are returned.

Parameters:

in	*f_c_key*	const std::string& - key to search
in	*f_c_value*	std::string& - value(string) associated with key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key

Prerequisite

Call the construct of CNSConfigReader

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::Set

E_CWORD33_Status CNSConfigReader::Parse (const std::string & *f_c_filepath*)

Brief

This function is used to parse configuration file

Parameters:

in	*f_c_filepath*	const std::string& - Name of the configuration file

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- if memory allocation failed or document is not proper
e_CWORD33_StatusFail	- invalid file type

e_CWORD33_StatusInvldParam	invalid parameter
e_CWORD33_StatusErrOther	- file data error

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::Parse, CNSConfigParser::Parse

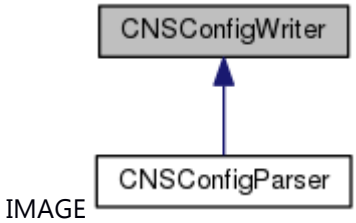
The documentation for this class was generated from the following file:

80 ns_config_parser_if.h

CNSConfigWriter Class Reference

File Write Service Class.

#include <ns_config_parser_if.h>
Inheritance diagram for CNSConfigWriter:



IMAGE

Public Member Functions

CNSConfigWriter ()
CNSConfigWriter (const std::string &f_c_filepath)
~CNSConfigWriter ()
E_CWORD33_Status Parse (const std::string &f_c_filepath)
E_CWORD33_Status Save ()
E_CWORD33_Status SetBool (const std::string &f_c_key, BOOL f_b_value)
E_CWORD33_Status Set (const std::string &f_c_key, F_64 f_f_value)
E_CWORD33_Status Set (const std::string &f_c_key, F_32 f_f_value)
E_CWORD33_Status Set (const std::string &f_c_key, SI_32 f_si_value)
E_CWORD33_Status Set (const std::string &f_c_key, const std::string &f_s_value)

Protected Member Functions

CNSConfigWriter (const std::string &f_c_filepath, BOOL f_b_noload)
VOID SetDataPtr (PVOID f_p_data)
E_CWORD33_Status Create (const std::string &f_c_filepath)

Detailed Description

File Write Service Class.

Brief Introduction

This class provides API's to application to write to configuration files.

Constructor & Destructor Documentation

CNSConfigWriter::CNSConfigWriter ()

Brief

Constructor of CNSConfigWriter class

Return values:

none	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::~CNSConfigWriter

CNSConfigWriter::CNSConfigWriter (const std::string & *f_c_filepath*)[explicit]

Brief

Constructor of CNSConfigWriter class This is depricated. Use parameterless constructor instead to create object and Parse(filepath) to parse the file using writer object.

Parameters:

in	*f_c_filepath*	const std::string& - Full path of the configuration file

Return values:

none	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::~CNSConfigWriter

CNSConfigWriter::~CNSConfigWriter ()

~

Brief

Destructor of CNSConfigWriter class

Return values:

none	

Prerequisite

Call the construct of CNSConfigWriter

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::CNSConfigWriter

CNSConfigWriter::CNSConfigWriter (const std::string & *f_c_filepath*, BOOL *f_b_noload*)[protected]

CNSConfigWriter Constructor of CNSConfigWriter class

Parameters:

in	*f_c_filepath*	const std::string& - Full path of the configuration file
in	*f_b_noload*	BOOL - load writer with no data

Returns:

Member Function Documentation

E_CWORD33_Status CNSConfigWriter::Create (const std::string & *f_c_filepath*)[protected]

Create This function is used to create config object

Parameters:

in	*f_c_filepath*	const std::string& - Name of the configuration file

Returns:

E_CWORD33_Status - success or failure

E_CWORD33_Status CNSConfigWriter::Parse (const std::string & *f_c_filepath*)

Brief

This function is used to parse the configuration file

Parameters:

in	*f_c_filepath*	const std::string& - Name of the configuration file

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- if memory allocation failed or document not proper
e_CWORD33_StatusFail	- invalid file type
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusErrOther	- file data error

Prerequisite

Call the construct of CNSConfigWriter

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigParser::Parse, CNSConfigWriter::Parse

E_CWORD33_Status CNSConfigWriter::Save ()

Brief

This function is used to save the configuration settings to config file

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusFail	- file access fail

Prerequisite

Call the construct of CNSConfigWriter

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigWriter::SetBool, CNSConfigWriter::Set

E_CWORD33_Status CNSConfigWriter::Set (const std::string & *f_c_key*, F_64 *f_f_value*)

Brief

This function is used to set the double value corresponding to key. If multiple identical keys are found it sets the value for all the keys.

Parameters:

in	*f_c_key*	const std::string& - key
in	*f_f_value*	F_64 - double value to set for key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key

Prerequisite

Call the construct of CNSConfigWriter

Change of internal state

None

Classification

Public

Please pay attention to the following points when you use this API.

We can take the value in specification only if it is in the range of [2.2250738585072013830 9e-308L, 1.7976931348623157e+308L](min value of double, max value of double).
Valid condition of search key string: there is no separator and line feed in config tag

Type

Sync Only

See also:

CNSConfigReader::GetDouble

E_CWORD33_Status CNSConfigWriter::Set (const std::string & *f_c_key*, F_32 *f_f_value*)

Brief

This function is used to set the float value corresponding to key. If multiple identical keys are found it sets the value for all the keys.

Parameters:

in	*f_c_key*	const std::string& - key
in	*f_f_value*	F_32 - float value to set for key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key

Prerequisite

Call the construct of CNSConfigWriter

Change of internal state

None

Classification

Public

Please pay attention to the following points when you use this API.

We can take the value in specification only if it is in the range of [1.17549435082228750797e-38F, 3.40282346638528859812e+38F](min value of double, max value of double).
Valid condition of search key string: there is no separator and line feed in config tag

Type

Sync Only

See also:

CNSConfigReader::GetFloat

E_CWORD33_Status CNSConfigWriter::Set (const std::string & *f_c_key*, SI_32 *f_si_value*)

Brief

This function is used to set the integer value corresponding to key. If multiple identical keys are found it sets the value for all the keys.

Parameters:

in	*f_c_key*	const std::string& - key

in	*f_si_value*	SI_32 - integer value to set for key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key

Prerequisite

Call the construct of CNSConfigWriter

Change of internal state

None

Classification

Public

Please pay attention to the following points when you use this API.

Valid condition of search key string: there is no separator and line feed in config tag

Type

Sync Only

See also:

CNSConfigReader::GetInt

E_CWORD33_Status CNSConfigWriter::Set (const std::string & *f_c_key*, const std::string & *f_s_value*)

Brief

This function is used to set the string value corresponding to key. If multiple identical keys are found it sets the value for all the keys.

Parameters:

in	*f_c_key*	const std::string& - key
in	*f_s_value*	const std::string& - string value to set for key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key

Prerequisite

Call the construct of CNSConfigWriter

Change of internal state

None

Classification

Public

Please pay attention to the following points when you use this API.

Valid condition of search key string: there is no separator and line feed in config tag

Type

Sync Only

See also:

CNSConfigReader::GetString

E_CWORD33_Status CNSConfigWriter::SetBool (const std::string & *f_c_key*, BOOL *f_b_value*)

Brief

This function is used to set the bool value corresponding to key This function is not overloaded with other Set() API's of this class because BOOL is typedef of UI_32 so when we pass integer or BOOL to Set API it would have called the same API. If multiple identical keys are found it sets the value for all the keys.

Parameters:

in	*f_c_key*	const std::string& - key
in	*f_b_value*	BOOL - bool value to set for key

Return values:

e_CWORD33_StatusOK	- success
e_CWORD33_StatusNullPointer	- memory allocation failed
e_CWORD33_StatusInvldParam	- invalid parameter
e_CWORD33_StatusFail	- can not find key
e_CWORD78_StatusErrOther	- if passsed value is invalid

Prerequisite

Call the construct of CNSConfigWriter

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CNSConfigReader::GetBool

VOID CNSConfigWriter::SetDataPtr (PVOID *f_p_data*)[protected]

SetDataPtr This function is used to set the data pointer in writer class with data pointer created in config reader class. This is needed to avoid recreation of same data structure object in both reader and writer when we create object of NSConfigParser

Parameters:

in	*f_p_data*	PVOID - Pointer to data structure

Returns:

VOID

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

81 ns_config_parser_if.h

CNSRcsPlugin Class Reference

CNSRcsPlugin.

#include <ns_rcs_plugin.hpp>

Public Member Functions

CNSRcsPlugin (std::string f_cPluginName, TFPNSRcsSendPassthruData f_fpNSRcsSendPassthruData)
virtual ~CNSRcsPlugin ()
UI_8 GetPluginId ()
std::string GetPluginName ()
virtual E_CWORD78_Status ProcessRequestMessage (UI_32 f_uiProtocol, UI_8 f_ui8ClientId, CPassThruInDataHandler *f_pData)=0
virtual E_CWORD33_Status ProcessResponseMessage (PCSTR f_cMsgSource, UI_32 f_uiProtocol, UI_8 *f_pui8Data, UI_32 f_uiDataLength)=0
virtual UI_32 const * GetResponseProtocolArray ()=0
virtual UI_32 const GetResponseProtocolArrayLength ()=0
E_CWORD78_Status SendPassthruOutData (UI_8 f_ui8clientId, PVOID f_data, UI_16 f_ui16PayloadLength)

Static Public Member Functions

static VOID SetParentName (const std::string &f_cParentName)

Static Public Attributes

static std::string **m_cParentName**

Detailed Description

CNSRcsPlugin.

Brief Introduction

Class to provide base class of CNSRcsPlugin

Constructor & Destructor Documentation

CNSRcsPlugin::CNSRcsPlugin (std::string *f_cPluginName*, TFPNSRcsSendPassthruData *f_fpNSRcsSendPassthruData*)[inline]

Summary

Constructor of CNSRcsPlugin class.

Parameters:

in	*f_cPluginName*	std::string - the name of plugin

in	*f_fpNSRcsSendPassthruData*	TFPNSRcsSendPassthruData - the pointer of send data function

Return values:

None	

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

~CNSRcsPlugin

virtual CNSRcsPlugin::~CNSRcsPlugin ()[inline], [virtual]

Summary

Destructor of CNSRcsPlugin 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

sync only

See also:

CNSRcsPlugin

Member Function Documentation

UI_8 CNSRcsPlugin::GetPluginId ()[inline]

Summary

get plugin id.

Parameters:

None	

Return values:

Plugin	Id UI_8 - the ID of plugin

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

CNSRcsPlugin

std::string CNSRcsPlugin::GetPluginName ()[inline]

Summary

get plugin id.

Parameters:

None	

Return values:

m_cPluginName	std::string - the name of plugin

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

CNSRcsPlugin

virtual UI_32 const* CNSRcsPlugin::GetResponseProtocolArray ()[pure virtual]

Summary

get response protocol array

Parameters:

none	

Return values:

UI_32	const *

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

CNSRcsPlugin

virtual UI_32 const CNSRcsPlugin::GetResponseProtocolArrayLength ()[pure virtual]

Summary

get length of response protocol array

Parameters:

none	

Return values:

UI_32	const

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

CNSRcsPlugin

virtual E_CWORD78_Status CNSRcsPlugin::ProcessRequestMessage (UI_32 *f_uiProtocol*, UI_8 *f_ui8ClientId*, CPassThruInDataHandler * *f_pData*)[pure virtual]

Summary

request from host side.

Parameters:

f_uiProtocol	UI_32 - protocol ID
f_ui8ClientId	UI_8 - client ID
f_pData	CPassThruInDataHandler - the handler of transfer data

Return values:

none	

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

CNSRcsPlugin

virtual E_CWORD33_Status CNSRcsPlugin::ProcessResponseMessage (PCSTR *f_cMsgSource*, UI_32 *f_uiProtocol*, UI_8 * *f_pui8Data*, UI_32 *f_uiDataLength*)[pure virtual]

Summary

response from service side on target

Parameters:

f_cMsgSource	PCSTR -
f_uiProtocol	UI_32 - protocol ID
f_pui8Data	UI_8 - pointer of data

f_uiDataLength	UI_32 - length of data

Return values:

E_CWORD33_Status	

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

CNSRcsPlugin

E_CWORD78_Status CNSRcsPlugin::SendPassthruOutData (UI_8 *f_ui8clientId*, PVOID *f_data*, UI_16 *f_ui16PayloadLength*)[inline]

Summary

the function of transport data

Parameters:

f_ui8ClientId	UI_8 - client id
f_data	PVOID - the pointer of transport data
f_ui16PayloadLength	UI_16 - the size of transport data

Return values:

E_CWORD33_Status	

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

CNSRcsPlugin

static VOID CNSRcsPlugin::SetParentName (const std::string & *f_cParentName*)[inline], [static]

Summary

set parent name.

Parameters:

f_cParentName	std::string - the name of parent

Return values:

none	

Preconditions

None.

Change of the internal state

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

Classification

Public

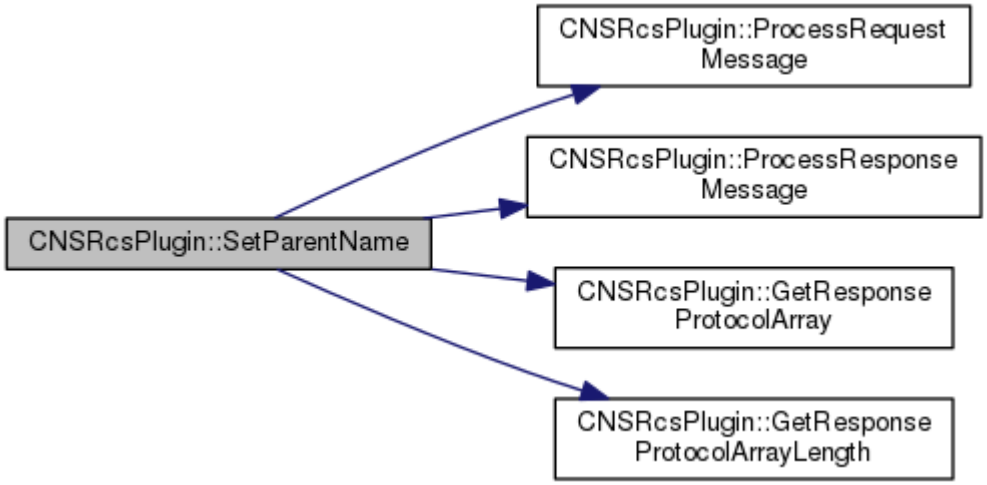
Type

sync only

See also:

CNSRcsPlugin

Here is the call graph for this function:



IMAGE

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

82 ns_rcs_plugin.hpp

CNSRingBuffer Class Reference

CNSRingBuffer.

#include <ns_ringbuffer.h>

Public Member Functions

CNSRingBuffer (const std::string &f_cMappedFilePath, const UI_32 f_uiSize, int f_lid=-1)
~CNSRingBuffer ()
E_CWORD33_Status Open ()
BOOL IsOpen ()
E_CWORD33_Status Close ()
SI_32 Read (PSTR buffer, const UI_32 f_uilength)
SI_32 Write (PCSTR buffer, const UI_32 f_uilength)
E_CWORD33_Status DumpToFile (PCSTR f_pPath, PUI_32 f_uiDumpSize)
SI_32 GetSize ()
E_CWORD33_Status ClearBuf ()
E_CWORD33_Status SetReadPtrToWritePtr ()

Detailed Description

CNSRingBuffer.

Brief Introduction

Class to provide class of ring buffer.

Constructor & Destructor Documentation

CNSRingBuffer::CNSRingBuffer (const std::string & *f_cMappedFilePath*, const UI_32 *f_uiSize*, int *f_lid* = -1)

Brief

Constructor for CNSRingBuffer

Parameters:

in	*f_cMappedFilePath*	const std::string& - path of the ring buffer mapped file
in	*f_uiSize*	const UI_32 - size of ring buffer
in	*f_lid*	int - Lock ID

Return values:

none	

Preconditons

Change of internal status

Classification

Public

Type

sync only

See also:

~CNSRingBuffer

CNSRingBuffer::~CNSRingBuffer ()

Brief

Destructor for CNSRingBuffer.

Parameters:

none	

Return values:

none	

Preconditons

Change of internal status

Classification

Public

Type

sync only

See also:

CNSRingBuffer

Member Function Documentation

E_CWORD33_Status CNSRingBuffer::ClearBuf ()

Brief

This function clears the ring buffer.

Parameters:

none	

Return values:

e_CWORD33_StatusOK	

e_CWORD33_StatusFail	

Preconditions

Open().

Change of the internal state

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

Classification

Public

Type

sync only

See also:

Open

E_CWORD33_Status CNSRingBuffer::Close ()

Brief

This function unmaps the ring buffer object.

Parameters:

none	

Return values:

e_CWORD33_StatusOK	
e_CWORD33_StatusFail	

Preconditons

Change of internal status

Classification

Public

Type

sync only

See also:

Open

E_CWORD33_Status CNSRingBuffer::DumpToFile (PCSTR *f_pPath*, PUI_32 *f_uiDumpSize*)

Brief

This function writes all the data in the buffer into provided file f_pPath.

Parameters:

in	*f_pPath*	PCSTR - file path.
in	*f_uiDumpSize*	PUI_32 - Returns The number of bytes written into file

Return values:

e_CWORD33_StatusOK	
e_CWORD33_StatusNullPointer	
e_CWORD33_StatusInvldParam	
e_CWORD33_StatusFail	
e_CWORD33_StatusFileLoadError	
e_CWORD33_StatusErrOther	

Preconditions

Open().

Change of the internal state

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

Classification

Public

Type

sync only

See also:

Open

SI_32 CNSRingBuffer::GetSize ()

Brief

This function returns the number of unread bytes which can be read by Read().

Return values:

SI_32	Returns The number of bytes which can be read by Read()
-1	error occurred

Preconditions

Open().

Change of the internal state

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

Classification

Public

Type

sync only

See also:

Open

BOOL CNSRingBuffer::IsOpen ()

Brief

This function is used to check whether the ring buffer is opened or not..

Parameters:

none	

Return values:

TRUE	- Open
FALSE	- Not open

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

CNSRingBuffer, Close

E_CWORD33_Status CNSRingBuffer::Open ()

Brief

This function opens and maps the ring buffer object.

Parameters:

none	

Return values:

e_CWORD33_StatusOK	
e_CWORD33_StatusInvldParam	
e_CWORD33_StatusFail	
e_CWORD33_StatusErrOther	

Preconditons

Change of internal status

Classification

Public

Type

sync only

See also:

CNSRingBuffer, Close

SI_32 CNSRingBuffer::Read (PSTR *buffer*, const UI_32 *f_uilength*)

Brief

This function reads data from the ring buffer.

Parameters:

[OUT]	buffer PSTR - pointer to the buffer in which the read data is stored
[IN]	f_uilength UI_32 - length of the data buffer provided

Return values:

SI_32	Returns The number of bytes actually read
-1	error occurred

Preconditions

Open().

Change of the internal state

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

Classification

Public

Type

sync only

See also:

Open

E_CWORD33_Status CNSRingBuffer::SetReadPtrToWritePtr ()

Brief

This function sets the position of read ptr to write ptr in buffer.

Parameters:

none	

Return values:

e_CWORD33_StatusOK	success

e_CWORD33_StatusFail	ring buffer is not open

Preconditions

Open().

Change of the internal state

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

Classification

Public

Type

sync only

See also:

Open

SI_32 CNSRingBuffer::Write (PCSTR *buffer*, const UI_32 *f_uilength*)

Brief

This function writes the data into the ring buffer.

Parameters:

[OUT]	buffer PSTR - pointer to the buffer containing the data to be written
[IN]	f_uilength UI_32 - length of the data buffer to be written

Return values:

SI_32	Returns The number of bytes written
-1	error occurred

Preconditions

Open().

Change of the internal state

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

Classification

Public

Type

sync only

See also:

Open

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

83 ns_ringbuffer.h

CNSSharedMem Class Reference

this file has the C_CWORD33_Guard class definitions

#include <ns_sharedmem.h>

Public Member Functions

CNSSharedMem (const std::string &f_cSharedMemName, const UI_32 f_uiSize)
~CNSSharedMem ()
E_CWORD33_Status Open ()
BOOL IsOpen ()
E_CWORD33_Status Close ()
SI_32 Read (PSTR buffer, const UI_32 f_uilength, const BOOL f_bBlock=TRUE)
SI_32 Write (PCSTR buffer, const UI_32 f_uilength)
E_CWORD33_Status DumpToFile (PCSTR f_pPath, PUI_32 f_uiDumpSize)
SI_32 GetSize ()
E_CWORD33_Status ClearBuf ()
E_CWORD33_Status SetReadPtrToWritePtr ()

Detailed Description

this file has the C_CWORD33_Guard class definitions

shared memory

Brief Introduction

This file has the C_CWORD33_Guard class definitions. This is an interface that defines the Evaluate interface for guard condition validation.

Brief Introduction

This class is used for handle shared memory.

Constructor & Destructor Documentation

CNSSharedMem::CNSSharedMem (const std::string & *f_cSharedMemName*, const UI_32 *f_uiSize*)

Brief

Constructor for CNSSharedMem

Parameters:

in	*f_cSharedMemName*	const std::string& - name of the shared memory
in	*f_uiSize*	const UI_32 - size of shared memory

Return values:

none	

Preconditons

None

Change of internal status

None

Classification

Public

Type

See also:

~CNSSharedMem

CNSSharedMem::~CNSSharedMem ()

Brief

Destructor for CNSSharedMem.

Return values:

none	

Preconditons

None

Change of internal status

Change the state of shared memory to closed state

Classification

Public

Type

See also:

CNSSharedMem(const std::string&, const UI_32), CNSSharedMem(), Close

---

Member Function Documentation

E_CWORD33_Status CNSSharedMem::ClearBuf ()

Brief

Clear buffer.

Parameters:

None	

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK on success, e_CWORD33_StatusSemLockFail/

	e_CWORD33_StatusSemLockFail - mutex locking/unlocking error e_CWORD33_StatusFail - shared memory is not opened

Preconditons

Change of internal status

set m_pShmHdr->m_uiReadPtr with 0;
set m_pShmHdr->m_uiWritePtr with 0;
set m_pShmHdr->m_uiUnReadSize with 0;
set m_pShmHdr->m_bIsFull with FALSE;

Classification

public

Type

sync only

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

None

E_CWORD33_Status CNSSharedMem::Close ()

Brief

This function unmaps the shared memory object.

Return values:

e_CWORD33_StatusOK	if shared memory closed successfully
e_CWORD33_StatusFail	if shared memory is not opened

Preconditons

Open the shared memory object

Change of internal status

Close the shared memory object

Classification

public

Type

Open Close

See also:

Open

E_CWORD33_Status CNSSharedMem::DumpToFile (PCSTR *f_pPath*, PUI_32 *f_uiDumpSize*)

Brief

Dump data to file.

Parameters:

in	*f_pPath*	PCSTR - file path.
in	*[in]*	f_uiDumpSize PUI_32 - Returns The number of bytes written into file

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK on success, e_CWORD33_StatusFileLoadError - file path is incorrect e_CWORD33_StatusSemLockFail/ e_CWORD33_StatusSemLockFail - mutex locking/unlocking error e_CWORD33_StatusFail - shared memory is not opened e_CWORD33_StatusInvldParam - invalid param

Preconditons

Change of internal status

-None

Classification

public

Type

sync only

e_CWORD33_Status:Result

e_CWORD33_StatusOK:Success

Except e_CWORD33_StatusOK:Failure

See also:

None

SI_32 CNSSharedMem::GetSize ()

Brief

This function gets size of unread bytes.

Parameters:

None	

Return values:

Except	NS_SHM_ERROR - Returns The number of unread bytes
NS_SHM_ERROR	-if an error occurred

Preconditons

Change of internal status

None

Classification

Public

Type

sync only

l_uiReadSize:Result
NS_SHM_ERROR:Failure
Except NS_SHM_ERROR:Success

See also:

None

BOOL CNSSharedMem::IsOpen ()

Brief

This function is used to check whether the shared memory buffer is opened or not.

Return values:

TRUE	- Open
FALSE	- Not open

Preconditons

None

Change of internal status

None

Classification

public

Type

Open Close

See also:

none

E_CWORD33_Status CNSSharedMem::Open ()

Brief

This function opens and maps the shared memory object.

Return values:

e_CWORD33_StatusOK	if shared memory opened successfully
e_CWORD33_StatusFail	unable to open shared memory
e_CWORD33_StatusErrOther	if shared memory already opened

Preconditons

None

Change of internal status

Open shared memory object

Classification

Public

Type

Open Close

See also:

Close

SI_32 CNSSharedMem::Read (PSTR *buffer*, const UI_32 *f_uilength*, const BOOL *f_bBlock* = TRUE)

Brief [out] buffer

PSTR - pointer to the buffer in which the read data is stored

Parameters:

in	*f_uilength*	UI_32 - length of the data buffer provided
in	*f_bBlock*	BOOL - TRUE - blocking call FALSE - non blocking call

Return values:

SI_32	Returns The number of bytes actually read, or NS_SHM_ERROR if an error occurred
NS_SHM_ERROR	

Preconditons

Open the shared memory object

Change of internal status

None

Classification

public

Type

Open Close

See also:

Write

E_CWORD33_Status CNSSharedMem::SetReadPtrToWritePtr ()

Brief

This function sets the position of read ptr to write ptr in buffer.

Parameters:

None	

Return values:

E_CWORD33_Status	e_CWORD33_StatusOK on success, e_CWORD33_StatusSemLockFail/ e_CWORD33_StatusSemLockFail - mutex locking/unlocking error e_CWORD33_StatusFail - shared memory is not opened

Preconditons

Change of internal status

set m_pShmHdr->m_uiReadPtr with m_pShmHdr->m_uiWritePtr;
set m_pShmHdr->m_uiUnReadSize with 0;

Classification

public

Type

sync only

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

None

SI_32 CNSSharedMem::Write (PCSTR *buffer*, const UI_32 *f_uilength*)

Brief

This function writes the data into the shared memory.

Parameters:

in	*buffer*	PCSTR - pointer to the buffer containing the data to be written
in	*f_uilength*	const UI_32 - length of the data buffer to be written

Return values:

SI_32	Returns The number of bytes written, or NS_SHM_ERROR if an error occurred

Preconditons

Open the shared memory object

Change of internal status

None

Classification

public

Type

Open Close

See also:

Read

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

84 ns_sharedmem.h

CNSSharedMemReader Class Reference

read shared memory

#include <ns_sharedmem_reader.h>

Public Member Functions

CNSSharedMemReader (const std::string &f_cSharedMemName, const BOOL f_bBlock=TRUE)
~CNSSharedMemReader ()
E_CWORD33_Status Open ()
BOOL IsOpen ()
E_CWORD33_Status Close ()
SI_32 Read (PSTR buffer, const UI_32 f_uilength)
E_CWORD33_Status DumpToFile (PCSTR f_pPath, PUI_32 f_uiDumpSize)
SI_32 GetSize ()
E_CWORD33_Status SetReadPtrToWritePtr ()

Detailed Description

read shared memory

Brief Introduction

This class is used for read shared memory.

Constructor & Destructor Documentation

CNSSharedMemReader::CNSSharedMemReader (const std::string & *f_cSharedMemName*, const BOOL *f_bBlock* = TRUE)

Brief

Constructor for CNSSharedMemReader

Parameters:

in	*f_cSharedMemName*	const std::string& - name of the shared memory
in	*f_bBlock*	BOOL - TRUE - blocking call for Read() API,Thread will get blocked the thread until data is available for read. FALSE - non blocking call for Read() API

Return values:

None	

Preconditons

-None

Change of internal status

None

Classification

public

Type

None

See also:

~CNSSharedMemReader

CNSSharedMemReader::~CNSSharedMemReader ()

Brief

Destructor for CNSSharedMemReader.

Parameters:

None	

Return values:

none	

Preconditons

None

Change of internal status

Change the state of shared memory to closed state

Classification

Public

Type

-None

See also:

CNSSharedMemReader(const std::string&, const BOOL), CNSSharedMemReader(), Close

---

Member Function Documentation

E_CWORD33_Status CNSSharedMemReader::Close ()

Brief

This function unmaps the shared memory object.

Return values:

e_CWORD33_StatusOK	if shared memory closed successfully
e_CWORD33_StatusFail	if shared memory is not opened

Preconditons

Open the shared memory object

Change of internal status

Close the shared memory object

Classification

public

Type

Open Close

See also:

Open

E_CWORD33_Status CNSSharedMemReader::DumpToFile (PCSTR *f_pPath*, PUI_32 *f_uiDumpSize*)

Brief

This function writes all the data in the buffer into provided file f_pPath.

Parameters:

in	*f_pPath*	PCSTR - file path.
in	*f_uiDumpSize*	PUI_32 - Returns The number of bytes written into file

Return values:

e_CWORD33_StatusOK	- on success
e_CWORD33_StatusNullPointer	
e_CWORD33_StatusInvldParam	- invalid param
e_CWORD33_StatusFail	- shared memory is not open
e_CWORD33_StatusFileLoadError	- file path is incorrect
e_CWORD33_StatusErrOther	
e_CWORD33_StatusSemUnlockFail	

Preconditons

File is opened.

Change of internal status

None

Classification

public

Type

none

See also:

Open

SI_32 CNSSharedMemReader::GetSize ()

Brief

This function gets size of unread .bytes.

Parameters:

None	

Return values:

Except	NS_SHM_ERROR - Returns The number of bytes written,
NS_SHM_ERROR	-if an error occurred

Preconditons

Change of internal status

None

Classification

Public

Type

sync only

l_uiReadSize:Result
NS_SHM_ERROR:Failure
Except NS_SHM_ERROR:Success

See also:

None

BOOL CNSSharedMemReader::IsOpen ()

Brief

This function is used to check whether the shared memory buffer is opened or not.

Return values:

TRUE	- Open
FALSE	- Not open

Preconditons

None

Change of internal status

None

Classification

public

Type

Open Close

See also:

none

E_CWORD33_Status CNSSharedMemReader::Open ()

Brief

This function opens and maps the shared memory object.

Return values:

e_CWORD33_StatusOK	if shared memory opened successfully
e_CWORD33_StatusFail	unable to open shared memory
e_CWORD33_StatusErrOther	if shared memory already opened

Preconditons

None

Change of internal status

Open shared memory object

Classification

Public

Type

Open Close

See also:

Close

SI_32 CNSSharedMemReader::Read (PSTR *buffer*, const UI_32 *f_uilength*)

Brief

This function reads data from the shared memory.

Parameters:

in	*buffer*	PSTR - pointer to the buffer in which the read data is stored

in	*f_uilength*	const UI_32 - length of the data buffer provided

Return values:

The	number of bytes actually read, or NS_SHM_ERROR if an error occurred

Preconditons

Open the shared memory object

Change of internal status

None

Classification

public

Type

Open Close

See also:

Write

E_CWORD33_Status CNSSharedMemReader::SetReadPtrToWritePtr ()

Brief

This function sets the position of read ptr to write ptr in buffer.

Return values:

e_CWORD33_StatusOK	- on success
e_CWORD33_StatusFail	- shared memory is not open
e_CWORD33_StatusSemLockFail	- mutex locking error
e_CWORD33_StatusSemUnlockFail	- mutex unlocking error

Preconditons

File is opened.

Change of internal status

None

Classification

public

Type

Open Close

See also:

Open

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

85 ns_sharedmem_reader.h

CNSSharedMemWriter Class Reference

write shared memory

#include <ns_sharedmem_writer.h>

Public Member Functions

CNSSharedMemWriter (const std::string &f_cSharedMemName, const UI_32 f_uiSize)
~CNSSharedMemWriter ()
E_CWORD33_Status Open ()
BOOL IsOpen ()
E_CWORD33_Status Close ()
SI_32 Write (PCSTR buffer, const UI_32 f_uilength)
E_CWORD33_Status ClearBuf ()

Detailed Description

write shared memory

Brief Introduction

This class is used for write shared memory.

Constructor & Destructor Documentation

CNSSharedMemWriter::CNSSharedMemWriter (const std::string & *f_cSharedMemName*, const UI_32 *f_uiSize*)

Brief

Constructor for CNSSharedMemWriter

Parameters:

in	*f_cSharedMemName*	const std::string& - name of the shared memory
in	*f_uiSize*	UI_32 - size of shared memory

Return values:

None	

Preconditons

-None

Change of internal status

None

Classification

public

Type

sync only

See also:

~CNSSharedMemWriter

CNSSharedMemWriter::~CNSSharedMemWriter ()

Brief

Destructor of CNSSharedMemWriter class.

Parameters:

None	

Return values:

none	

Preconditons

None

Change of internal status

-Change the state of shared memory to closed state

Classification

Public

Type

sync only

See also:

CNSSharedMemWriter(const std::string&, const BOOL), CNSSharedMemWriter(), Close

---

Member Function Documentation

E_CWORD33_Status CNSSharedMemWriter::ClearBuf ()

Brief

This function clears the shared memory buffer.

Return values:

e_CWORD33_StatusOK	if shared memory cleared successfully
e_CWORD33_StatusNullPointer	pointer of the CNSSharedMem object is NULL
e_CWORD33_StatusFail	if shared memory is not open

Preconditons

None

Change of internal status

shared memory is changed.

Classification

Public

Type

Sync only

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:
None

E_CWORD33_Status CNSSharedMemWriter::Close ()

Brief
This function unmaps the shared memory object.

Return values:

e_CWORD33_StatusOK	if shared memory closed successfully
e_CWORD33_StatusNullPointer	pointer of the CNSSharedMem object is NULL
e_CWORD33_StatusFail	if shared memory is not open

Preconditons

None

Change of internal status

Close the shared memory object

Classification

Public

Type

Sync only

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

Open

BOOL CNSSharedMemWriter::IsOpen ()

Brief

This function is used to check whether the shared memory buffer is opened or not.

Return values:

TRUE	- Open
FALSE	- Not open

Preconditons

None

Change of internal status

-None

Classification

public

Type

Sync only

See also:

none

E_CWORD33_Status CNSSharedMemWriter::Open ()

Brief

This function opens and maps the shared memory object.

Parameters:

None	

Return values:

e_CWORD33_StatusOK	if shared memory opened successfully
e_CWORD33_StatusNullPointer	pointer of the CNSSharedMem object is NULL
e_CWORD33_StatusFail	unable to open shared memory

Preconditons

None

Change of internal status

-Open shared memory object

Classification

public

Type

sync only

e_CWORD33_Status:Result
e_CWORD33_StatusOK:Success
Except e_CWORD33_StatusOK:Failure

See also:

Close

SI_32 CNSSharedMemWriter::Write (PCSTR *buffer*, const UI_32 *f_uilength*)

Brief

This function writes the data into the shared memory.

Parameters:

in	*buffer*	PSTR - pointer to the buffer containing the data to be written
in	*f_uilength*	const UI_32 - length of the data buffer to be written

Return values:

The	number of bytes written
NS_SHM_ERROR	if an error occurred

Preconditons

-Open the shared memory object

Change of internal status

shared memory is changed.

Classification

Public

Type

Sync only

l_iWriteSize:Result
NS_SHM_ERROR:Failure
Except NS_SHM_ERROR:Success

See also:

None

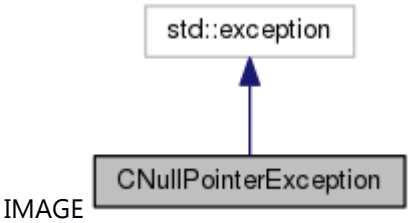
The documentation for this class was generated from the following file:

86 ns_sharedmem_writer.h

CNullPointerException Class Reference

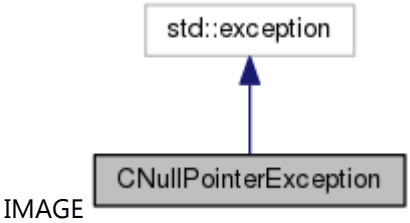
Defines the null pointer exception.

#include <_CWORD78__sm_framework_types.h>
Inheritance diagram for CNullPointerException:



IMAGE

Collaboration diagram for CNullPointerException:



IMAGE

Detailed Description

Defines the null pointer exception.

The documentation for this class was generated from the following file:
87 _CWORD78__sm_framework_types.h

CPassThruInDataHandler Class Reference

Class: CPassThruInDataHandler.

#include <ns_rcs_data_handler.hpp>

Public Member Functions

CPassThruInDataHandler (UI_8 *f_pui8Data, UI_16 f_ui16Length, UI_16 f_ui16ParamCount)
~CPassThruInDataHandler ()
E_CWORD33_Status GetBool (NSRCS_BOOL &f_rcsboolOutValue)
E_CWORD33_Status GetUI8 (UI_8 &f_ui8OutValue)
E_CWORD33_Status GetUI16 (UI_16 &f_ui16OutValue)
E_CWORD33_Status GetUI32 (UI_32 &f_ui32OutValue)
E_CWORD33_Status GetUI64 (UI_64 &f_ui64OutValue)
E_CWORD33_Status GetSI8 (SI_8 &f_si8OutValue)
E_CWORD33_Status GetSI16 (SI_16 &f_si16OutValue)
E_CWORD33_Status GetSI32 (SI_32 &f_si32OutValue)
E_CWORD33_Status GetSI64 (SI_64 &f_si64OutValue)
E_CWORD33_Status GetBuffer (UI_8 *&f_pui8Buffer, UI_16 &f_ui16BufLength)
E_CWORD33_Status GetArrayCount (UI_16 &f_ui16NoOfArrayElements)
UI_16 **GetDataLength** ()

Detailed Description

Class: CPassThruInDataHandler.

Data Handler

Brief Introduction

Class to get/set function to do input data handler

Constructor & Destructor Documentation

CPassThruInDataHandler::CPassThruInDataHandler (UI_8 * ***f_pui8Data*****, UI_16** ***f_ui16Length*****, UI_16** ***f_ui16ParamCount*****)[inline]**

Summary

Constructor of CPassThruInDataHandler 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:

~CPassThruInDataHandler

CPassThruInDataHandler::~CPassThruInDataHandler ()[inline]

~

Summary

Destructor of CPassThruInDataHandler 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:

CPassThruInDataHandler

Member Function Documentation

E_CWORD33_Status CPassThruInDataHandler::GetArrayCount (UI_16 & *f_ui16NoOfArrayElements*)[inline]

Brief

This function is used to get the buffer and its size.

Parameters:

out	*f_ui16NoOfArrayElements*	UI_16 - buffer size

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- If no array id found at the current location
e_CWORD33_StatusFail	- no sufficient buffer to read the data

Prerequisite

None

Change of internal state

None

Classification

Public

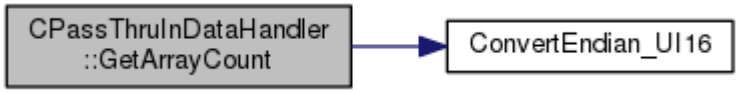
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

**E_CWORD33_Status CPassThruInDataHandler::GetBool (NSRCS_BOOL &
f_rcsboolOutValue)[inline]**

Brief

This function is used to convert value to type NSRCS_BOOL

Parameters:

out	*f_rcsboolOutValue*	NSRCS_BOOL & - the given type of covert result

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed
e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

E_CWORD33_Status CPassThruInDataHandler::GetBuffer (UI_8 *& *f_pui8Buffer*, UI_16 & *f_ui16BufLength*)[inline]

Brief

This function is used to get the buffer and its size.

Parameters:

out	*f_pui8Buffer*	UI_8 *& - buffer address
out	*f_ui16BufLength*	UI_16 - buffer size

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed
e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

Type

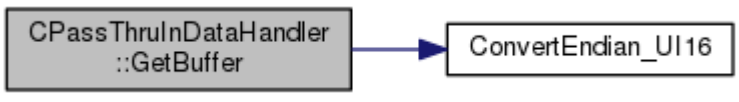
Sync Only

See also:

none

Here is the call graph for this function:

IMAGE



E_CWORD33_Status CPassThruInDataHandler::GetSI16 (SI_16 & *f_si16OutValue*)[inline]

Brief

This function is used to convert value to type SI_16

Parameters:

out	*f_rcsboolOutValue*	SI_16 & - the given type of covert result

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed
e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

Type

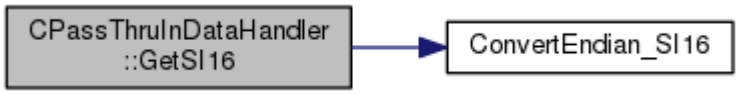
Sync Only

See also:

none

Here is the call graph for this function:

IMAGE



E_CWORD33_Status CPassThruInDataHandler::GetSI32 (SI_32 & *f_si32OutValue*)[inline]

Brief

This function is used to convert value to type SI_32

Parameters:

out	*f_rcsboolOutValue*	SI_32 & - the given type of covert result

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed
e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

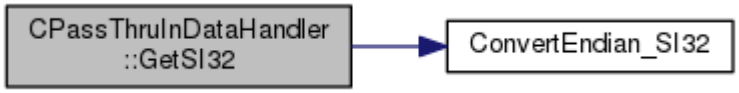
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruInDataHandler::GetSI64 (SI_64 & *f_si64OutValue*)[inline]

Brief

This function is used to convert value to type SI_64

Parameters:

out	*f_rcsboolOutValue*	SI_64 & - the given type of covert result

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed
e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

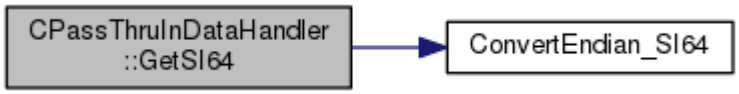
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruInDataHandler::GetSI8 (SI_8 & *f_si8OutValue*)[inline]

Brief

This function is used to convert value to type SI_8

Parameters:

out	*f_rcsboolOutValue*	SI_8 & - the given type of covert result

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed
e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

E_CWORD33_Status CPassThruInDataHandler::GetUI16 (UI_16 & *f_ui16OutValue*)[inline]

Brief

This function is used to convert value to type UI_16

Parameters:

out	*f_rcsboolOutValue*	UI_16 & - the given type of covert result

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed

e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

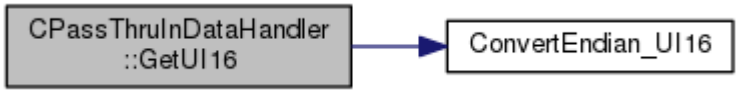
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruInDataHandler::GetUI32 (UI_32 & *f_ui32OutValue*)[inline]

Brief

This function is used to convert value to type UI_32

Parameters:

out	*f_rcsboolOutValue*	UI_32 & - the given type of covert result

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed
e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

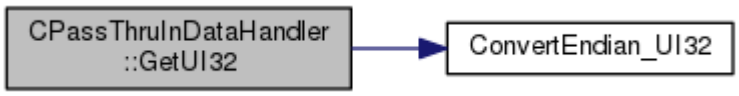
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruInDataHandler::GetUI64 (UI_64 & *f_ui64OutValue*)[inline]

Brief

This function is used to convert value to type UI_64

Parameters:

out	*f_rcsboolOutValue*	UI_64 & - the given type of covert result

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed
e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

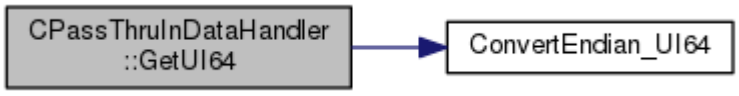
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruInDataHandler::GetUI8 (UI_8 & *f_ui8OutValue*)[inline]

Brief

This function is used to convert value to type UI_8

Parameters:

out	*f_rcsboolOutValue*	UI_8 & - the given type of covert result

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD78_StatusInvldID	- invalid data type id passed
e_CWORD33_StatusFail	- no sufficient data is available

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

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

88 ns_rcs_data_handler.hpp

CPassThruOutDataHandler Class Reference

Data Handler.

#include <ns_rcs_data_handler.hpp>

Public Member Functions

CPassThruOutDataHandler (UI_8 f_ui8ClientId, UI_8 f_ui8PluginId, UI_32 f_ui32CommandId)
~CPassThruOutDataHandler ()
UI_8 GetClientID ()
UI_8 * GetData ()
UI_16 GetLength ()
E_CWORD33_Status SetBool (NSRCS_BOOL &f_rcsboolValue)
E_CWORD33_Status SetUI8 (UI_8 f_ui8Value)
E_CWORD33_Status SetUI16 (UI_16 f_ui16Value)
E_CWORD33_Status SetUI32 (UI_32 f_ui32Value)
E_CWORD33_Status SetUI64 (UI_64 f_ui64Value)
E_CWORD33_Status SetSI8 (SI_8 f_si8Value)
E_CWORD33_Status SetSI16 (SI_16 f_si16Value)
E_CWORD33_Status SetSI32 (SI_32 f_si32Value)
E_CWORD33_Status SetSI64 (SI_64 f_si64Value)
E_CWORD33_Status SetBuffer (UI_8 *f_pui8Buffer, UI_16 f_ui16BufLength)
E_CWORD33_Status SetArrayCount (UI_16 f_ui16ArrayCount)

Detailed Description

Data Handler.

Brief Introduction

Class to get/set function to do output data handler

Constructor & Destructor Documentation

CPassThruOutDataHandler::CPassThruOutDataHandler (UI_8 *f_ui8ClientId*, UI_8 *f_ui8PluginId*, UI_32 *f_ui32CommandId*)[inline]

Summary

Constructor of CPassThruOutDataHandler class.

Parameters:

None	

Return values:

None	

Preconditions

None.

Change of the internal state

The internal state is not changed.

Classification

Public

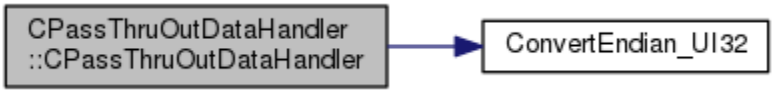
Type

None

See also:

~CPassThruOutDataHandler

Here is the call graph for this function:

IMAGE 

CPassThruOutDataHandler::~CPassThruOutDataHandler ()[inline]

~

Summary

Destructor of CPassThruOutDataHandler 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:

CPassThruOutDataHandler

Member Function Documentation

UI_8 CPassThruOutDataHandler::GetClientID ()[inline]

Brief

This function is used to get clinet id.

Parameters:

none	

Return values:

UI_8	- client id

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

UI_8* CPassThruOutDataHandler::GetData ()[inline]

Brief

This function is used to get m_vui8Data data address.

Parameters:

none	

Return values:

UI_8	* - m_vui8Data data address

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

UI_16 CPassThruOutDataHandler::GetLength ()[inline]

Brief

This function is used to get m_ui16CurOffset value.

Parameters:

none	

Return values:

UI_16	- m_ui16CurOffset value

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

E_CWORD33_Status CPassThruOutDataHandler::SetArrayCount (UI_16 *f_ui16ArrayCount*)[inline]

Brief

This function is used to set ArrayCount.

Parameters:

f_ui16ArrayCount	UI_16 - ArrayCount

Return values:

none	

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruOutDataHandler::SetBool (NSRCS_BOOL & *f_rcsboolValue*)[inline]

Brief

This function is used to set NSRCS_BOOL type value.

Parameters:

in	*f_rcsboolValue*	NSRCS_BOOL - new NSRCS_BOOL value

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- set array element of different type

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

E_CWORD33_Status CPassThruOutDataHandler::SetBuffer (UI_8 * *f_pui8Buffer*, UI_16 *f_ui16BufLength*)[inline]

Brief

This function is used to set buffer and size.

Parameters:

in	*f_pui8Buffer*	UI_8* - buffer address
in	*f_ui16BufLength*	UI_16 - buffer size

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- data member(m_ui8LastArrDataType) is invalid value

Prerequisite

None

Change of internal state

None

Classification

Public

Type

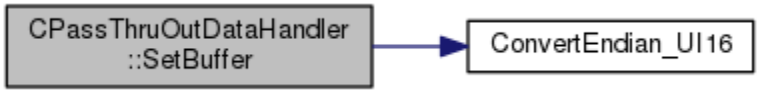
Sync Only

See also:

none

Here is the call graph for this function:

IMAGE



E_CWORD33_Status CPassThruOutDataHandler::SetSI16 (SI_16 *f_si16Value*)[inline]

Brief

This function is used to set SI_16 type value.

Parameters:

in	*f_si16Value*	SI_16 - new SI_16 value

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- set array element of different type

Prerequisite

None

Change of internal state

None

Classification

Public

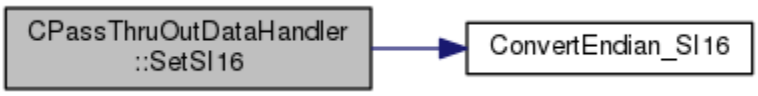
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruOutDataHandler::SetSI32 (SI_32 *f_si32Value*)[inline]

Brief

This function is used to set SI_32 type value.

Parameters:

in	*f_si32Value*	SI_32 - new SI_32 value

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- set array element of different type

Prerequisite

None

Change of internal state

None

Classification

Public

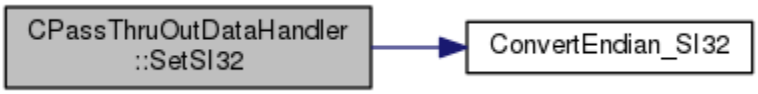
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruOutDataHandler::SetSI64 (SI_64 *f_si64Value*)[inline]

Brief

This function is used to set SI_64 type value.

Parameters:

in	*f_si64Value*	SI_64 - new SI_64 value

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- set array element of different type

Prerequisite

None

Change of internal state

None

Classification

Public

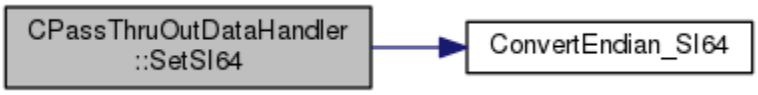
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruOutDataHandler::SetSI8 (SI_8 *f_si8Value*)[inline]

Brief

This function is used to set SI_8 type value.

Parameters:

in	*f_si8Value*	SI_8 - new SI_8 value

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- set array element of different type

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

E_CWORD33_Status CPassThruOutDataHandler::SetUI16 (UI_16 *f_ui16Value*)[inline]

Brief

This function is used to set UI_16 type value.

Parameters:

in	*f_ui16Value*	UI_16 - new UI_16 value

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- set array element of different type

Prerequisite

None

Change of internal state

None

Classification

Public

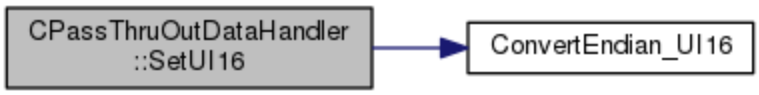
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruOutDataHandler::SetUI32 (UI_32 *f_ui32Value*)[inline]

Brief

This function is used to set UI_32 type value.

Parameters:

in	*f_ui32Value*	UI_32 - new UI_32 value

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- set array element of different type

Prerequisite

None

Change of internal state

None

Classification

Public

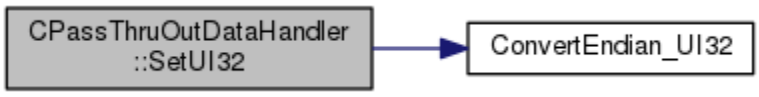
Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruOutDataHandler::SetUI64 (UI_64 *f_ui64Value*)[inline]

Brief

This function is used to set UI_64 type value.

Parameters:

in	*f_ui64Value*	UI_64 - new UI_64 value

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- set array element of different type

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

Here is the call graph for this function:

IMAGE 

E_CWORD33_Status CPassThruOutDataHandler::SetUI8 (UI_8 *f_ui8Value*)[inline]

Brief

This function is used to set UI_8 type value.

Parameters:

in	*f_ui8Value*	UI_8 - new UI_8 value

Return values:

E_CWORD33_Status	
e_CWORD33_StatusOK	- data retrieved
e_CWORD33_StatusInvldID	- set array element of different type

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

none

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

89 ns_rcs_data_handler.hpp

CResCallback< C, M > Class Template Reference

Static Public Member Functions

static ResponseServiceTo_CWORD77_ **set** (void *pInst)
static E_CWORD33_Status **call** (HANDLE y, E_CWORD77_ResponseType ResponseType, std::string ServiceName)

The documentation for this class was generated from the following file:
90 ns__CWORD77__common.h

CRWLock Class Reference

Read/Write Lock Class.

#include <ns_utility_sys.hpp>

Public Member Functions

CRWLock ()
~CRWLock ()
void ReadLock ()
void WriteLock ()
void Unlock ()

Detailed Description

Read/Write Lock Class.

Brief Introduction

This class defines locking policy for reader/writer.

Constructor & Destructor Documentation

CRWLock::CRWLock ()

Brief

Construct a CRWLock object.

Prerequisite

None

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

~CRWLock()

CRWLock::~CRWLock ()

Brief

Destruct the CRWLock object.

Prerequisite

Construct a CRWLock object.

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CRWLock()

Member Function Documentation

void CRWLock::ReadLock ()

Brief

Lock the CRWLock object when reading.

Return values:

void	

Prerequisite

Construct a CRWLock object.

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CRWLock(), WriteLock(), UnLoack()

void CRWLock::Unlock ()

Brief

Open the lock of CRWLock object

Return values:

void	

Prerequisite

Construct a CRWLock object.

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CRWLock(), ReadLock(), WriteLock()

void CRWLock::WriteLock ()

Brief

Lock the CRWLock object when writing.

Return values:

void	

Prerequisite

Construct a CRWLock object.

Change of internal state

None

Classification

Public

Type

Sync Only

See also:

CRWLock(), ReadLock(), UnLock()

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

91 ns_utility_sys.hpp

CTestCaseData Class Reference

CTestCaseData : Class used to fill testcase data from XML.

#include <XMLParser.h>

Public Member Functions

CTestCaseData (const xmlChar *name, std::string testDelaySec, std::string testDelayMSec, std::string testDelayUSec, std::string testLoopVal, std::string testReset, std::string testPrintResult, std::string testCategory, std::string testResultType, std::vector< std::string > paramMacroVector, std::string macroDelaySecVal, std::string macroDelayMSecVal, std::string macroDelayUSecVal, std::list< std::string > expOutputVector)

CTestCaseData (const CTestCaseData &value)

Public Attributes

std::string **key**
SI_32 **m_testDelaySec**
SI_32 **m_testDelayMSec**
SI_32 **m_testDelayUSec**
SI_32 **m_testLoop**
SI_32 **m_testReset**
SI_32 **m_testPrintResult**
std::string **m_testCategory**
std::string **m_testResultType**
std::vector< std::string > **macroVector**
std::string **m_macroDelaySecVal**
std::string **m_macroDelayMsecVal**
std::string **m_macroDelayUsecVal**
std::list< std::string > **expOut**

Detailed Description

CTestCaseData : Class used to fill testcase data from XML.

The documentation for this class was generated from the following file:
92 XMLParser.h

CXmlAttr Class Reference

This class represents the attribute of an xml node.

#include <ns_xmlparser_if.h>

Public Member Functions

CXmlAttr ()
CXmlAttr (std::string f_cKey, std::string f_cValue)
~CXmlAttr ()

Public Attributes

std::string **m_cKey**
std::string **m_cValue**

Detailed Description

This class represents the attribute of an xml node.

Brief Introduction

This class represents the attribute of an xml node.

Constructor & Destructor Documentation

CXmlAttr::CXmlAttr ()

Brief

CXmlAttr constructor

Parameters:

None	

Return values:

None	

Prerequisite

Load the shared library first

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

None

CXmlAttr::CXmlAttr (std::string *f_cKey*, std::string *f_cValue*)

Brief

CXmlAttr constructor

Parameters:

[IN]	f_cKey std::string - attributes key
[IN]	f_cValue std::string - attributes value

Return values:

None	

Prerequisite

Load the shared library first

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

None

CXmlAttr::~CXmlAttr ()

Brief

Destructor of CXmlAttr class

Parameters:

None	

Return values:

None	

Prerequisite

None

Inside state change

None

None

Classification

Public

Type

Sync Only

See also:

None

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

93 ns_xmlparser_if.h

CXmlNode Class Reference

This class represents node of an XML.

#include <ns_xmlparser_if.h>

Public Member Functions

~CXmlNode ()
BOOL IsNull ()
E_CWORD33_XmlNodeTypes Type ()
std::string Name ()
std::string GetContent ()
E_CWORD33_Status SetContent (std::string f_cUpdatedText)
E_CWORD33_Status GetAttributeValue (std::string f_cKey, std::string &f_cAttributeValue)
E_CWORD33_Status SetAttributeValue (std::string f_cKey, std::string f_cValue)
E_CWORD33_Status AddNewAttribute (std::string f_cKey, std::string f_cAttributeValue)
E_CWORD33_Status RemoveAttribute (std::string f_cKey)
TAttrList GetAttributeItems ()
CXmlNode Parent ()
CXmlNode FirstChild ()
CXmlNode LastChild ()
TNodeList Children ()
CXmlNode NextSibling ()
CXmlNode PrevSibling ()
CXmlNode FindChildNode (std::string f_cNodeName)
TNodeList FindAllChildNodes (std::string f_cNodeName)
E_CWORD33_Status GetContentOfChildNode (std::string f_cNodeName, std::string &f_cContent)
CXmlNode AddChildNode (std::string f_cNodeName, std::string f_cText="")
CXmlNode AddSiblingNode (std::string f_cNodeName, std::string f_cText="")
E_CWORD33_Status RemoveChildNode (std::string f_cNodeName)
E_CWORD33_Status ClearNode ()

Friends

class **CXmlParser**

Detailed Description

This class represents node of an XML.

Brief Introduction

Provides API for performing operations on node.

Constructor & Destructor Documentation

CXmlNode::~CXmlNode ()

Brief

Destructor of CXmlNode class

Parameters:

None	

Return values:

None	

Prerequisite

None

Inside state change

None

None

Classification

Public

Type

Sync Only

See also:

None

Member Function Documentation

CXmlNode CXmlNode::AddChildNode (std::string *f_cNodeName*, std::string *f_cText* = "")

Brief

Add sub node

Parameters:

[IN]	f_cNodeName std::string - tag of the new node
[IN]	f_cText std::string - content of the new node

Return values:

CXmlNode	- object of child node

Prerequisite

Load the shared library first

Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

E_CWORD33_Status CXmlNode::AddNewAttribute (std::string *f_cKey*, std::string *f_cAttributeValue*)

Brief

Add new attribute

Parameters:

[IN]	f_cKey std::string - attribute name
[IN]	f_cValue std::string - attribute value

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

CXmlNode CXmlNode::AddSiblingNode (std::string *f_cNodeName*, std::string *f_cText* = "")

Brief

Add new node to siblings

Parameters:

[IN]	f_cNodeName std::string - tag of the new node
[IN]	f_cText std::string - content of the new node

Return values:

CXmlNode	- object of sibling node

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

TNodeList CXmlNode::Children ()

Brief

Get child list

Parameters:

None	

Return values:

TNodeList	- list of all child node object

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

E_CWORD33_Status CXmlNode::ClearNode ()

Brief

Removes elements and attributes

Parameters:

None	

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

TNodeList CXmlNode::FindAllChildNodes (std::string *f_cNodeName*)

Brief

Get list of all the immediate sub node

Parameters:

[IN]	f_cNodeName std::string - tag name of the node

Return values:

TNodeList	- list of matching nodes

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

CXmlNode CXmlNode::FindChildNode (std::string *f_cNodeName*)

Brief

Find child node

Parameters:

[IN]	f_cNodeName std::string - tag name of the node

Return values:

CXmlNode	- node object

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

CXmlNode CXmlNode::FirstChild ()

Brief

Get first child

Parameters:

None	

Return values:

CXmlNode	- node object

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

TAttrList CXmlNode::GetAttributeItems ()

Brief

Get attributes list

Parameters:

None	

Return values:

TAttrList	- list of all attributes of node or empty list in case of no attributes

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

E_CWORD33_Status CXmlNode::GetAttributeValue (std::string *f_cKey*, std::string & *f_cAttributeValue*)

Brief

Get attribute

Parameters:

[IN]	f_cKey std::string - attribute name
[OUT]	f_cAttributeValue std::string& - attribute value

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

std::string CXmlNode::GetContent ()

Brief

Get content

Parameters:

None	

Return values:

std::string	content of current node

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

E_CWORD33_Status CXmlNode::GetContentOfChildNode (std::string *f_cNodeName*, std::string & *f_cContent*)

Brief

Find first level node

Parameters:

[IN]	f_cNodeName std::string - name of the node
[OUT]	f_cContent std::string& - content of the node

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusFail	Failed

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

BOOL CXmlNode::IsNull ()

Brief

Check xmlNodePtr

Parameters:

None	

Return values:

BOOL	true if valid else false

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

CXmlNode CXmlNode::LastChild ()

Brief

Get last child

Parameters:

None	

Return values:

CXmlNode	- node object

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

std::string CXmlNode::Name ()

Brief

Get name

Parameters:

None	

Return values:

std::string	node name

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

CXmlNode CXmlNode::NextSibling ()

Brief

Get next node

Parameters:

None	

Return values:

CXmlNode	- node object

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

CXmlNode CXmlNode::Parent ()

Brief

Get parent

Parameters:

None	

Return values:

CXmlNode	- node object

Prerequisite

Load the shared library first

Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

CXmlNode CXmlNode::PrevSibling ()

Brief

Get previous node

Parameters:

None	

Return values:

CXmlNode	- node object

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

E_CWORD33_Status CXmlNode::RemoveAttribute (std::string *f_cKey*)

Brief

Removes attribute

Parameters:

[IN]	f_cKey std::string - attribute name

Return values:

e_CWORD33_StatusOK	Success

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

E_CWORD33_Status CXmlNode::RemoveChildNode (std::string *f_cNodeName*)

Brief

Remove child node

Parameters:

[IN]	f_cNodeName std::string - name of child node

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

E_CWORD33_Status CXmlNode::SetAttributeValue (std::string *f_cKey*, std::string *f_cValue*)

Brief

Set attribute

Parameters:

[IN]	f_cKey std::string - attribute name
[IN]	f_cValue std::string - attribute value

Return values:

e_CWORD33_StatusOK	Success

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

E_CWORD33_Status CXmlNode::SetContent (std::string *f_cUpdatedText*)

Brief

Set content

Parameters:

[IN]	f_cUpdatedText std::string - updated value

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

E_CWORD33_XmlNodeTypes CXmlNode::Type ()

Brief

Get type

Parameters:

None	

Return values:

_CWORD33__XML_NODE_NONE	not exist node
_CWORD33__XML_ELEMENT_NODE	element node
_CWORD33__XML_TEXT_NODE	text node
_CWORD33__XML_CDATA_SECTION_NODE	section node
_CWORD33__XML_ENTITY_REF_NODE	entity ref node
_CWORD33__XML_ENTITY_NODE	entity node
_CWORD33__XML_PI_NODE	PI node
_CWORD33__XML_COMMENT_NODE	comment node
_CWORD33__XML_DOCUMENT_NODE	document node
_CWORD33__XML_DOCUMENT_TYPE_NODE	document type node
_CWORD33__XML_DOCUMENT_FRAG_NODE	document frag node
_CWORD33__XML_NOTATION_NODE	notation node
_CWORD33__XML_HTML_DOCUMENT_NODE	document node
_CWORD33__XML_DTD_NODE	dtd node
_CWORD33__XML_ELEMENT_DECL	element decl
_CWORD33__XML_ATTRIBUTE_DECL	attribute decl

_CWORD33__XML_ENTITY_DECL	entity decl
_CWORD33__XML_NAMESPACE_DECL	namespace decl
_CWORD33__XML_XINCLUDE_START	include start
_CWORD33__XML_XINCLUDE_END	include end
_CWORD33__XML_DOCB_DOCUMENT_NODE	document node

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlNode

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

94 ns_xmlparser_if.h

CXmlParser Class Reference

This class represents the XML parser.

#include <ns_xmlparser_if.h>

Public Member Functions

CXmlParser ()
~CXmlParser ()
E_CWORD33_Status ParseXml (std::string f_cFileName)
E_CWORD33_Status CreateNewXmlDoc (std::string f_cRootNodeName)
E_CWORD33_Status SaveXml (std::string f_cFileName="")
CXmlNode GetRootNode ()
CXmlNode AddNewNode (CXmlNode m_pParentNode, std::string f_cNewNodeName, std::string f_cContent="")
E_CWORD33_Status RemoveNode (CXmlNode m_pNode)
CXmlNode FindNode (std::string f_cNodePath, CXmlNode f_pCurrentNode)
TNodeList FindAllNodes (std::string f_cNodePath, CXmlNode f_pCurrentNode)

Detailed Description

This class represents the XML parser.

Brief Introduction

Provides API for parsing xml file. Also provides API to search node/nodes based on Xml Path

---

Constructor & Destructor Documentation

CXmlParser::CXmlParser ()

Brief

CXmlParser constructor

Parameters:

None	

Return values:

None	

Prerequisite

Load the shared library first

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

None

CXmlParser::~CXmlParser ()

Brief

Destructor of CXmlParser class

Parameters:

None	

Return values:

None	

Prerequisite

None

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlParser

Member Function Documentation

CXmlNode CXmlParser::AddNewNode (CXmlNode *m_pParentNode*, std::string *f_cNewNodeName*, std::string *f_cContent* = "")

Brief

Create and add node

Parameters:

[IN]	f_pParentNode CXmlNode - parent node
[IN]	f_cNewNodeName std::string - tag name of the new node
[IN]	f_cContent std::string - text to set for the new node

Return values:

CXmlNode	- new node object

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlParser

E_CWORD33_Status CXmlParser::CreateNewXmlDoc (std::string *f_cRootNodeName*)

Brief

Creates new xml

Parameters:

[IN]	f_cRootNodeName std::string - root node name in new doc

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlParser

TNodeList CXmlParser::FindAllNodes (std::string *f_cNodePath*, CXmlNode *f_pCurrentNode*)

Brief

Find all matching node

Parameters:

[IN]	f_cNodePath std::string - name or path of node(XPath) path is relative to current node or absolute path
[IN]	f_pCurrentNode CXmlNode - current node

Return values:

TNodeList	- list of matching nodes

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlParser

CXmlNode CXmlParser::FindNode (std::string *f_cNodePath*, CXmlNode *f_pCurrentNode*)

Brief

Find first matching node

Parameters:

[IN]	f_cNodePath std::string - name or path of node(XPath) path is relative to current node or absolute path

[IN]	f_pCurrentNode CXmlNode - current node

Return values:

CXmlNode	- node object

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlParser

CXmlNode CXmlParser::GetRootNode ()

Brief

Get root node

Parameters:

None	

Return values:

CXmlNode	- root node object

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlParser

E_CWORD33_Status CXmlParser::ParseXml (std::string *f_cFileName*)

Brief

Parses xml file

Parameters:

[IN]	f_cFileName std::string - full path of xml file to be parsed

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlParser

E_CWORD33_Status CXmlParser::RemoveNode (CXmlNode *m_pNode*)

Brief

Remove node

Parameters:

[IN]	f_pNode CXmlNode - node object

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlParser

E_CWORD33_Status CXmlParser::SaveXml (std::string *f_cFileName* = "")

Brief

Save XML file

Parameters:

[IN]	f_cFileName std::string - full path of output xml file In case of parsing an xml file, if f_cFileName is empty, output will be saved in source xml file else output will be saved in user provided file path In case of creating new xml document, if f_cFileName is empty, returns error else output will be saved in user provided file path

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Get CXmlNode from CXmlParser

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync Only

See also:

CXmlParser

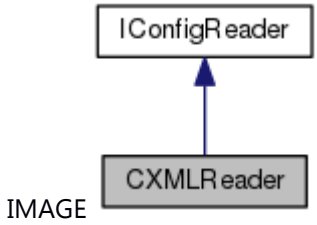
The documentation for this class was generated from the following file:

95 ns_xmlparser_if.h

CXMLReader Class Reference

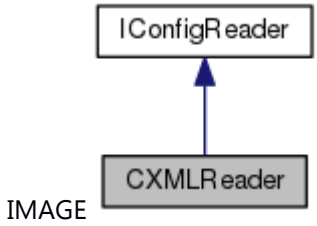
This file contains declaration of class CXMLReader.

#include <ns_xml_reader.h>
Inheritance diagram for CXMLReader:



IMAGE

Collaboration diagram for CXMLReader:



IMAGE

Public Member Functions

CXMLReader ()
CXMLReader (const std::string &f_cFilePath)
~CXMLReader ()
E_CWORD33_Status ParseFile (const std::string &f_cFilePath)
std::string GetValue (const std::string &f_cKey)
E_CWORD33_Status GetValue (const std::string &f_cKey, std::string &f_cValue)
PVOID GetDataPtr ()

Detailed Description

This file contains declaration of class CXMLReader.

this file has the CXMLReader class definitions

Brief Introduction

This class provides functionalities to read from XML config file.

Constructor & Destructor Documentation

CXMLReader::CXMLReader ()

Brief

This API constructor of CXMLReader class.

Parameters:

None	

Return values:

None	

Prerequisite

Load the shared library first

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

None

CXMLReader::CXMLReader (const std::string & *f_cFilePath*)

Brief

This API constructor of CXMLReader class.

Parameters:

[IN]	f_cFilePath std::string - Full path of the configuration file

Return values:

None	

Prerequisite

Load the shared library first

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

GetCXMLReaderObject

CXMLReader::~CXMLReader ()

~

CXMLReader

Destructor of CXMLReader class.

Parameters:

None	

Return values:

none	

Preconditons

None

Change of internal status

Close shared memory object.

Classification

Public

Type

sync only

See also:

CXMLReader

Member Function Documentation

PVOID CXMLReader::GetDataPtr ()[virtual]

Brief

Get Value with key

Parameters:

None	

Return values:

PVOID	- pointer of data structure

Prerequisite

Load the shared library first.
Create XmlReader object.

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

CXMLReader or GetCXMLReaderObject

Implements IConfigReader.

std::string CXMLReader::GetValue (const std::string & *f_cKey*)[virtual]

Brief

Get Value with key

Parameters:

[IN]	f_cKey std::string - key to search

Return values:

std::string	- value for key

Prerequisite

Load the shared library first.
Create XmlReader object.

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

CXMLReader or GetCXMLReaderObject

Implements IConfigReader.

E_CWORD33_Status CXMLReader::GetValue (const std::string & *f_cKey*, std::string & *f_cValue*)[virtual]

Brief

Get Value with key

Parameters:

[IN]	f_cKey std::string - key to search
[REF]	f_cValue std::string - Value of key

Return values:

e_CWORD78_StatusOK	Success / Pass / OK
e_CWORD33_StatusFail	Failed
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first.
Create XmlReader object.

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

CXMLReader or GetCXMLReaderObject

Implements IConfigReader.

E_CWORD33_Status CXMLReader::ParseFile (const std::string & *f_cFilePath*)[virtual]

Brief

This API parser file

Parameters:

[IN]	f_cFilePath std::string - path of file to parse

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first.
Create XmlReader object.

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

CXMLReader or GetCXMLReaderObject

Implements IConfigReader.

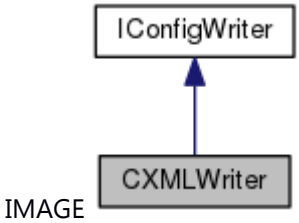
The documentation for this class was generated from the following file:

96 ns_xml_reader.h

CXMLWriter Class Reference

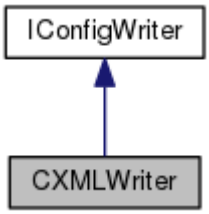
This file contains declaration of class CXMLWriter.

#include <ns_xml_writer.h>
Inheritance diagram for CXMLWriter:



IMAGE

Collaboration diagram for CXMLWriter:



IMAGE

Public Member Functions

CXMLWriter ()
CXMLWriter (const std::string &f_cFilePath)
~CXMLWriter ()
E_CWORD33_Status ParseFile (const std::string &f_cFilePath)
E_CWORD33_Status SetValue (const std::string &f_cKey, std::string f_cValue)
E_CWORD33_Status SaveData ()
E_CWORD33_Status SetPath (const std::string &f_cPath)
VOID SetDataPtr (PVOID f_pData)

Detailed Description

This file contains declaration of class CXMLWriter.

this file has the CXMLWriter class definitions

Brief Introduction

This class provides functionalities to write to XML config file

---

Constructor & Destructor Documentation

CXMLWriter::CXMLWriter ()

Brief

Parameterless Constructor of CXMLWriter class

Parameters:

None	

Return values:

None	

Prerequisite

Load the shared library first

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

GetCXMLWriterObjectNoParam

CXMLWriter::CXMLWriter (const std::string & *f_cFilePath*)

Brief

Constructor of CXMLWriter class

Parameters:

[IN]	f_cFilePath std::string - Full path of the configuration file

Return values:

None	

Prerequisite

Load the shared library first

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

GetCXMLWriterObject

CXMLWriter::~CXMLWriter ()

~

CXMLWriter

Destructor of CXMLWriter class.

Parameters:

None	

Return values:

none	

Preconditons

None

Change of internal status

-None

Classification

Public

Type

sync only

See also:

CXMLWriter

Member Function Documentation

E_CWORD33_Status CXMLWriter::ParseFile (const std::string & *f_cFilePath*)[virtual]

Brief

Parse file

Parameters:

[IN]	f_cFilePath std::string - path of file to parse

Return values:

e_CWORD33_StatusOK	Success
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Create CXMLWriter object

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

CXMLWriter

Implements IConfigWriter.

E_CWORD33_Status CXMLWriter::SaveData ()[virtual]

Brief

Save changed data

Parameters:

None	

Return values:

e_CWORD78_StatusOK	Success / Pass / OK
e_CWORD33_StatusFail	Failed

Prerequisite

Load the shared library first
Create CXMLWriter object
ParseFile set the path
Change the data

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

GetCXMLWriterObjectNoParam

Implements IConfigWriter.

VOID CXMLWriter::SetDataPtr (PVOID *f_pData*)[virtual]

Brief

Parse file

Parameters:

[IN]	f_pData PVOID - Pointer to data structure

Return values:

VOID	

Prerequisite

Load the shared library first
Create CXMLWriter object

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

CXMLWriter

Implements IConfigWriter.

E_CWORD33_Status CXMLWriter::SetPath (const std::string & *f_cPath*)[virtual]

Brief

Updata file path

Parameters:

[IN]	f_cPath std::string - Path of file

Return values:

e_CWORD78_StatusOK	Success / Pass / OK
e_CWORD33_StatusInvldParam	Invalid parameter

Prerequisite

Load the shared library first
Create CXMLWriter object

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

GetCXMLWriterObjectNoParam

Implements IConfigWriter.

E_CWORD33_Status CXMLWriter::SetValue (const std::string & *f_cKey*, std::string *f_cValue*)[virtual]

Brief

Set vaule for key

Parameters:

[IN]	f_cKey std::string - key to search
[IN]	f_cValue std::string - value to set

Return values:

e_CWORD78_StatusOK	Success / Pass / OK
e_CWORD33_StatusFail	Failed
e_CWORD33_StatusInvldParam	Invalid parameter
e_CWORD33_StatusNullPointer	Null pointer

Prerequisite

Load the shared library first
Create CXMLWriter object
ParseFile set the path

Inside state change

The API no inside state change

None

Classification

Public

Type

Sync

See also:

CXMLWriter,ParseFile

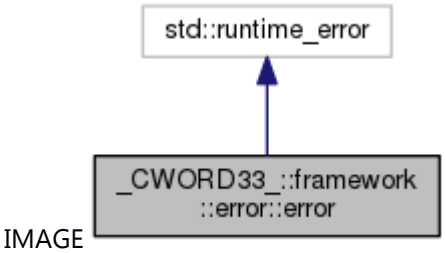
Implements IConfigWriter.

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

97 ns_xml_writer.h

CWORD33::framework::error::error Class Reference

Inheritance diagram for _CWORD33_::framework::error::error:



IMAGE

Collaboration diagram for _CWORD33_::framework::error::error:

IMAGE

Public Member Functions

error (E_CWORD33_Status error, PCSTR errorMsg)

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

98 _CWORD78__framework_error.hpp

FSig< R(*)() > Struct Template Reference

Public Types

typedef R **RType**

Static Public Attributes

static const UI_32 **argCount** = ***

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

99 ns_utility.hpp

FSig< R(*)(T1) > Struct Template Reference

Public Types

typedef R **RType**
typedef T1 **TArg1**

Static Public Attributes

static const UI_32 **argCount** = ***

The documentation for this struct was generated from the following file:
100 ns_utility.hpp

FSig< R(*)(T1, T2) > Struct Template Reference

Public Types

typedef R **RType**
typedef T1 **TArg1**
typedef T2 **TArg2**

Static Public Attributes

static const UI_32 **argCount** = ***

The documentation for this struct was generated from the following file:
101 ns_utility.hpp

FSig< R(*)(T1, T2, T3) > Struct Template Reference

Public Types

typedef R **RType**
typedef T1 **TArg1**
typedef T2 **TArg2**
typedef T3 **TArg3**

Static Public Attributes

static const UI_32 **argCount** = ***

The documentation for this struct was generated from the following file:
102 ns_utility.hpp

FSig< R(*)(T1, T2, T3, T4) > Struct Template Reference

Public Types

typedef R **RType**
typedef T1 **TArg1**
typedef T2 **TArg2**
typedef T3 **TArg3**
typedef T4 **TArg4**

Static Public Attributes

static const UI_32 **argCount** = ***

The documentation for this struct was generated from the following file:
103 ns_utility.hpp

FSig< R(*)(T1, T2, T3, T4, T5) > Struct Template Reference

Public Types

typedef R **RType**
typedef T1 **TArg1**
typedef T2 **TArg2**
typedef T3 **TArg3**
typedef T4 **TArg4**
typedef T5 **TArg5**

Static Public Attributes

static const UI_32 **argCount** = ***

The documentation for this struct was generated from the following file:
104 ns_utility.hpp

FSig< R(*)(T1, T2, T3, T4, T5, T6) > Struct Template Reference

Public Types

typedef R **RType**
typedef T1 **TArg1**
typedef T2 **TArg2**
typedef T3 **TArg3**
typedef T4 **TArg4**
typedef T5 **TArg5**
typedef T6 **TArg6**

Static Public Attributes

static const UI_32 **argCount** = ***

The documentation for this struct was generated from the following file:
105 ns_utility.hpp

FSig< R(*)(T1, T2, T3, T4, T5, T6, T7) > Struct Template Reference

Public Types

typedef R **RType**
typedef T1 **TArg1**
typedef T2 **TArg2**
typedef T3 **TArg3**
typedef T4 **TArg4**
typedef T5 **TArg5**
typedef T6 **TArg6**
typedef T7 **TArg7**

Static Public Attributes

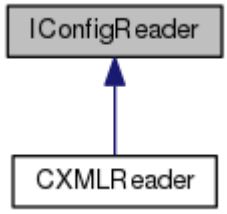
static const UI_32 **argCount** = ***

The documentation for this struct was generated from the following file:
106 ns_utility.hpp

IConfigReader Class Reference

IConfigReader.

#include <ns_reader.h>
Inheritance diagram for IConfigReader:

IMAGE 

Public Member Functions

IConfigReader ()
virtual ~IConfigReader ()
virtual E_CWORD33_Status ParseFile (const std::string &f_c_filepath)=0
virtual std::string GetValue (const std::string &f_c_key)=0
virtual E_CWORD33_Status GetValue (const std::string &f_c_key, std::string &f_c_value)=0
virtual PVOID GetDataPtr ()=0

Detailed Description

IConfigReader.

Brief Introduction

Class to provide class of parse config file.

Constructor & Destructor Documentation

IConfigReader::IConfigReader ()

Summary

Constructor of IConfigReader 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

sync only

See also:

~IConfigReader

virtual IConfigReader::~IConfigReader ()[virtual]

Summary

Destructor of IConfigReader 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

sync only

See also:

IConfigReader

Member Function Documentation

virtual PVOID IConfigReader::GetDataPtr ()[pure virtual]

Summary

This function is used to get the data pointer.

Parameters:

none.	

Return values:

PVOID	- pointer of data structure

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

IConfigReader

Implemented in CXMLReader.

virtual std::string IConfigReader::GetValue (const std::string & *f_c_key*)[pure virtual]

Summary

This function is used to get the value associated with the key.

Parameters:

[IN]	f_c_key const std::string - key to search.

Return values:

l_cValue	value for key

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

IConfigReader

Implemented in CXMLReader.

virtual E_CWORD33_Status IConfigReader::GetValue (const std::string & *f_c_key*, std::string & *f_c_value*)[pure virtual]

Summary

This function is used to get the value associated with the key.

Parameters:

[IN]	f_c_key const std::string - key to search.
[OUT]	f_c_value const std::string - value for key.

Return values:

E_CWORD33_Status	error error if key not found else e_CWORD33_StatusOK

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

IConfigReader

Implemented in CXMLReader.

virtual E_CWORD33_Status IConfigReader::ParseFile (const std::string & *f_c_filepath*)[pure virtual]

Summary

This function is used to parse the file.

Parameters:

[IN]	f_c_filepath const std::string - path of file to parse.

Return values:

E_CWORD33_Status	

Preconditions

None.

Change of the internal state

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

Classification

Public

Type

sync only

See also:

IConfigReader

Implemented in CXMLReader.

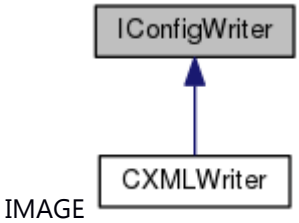
The documentation for this class was generated from the following file:

107 ns_reader.h

IConfigWriter Class Reference

Config writer abstract class.

#include <ns_writer.h>
Inheritance diagram for IConfigWriter:



IMAGE

Public Member Functions

IConfigWriter ()
virtual ~IConfigWriter ()
virtual E_CWORD33_Status ParseFile (const std::string &f_c_filepath)=0
virtual E_CWORD33_Status SetValue (const std::string &f_c_key, std::string f_c_value)=0
virtual E_CWORD33_Status SaveData ()=0
virtual E_CWORD33_Status SetPath (const std::string &f_cpath)=0
virtual VOID SetDataPtr (PVOID f_pdata)=0

Detailed Description

Config writer abstract class.

Brief Introduction

IConfigWriter is an abstract class which is inherited by
different type configuration file writer class.

Constructor & Destructor Documentation

IConfigWriter::IConfigWriter ()

Brief

Constructor of IConfigWriter class

Parameters:

None	

Return values:

none	

Preconditons

None

Change of internal status

None

Classification

Public

Type

Sync only

See also:

~IConfigWriter

virtual IConfigWriter::~IConfigWriter ()[virtual]

~

Brief

Destructor of IConfigWriter class

Parameters:

None	

Return values:

none	

Preconditons

Calls constructor to successfully create object.

Change of internal status

None

Classification

Public

Type

Sync only

See also:

IConfigWriter

Member Function Documentation

virtual E_CWORD33_Status IConfigWriter::ParseFile (const std::string & *f_c_filepath*)[pure virtual]

Brief

This function is used to parse the file.

Parameters:

in	*f_c_filepath*	std::string - path of file to parse

Return values:

E_CWORD33_Status	- success or failure

Preconditons

Calls constructor to successfully create object.

Change of internal status

None

Classification

Public

Type

Sync only

See also:

None

Implemented in CXMLWriter.

virtual E_CWORD33_Status IConfigWriter::SaveData ()[pure virtual]

Brief

This function is used to save the changed value permanently to the config source

Parameters:

None	

Return values:

none	

Preconditons

None

Change of internal status

None

Classification

Public

Type

Sync only

See also:

None

Implemented in CXMLWriter.

virtual VOID IConfigWriter::SetDataPtr (PVOID *f_pdata*)[pure virtual]

Brief

This function is used to set the data pointer in config writer class with data pointer created in config reader class.
This is needed to avoid recreation of same data structure object in both reader and writer
when we create object of NSConfigParser.

Parameters:

in	*f_pdata*	PVOID - Pointer to data structure

Return values:

None	

Preconditons

None

Change of internal status

None

Classification

Public

Type

Sync only

See also:

None

Implemented in CXMLWriter.

virtual E_CWORD33_Status IConfigWriter::SetPath (const std::string & *f_cpath*)[pure virtual]

Brief

This function is used to set the config file path

Parameters:

in	*f_cpath*	std::string - Path of file

Return values:

E_CWORD33_Status	- success or failure

Preconditons

None

Change of internal status

None

Classification

Public

Type

Sync only

See also:

None

Implemented in CXMLWriter.

virtual E_CWORD33_Status IConfigWriter::SetValue (const std::string & *f_c_key*, std::string *f_c_value*)[pure virtual]

Brief

This function is used to set the value for the key

Parameters:

in	*f_c_key*	std::string - key to search
in	*f_c_value*	std::string - value to set

Return values:

E_CWORD33_Status	- error if key not found else e_CWORD33_StatusOK

Preconditons

Calls constructor to successfully create object.

Change of internal status

None

Classification

Public

Type

Sync only

See also:

None

Implemented in CXMLWriter.

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

108 ns_writer.h

IFunctor< R > Class Template Reference

Public Member Functions

virtual R **operator()** () const =0
virtual UI_32 **size** () const =0

The documentation for this class was generated from the following file:
109 ns_utility.hpp

MemTraits< T > Class Template Reference

Public Types

typedef T * **Type**
typedef std::bad_alloc **Exception**

Static Public Member Functions

static void **Release** (Type rsrc)
static BOOL **BadValue** (Type rsrc)

The documentation for this class was generated from the following file:
110 ns_utility.hpp

NSTimer Class Reference

Handle Timer.

#include <ns_timer_if.hpp>

Public Member Functions

NSTimer ()
~NSTimer ()
E_CWORD33_Status SetNotifyMethod (UI_16 notifyCmdId, PCSTR notifyToAppName)
VOID SetRepeatTimer (BOOL repeatTimer)
VOID SetTime (UI_32 seconds, UI_64 msecs)
VOID SetTime (UI_32 hrs, UI_32 mins, UI_32 seconds, UI_64 msecs)
E_CWORD33_Status Start (UI_32 seconds, UI_64 msecs)
E_CWORD33_Status Start ()
E_CWORD33_Status Stop ()
BOOL IsRunning ()
UI_64 GetInterval ()

Detailed Description

Handle Timer.

Brief Introduction

Timer class is using to handle the timer.creating, starting, stopping,and so on.

Constructor & Destructor Documentation

NSTimer::NSTimer ()

Brief

Creates a timer object with default behavior as single timeout timer.

Return values:

none	

Preconditons

-None

Change of internal status

None

Classification

Public

Type

See also:

~NSTimer

NSTimer::~NSTimer ()

Brief

Deletes a timer object.

Return values:

none	

Preconditons

-Instance of this class is created successfully.

Change of internal status

None

Classification

Public

Type

-None

See also:

NSTimer

Member Function Documentation

UI_64 NSTimer::GetInterval ()

Brief

Returns timeout interval in milliseconds

Return values:

UI_64	- Timeout interval in milliseconds

Preconditons

-None

Change of internal status

None

Classification

Public

Type

-None

See also:

None

BOOL NSTimer::IsRunning ()

Brief

Returns timer running status

Return values:

BOOL	- TRUE if timer is running else FALSE

Preconditons

-None

Change of internal status

None

Classification

Public

Type

-None

See also:

Start, Start(UI_32, UI_64), SetNotifyMethod, NS_TimerSetTime

E_CWORD33_Status NSTimer::SetNotifyMethod (UI_16 *notifyCmdId*, PCSTR *notifyToAppName*)

Brief

This function overloads SetNotifyMethod(). On timeout, sends the command id to the message queue of specified application. Resets previous notify method if any.

Parameters:

in	*notifyCmdId*	UI_16 - command id.
in	*notifyToAppName*	PCSTR - Send command id to the message queue of this application

Return values:

E_CWORD33_Status	- Returns e_CWORD33_StatusOK if the action is set otherwise e_CWORD33_StatusFail
e_CWORD33_StatusOK	- Success
e_CWORD33_StatusInvldParam	- Invalid parameter
e_CWORD33_StatusFail	- process error

Preconditons

-None

Change of internal status

None

Classification

Public

Type

-None

See also:

McOpenSender, McClose, NS_TimerCreate

VOID NSTimer::SetRepeatTimer (BOOL *repeatTimer*)

Brief

Sets the timer as a repeat timer. Timer timeouts repeatedly as per the interval set. Default timer fires only once. Not effective, if the timer is running. Effective on next Start() or Start(seconds, msecs) call.

Parameters:

in	*repeatTimer*	BOOL - TRUE - set repeat timer FALSE - set single timeout timer. (Default behavior)

Return values:

none	

Preconditons

-None

Change of internal status

-None

Classification

Public

Type

-None

See also:

none

VOID NSTimer::SetTime (UI_32 *seconds*, UI_64 *msecs*)

Brief

Set the interval of timeout. Interval is addition of seconds and milliseconds passed as an arguments.

Parameters:

in	*seconds*	UI_32 - Time in seconds
in	*msecs*	UI_64 - Time in milliseconds

Return values:

none	

Preconditons

-None

Change of internal status

None

Classification

Public

Type

-None

See also:

none

VOID NSTimer::SetTime (UI_32 *hrs*, UI_32 *mins*, UI_32 *seconds*, UI_64 *msecs*)

Brief

This function overloads SetTime(). Set the interval of timeout. Interval is addition of hrs, mins, seconds and milliseconds passed as an arguments.

Parameters:

in	*hrs*	UI_32 - Time in hours
in	*mins*	UI_64 - Time in minutes
in	*seconds*	UI_32 - Time in seconds
in	*msecs*	UI_64 - Time in milliseconds

Return values:

none	

Preconditons

-None

Change of internal status

None

Classification

Public

Type

-None

See also:

none

E_CWORD33_Status NSTimer::Start (UI_32 *seconds*, UI_64 *msecs*)

Brief

Starts or restarts timer with specified time interval. Interval is addition of seconds and milliseconds passed as an arguments.

Parameters:

in	*seconds*	UI_32 - Time in seconds
in	*msecs*	UI_64 - Time in milliseconds

Return values:

E_CWORD33_Status	- Returns e_CWORD33_StatusOK on success otherwise e_CWORD33_StatusFail

Preconditons

Create a timer handle of instance by SetNotifyMethod()

Change of internal status

None

Classification

Public

Type

None

See also:

SetTime(UI_32, UI_64), Start, Stop, SetNotifyMethod

E_CWORD33_Status NSTimer::Start ()

Brief

This function overloads Start(). Starts or restarts timer with interval specified using SetTime() or last interval (if the timer was started before using Start(seconds, msecs)).

Return values:

E_CWORD33_Status	- Returns e_CWORD33_StatusOK on success otherwise e_CWORD33_StatusFail

Preconditons

Create a timer handle of instance by SetNotifyMethod()

Change of internal status

None

Classification

Public

Type

None

See also:

SetTime(UI_32, UI_64), Stop, SetNotifyMethod, WholeSeconds, MSToNS, NS_TimerSetTime

E_CWORD33_Status NSTimer::Stop ()

Brief

Stops the timer

Return values:

E_CWORD33_Status	- Returns e_CWORD33_StatusOK on success otherwise e_CWORD33_StatusFail

Preconditons

Create a timer handle of instance by SetNotifyMethod()

Change of internal status

None

Classification

Public

Type

None

See also:

Start, Start(UI_32, UI_64), SetNotifyMethod, NS_TimerSetTime

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

111 ns_timer_if.hpp

RaiseExceptionPolicy< RsrcTraits > Class Template Reference

Public Types

typedef RsrcTraits::Type **Type**

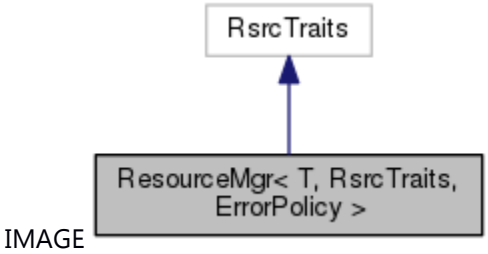
Static Public Member Functions

static Type **check** (Type t)

The documentation for this class was generated from the following file:
112 ns_utility.hpp

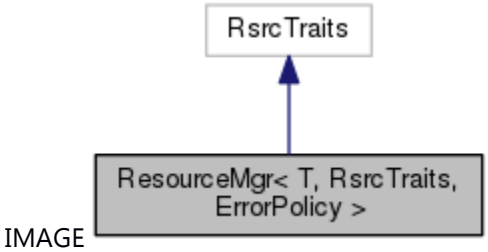
ResourceMgr< T, RsrcTraits, ErrorPolicy > Class Template Reference

Inheritance diagram for ResourceMgr< T, RsrcTraits, ErrorPolicy >:



IMAGE

Collaboration diagram for ResourceMgr< T, RsrcTraits, ErrorPolicy >:



IMAGE

Public Types

typedef RsrcTraits::Type **Type**
typedef ErrorPolicy< RsrcTraits > **TFullErrorPolicy**

Public Member Functions

ResourceMgr (Type rsrc)
operator Type ()
operator const Type () const
bool **isValid** () const

The documentation for this class was generated from the following file:
113 ns_utility.hpp

TList< T, N > Struct Template Reference

Public Member Functions

T * **operator&** ()
const T * **operator&** () const

Public Attributes

T **datum**
TList< T, N-1 > **data**

Static Public Attributes

static const UI_32 **count** = N

The documentation for this struct was generated from the following file:
114 ns_utility.hpp

TList< T, 1 > Struct Template Reference

Public Member Functions

T * **operator&** ()
const T * **operator&** () const

Public Attributes

T **datum**

Static Public Attributes

static const UI_32 **count** = ***

The documentation for this struct was generated from the following file:
115 ns_utility.hpp

XMLParser Class Reference

Public Member Functions

XMLParser ()
~XMLParser ()
void readAllXml (const std::string &xmlInputFile, std::list< std::string > &xmlList)
void getChildNodeList (xmlNodePtr node, std::vector< xmlNodePtr > &nodeVector)
void fillMacroVector (xmlDocPtr &xmlFileHandle, std::vector< xmlNodePtr > &macroNodeVector, std::string &expOutValue, std::list< std::string > &expOutputVector, std::string &macroSecDelayVal, std::string &macroMSecDelayVal, std::string &macroUSecDelayVal, std::vector< std::string > &paramMacroVector)
void createTestCaseVector (std::vector< xmlNodePtr > &testCaseNodeVector, xmlDocPtr &xmlFileHandle, std::vector< CTestCaseData > &testCaseDataVector)
std::string GetXmlNodeAttributeValueToString (void *arg, const char *str, xmlDocPtr xmlFileHandle)
void populateTestCaseMap (std::vector< CTestCaseData > &testCaseDataVector, xmlDocPtr xmlFileHandle)
void updateXmlActualOutput (const std::string &xmlName, std::string &testCaseName, std::string &actualOutput)
void GetXmlExpectedOutput (std::string pKey, std::string pValue)

Static Public Member Functions

static std::string **xmlCharToString** (const xmlChar *value)
static int StringToNumber (const std::string &Text)

Constructor & Destructor Documentation

XMLParser::XMLParser ()

XMLParser Class Constructor

XMLParser::~XMLParser ()

XMLParser Class Destructor

Member Function Documentation

void XMLParser::createTestCaseVector (std::vector< xmlNodePtr > & *testCaseNodeVector*, xmlDocPtr & *xmlFileHandle*, std::vector< CTestCaseData > & *testCaseDataVector*)

createTestCaseVector

Parameters:

in	*testCaseNodeVector*	vector<xmlNodePtr> xmlFileHandle xmlDocPtr testCaseDataVector vector<CTestCaseData>

Returns:

none

void XMLParser::fillMacroVector (xmlDocPtr & *xmlFileHandle*, std::vector< xmlNodePtr > & *macroNodeVector*, std::string & *expOutValue*, std::list< std::string > & *expOutputVector*, std::string & *macroSecDelayVal*, std::string & *macroMSecDelayVal*, std::string & *macroUSecDelayVal*, std::vector< std::string > & *paramMacroVector*)

fillMacroVector

Parameters:

in	*node*	xmlFileHandle xmlDocPtr macroNodeVector vector<xmlNodePtr> expOutValue string expOutputVector list<std::string> macroDelayVal string paramMacroVector vector<std::string>

Returns:

none

void XMLParser::getChildNodeList (xmlNodePtr *node*, std::vector< xmlNodePtr > & *nodeVector*)

getChildNodeList

Parameters:

in	*node*	xmlNodePtr nodeVector vector<xmlNodePtr>

Returns:

none

void XMLParser::GetXmlExpectedOutput (std::string *pKey*, std::string *pValue*)

GetXmlExpectedOutput

Parameters:

in	*pKey*	string pValue string

Returns:

none

std::string XMLParser::GetXmlNodeAttributeValueToString (void * *arg*, const char * *str*, xmlDocPtr *xmlFileHandle*)

GetXmlNodeAttributeValueToString

Parameters:

in	*arg*	void * str const char * xmlFileHandle xmlDocPtr

Returns:

none

void XMLParser::populateTestCaseMap (std::vector< CTestCaseData > & *testCaseDataVector*, xmlDocPtr *xmlFileHandle*)

populateTestCaseMap

Parameters:

in	*testCaseDataVector*	vector<CTestCaseData> xmlFileHandle xmlDocPtr

Returns:

none

void XMLParser::readAllXml (const std::string & *xmlInputFile*, std::list< std::string > & *xmlList*)

readAllXml read all XML files from xmlInputFile

Parameters:

in	*xmlInputFile*	string xmlList string

Returns:

none

static int XMLParser::StringToNumber (const std::string & *Text*)[inline], [static]

StringToNumber Convert the string to number

Parameters:

in	*Text*	string

Returns:

number value

void XMLParser::updateXmlActualOutput (const std::string & *xmlName*, std::string & *testCaseName*, std::string & *actualOutput*)

updateXmlActualOutput

Parameters:

in	*xmlName*	string testCaseName string

Returns:

none

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

116 XMLParser.h

File Documentation

_CWORD78_log_if.h File Reference

This file contains defines and _CWORD33_LOG_Freeze API for free _CWORD33_log.

Macros

#define **_CWORD33_LOG_SHARED_MEM_NAME** "_CWORD33_logconfig.cfg"
#define
_CWORD33_LOG_RAMDISC_NAME "/ramd/log/_CWORD33_log/_CWORD33__debug.log"
#define **_CWORD33_LOG_RAMDISC_PATH** "/ramd/log/_CWORD33_log"
#define **_CWORD33_LOG_FLAG_MODE_DEBUG** 1
#define **_CWORD33_LOG_FLAG_MODE_RELEASE** 0
#define **_CWORD33_LOG_REALTIMELOG_DISABLE_MASK** 0x80
#define **_CWORD33_LOG_REALTIMELOG_MODE_FREEZE** 0xFF
#define **_CWORD33_LOG_REALTIMELOG_MODE_OFF** 0
#define **_CWORD33_LOG_REALTIMELOG_MODE_UART** 1
#define **_CWORD33_LOG_REALTIMELOG_MODE_USB** 2
#define
_CWORD33_LOG_REALTIMELOG_MODE_USB_DISABLE (_CWORD33_LOG_REALTIMELOG_
MODE_USB | _CWORD33_LOG_REALTIMELOG_DISABLE_MASK)
#define **_CWORD33_LOG_REALTIMELOG_MODE_ETHER** 3
#define **_CWORD33_LOG_REALTIMELOG_MODE_MAX** 4

Functions

int _CWORD33_LOG_Freeze (void)

Detailed Description

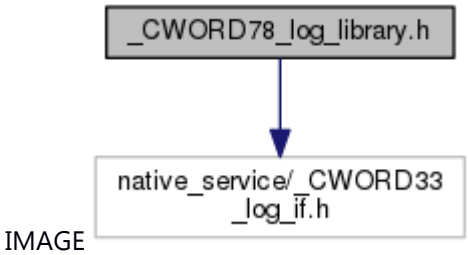
This file contains defines and _CWORD33_LOG_Freeze API for free _CWORD33_log.

_CWORD78_log_library.h File Reference

include all _CWORD33_log_library head files

#include <native_service/_CWORD33_log_if.h>

Include dependency graph for _CWORD78_log_library.h:



IMAGE

Detailed Description

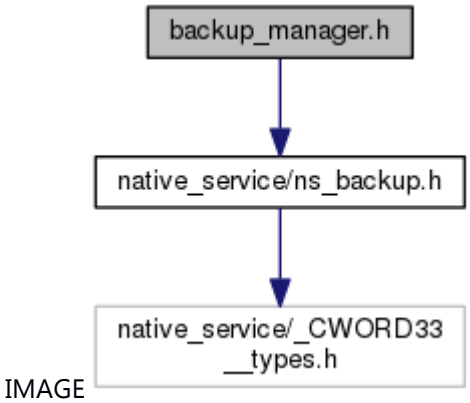
include all _CWORD33_log_library head files

backup_manager.h File Reference

backup_manager unit common header file.

#include <native_service/ns_backup.h>

Include dependency graph for backup_manager.h:



IMAGE

Detailed Description

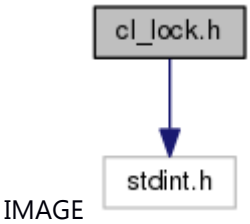
backup_manager unit common header file.

cl_lock.h File Reference

This file contains the base api of cl_clock.

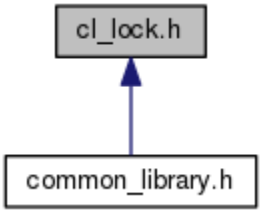
#include <stdint.h>

Include dependency graph for cl_lock.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

int32_t CL_LockSystemInit (void)
int32_t CL_LockProcessInit (void)
void * CL_LockMap (int32_t lid)
int32_t CL_LockUnmap (void *addr)
int32_t CL_LockGet (void *addr)
int32_t CL_LockNowait (void *addr)
int CL_LockRelease (void *addr)

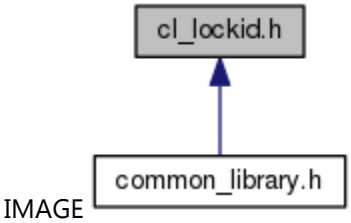
Detailed Description

This file contains the base api of cl_clock.

cl_lockid.h File Reference

LockID(LID)

This graph shows which files directly or indirectly include this file:



IMAGE

Enumerations

enum **ENUM_LOCK_ID** { **LOCK_ANA_LOG_SEM** = ***, **LOCK_POS_MTX_1**, **LOCK_POS_MTX_2**, **LOCK_POS_MTX_3**, **LOCK_POS_MTX_4**, **LOCK_POS_MTX_5**, **LOCK_POS_MTX_6**, **LOCK_POS_MTX_7**, **LOCK_POS_MTX_8**, **LOCK_POS_MTX_9**, **LOCK_POS_MTX_10**, **LOCK_POS_MTX_11**, **LOCK_POS_MTX_12**, **LOCK_POS_MTX_13**, **LOCK_POS_MTX_14**, **LOCK_POS_MTX_15**, **LOCK_POS_MTX_16**, **LOCK_POS_MTX_17**, **LOCK_POS_MTX_18**, **LOCK_POS_MTX_19**, **LOCK_POS_MTX_20**, **LOCK_POS_MTX_21**, **LOCK_POS_MTX_22**, **LOCK_POS_MTX_23**, **LOCK_POS_MTX_24**, **LOCK_POS_MTX_25**, **LOCK_POS_MTX_26**, **LOCK_POS_MTX_27**, **LOCK_POS_MTX_28**, **LOCK_POS_MTX_29**, **LOCK_POS_MTX_30**, **LOCK_POS_MTX_31**, **LOCK_POS_MTX_32**, **LOCK_CLOCK_SEM_TZ**, **LOCK_SEC_DATA_PROTECT**, **LOCK_OS_SEM_ID__CWORD68_**, **LOCK_ROM_ACCESS_IF**, **LOCK_BOOT_ACCESS_IF**, **LOCK_RAM_ACCESS_IF**, **LOCK_HRDS_1**, **LOCK_HRDS_2**, **LOCK_HRDS_3**, **LOCK_HRDS_4**, **LOCK_HRDS_5**, **LOCK_HRDS_6**, **LOCK_HRDS_7**, **LOCK_HRDS_8**, **LOCK_LOGG_ACCESS_IF**, **LOCK__CWORD76__HMI**, **LOCK__CWORD76___CWORD58_**, **LOCK_DIAGCODE_MEM**, **LOCK_MOUNTER_MOUNT**, **LOCK_NOR_ERCNT**, **LOCK_RS_WNG_HIS_LIST_DATA**, **LOCK_RS_WNG_SCREEN_DATA**, **LOCK_RS_SCREEN_DATA**, **LOCK_NSLOG_ACCES_IF_1**, **LOCK_NSLOG_ACCES_IF_2**, **LOCK_NSLOG_ACCES_IF_3**, **LOCK_NSLOG_ACCES_IF_4**, **LOCK_NSLOG_ACCES_IF_5**, **LOCK_NSLOG_ACCES_IF_6**, **LOCK_NSLOG_ACCES_IF_7**, **LOCK_NSLOG_ACCES_IF_8**, **LOCK_NSLOG_ACCES_IF_9**, **LOCK_NSLOG_ACCES_IF_10**, **LOCK_NSLOG_ACCES_IF_11**, **LOCK_NSLOG_ACCES_IF_12**, **LOCK_NSLOG_ACCES_IF_13**, **LOCK_NSLOG_ACCES_IF_14**, **LOCK_NSLOG_ACCES_IF_15**, **LOCK_NSLOG_ACCES_IF_16**, **LOCK_NSLOG_ACCES_IF_17**, **LOCK_NSLOG_ACCES_IF_18**, **LOCK_NSLOG_ACCES_IF_19**, **LOCK_NSLOG_ACCES_IF_20**, **LOCK_NSLOG_ACCES_IF_21**, **LOCK_NSLOG_ACCES_IF_22**, **LOCK_NSLOG_ACCES_IF_23**, **LOCK_NSLOG_ACCES_IF_24**, **LOCK_NSLOG_ACCES_IF_25**, **LOCK_NSLOG_ACCES_IF_26**, **LOCK_NSLOG_ACCES_IF_27**, **LOCK_NSLOG_ACCES_IF_28**, **LOCK_NSLOG_ACCES_IF_29**, **LOCK_NSLOG_ACCES_IF_30**, **LOCK_NSLOG_ACCES_IF_31**, **LOCK_NSLOG_ACCES_IF_32**, **LOCK_NSLOG_ACCES_IF_33**, **LOCK_NSLOG_ACCES_IF_34**, **LOCK_NSLOG_ACCES_IF_35**, **LOCK_NSLOG_ACCES_IF_36**, **LOCK_NSLOG_ACCES_IF_37**, **LOCK_NSLOG_ACCES_IF_38**, **LOCK_NSLOG_ACCES_IF_39**, **LOCK_NSLOG_ACCES_IF_40**, **LOCK_NSLOG_ACCES_IF_41**, **LOCK_NSLOG_ACCES_IF_42**, **LOCK_NSLOG_ACCES_IF_43**, **LOCK_INFOSETTING_REV**, **LOCK_SUBMENU_SELECT**, **LOCK_DIAG_SEM1**, **LOCK_DIAG_SEM2**, **LID_NUM** }

Detailed Description

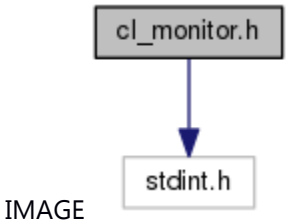
LockID(LID)

cl_monitor.h File Reference

This file contains the enum,struct and api of cl_monitor.

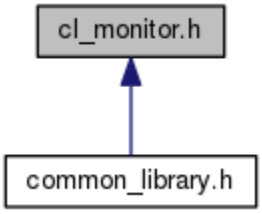
#include <stdint.h>

Include dependency graph for cl_monitor.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct CL_MonitorEntry_t
struct CL_MonitorSearch_t

Enumerations

enum CL_MonitorInit_t { **CL_MONITOR_INIT_SYSTEM** = ***, **CL_MONITOR_INIT_USER** }
enum CL_MonitorType_t { **CL_MONITOR_TYPE_GENERIC** = ***, **CL_MONITOR_TYPE_RPC** }
enum CL_MonitorState_t { **CL_MONITOR_STATE_SLEEP** = ***, **CL_MONITOR_STATE_RUN** }

Functions

int CL_MonitorInit (CL_MonitorInit_t init_type)
int CL_MonitorSetEntry (CL_MonitorType_t type, uint32_t id, CL_MonitorState_t state, uint32_t timeout, uint32_t user_data)
int CL_MonitorGetEntry (CL_MonitorType_t type, uint32_t id, CL_MonitorEntry_t *entry)
int CL_MonitorSearchInit (CL_MonitorSearch_t *serch)
int CL_MonitorSearchDestroy (CL_MonitorSearch_t *serch)
int CL_MonitorSearchTimeout (CL_MonitorSearch_t *search)

Detailed Description

This file contains the enum,struct and api of cl_monitor.

Class Documentation

struct CL_MonitorEntry_t

Brief

the struct of Monitor Entry

Class Members:

uint32_t	id	
uint16_t	pid	
uint8_t	state	
time_t	timeout	
uint8_t	type	
uint32_t	user_data	

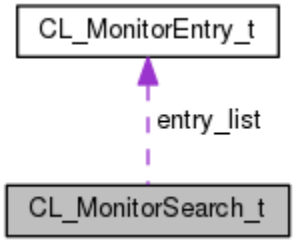
struct CL_MonitorSearch_t

Brief

the struct of Monitor Search

Collaboration diagram for CL_MonitorSearch_t:

IMAGE



Class Members:

CL_MonitorEntry_t *	entry_list	
int	entry_num	

Enumeration Type Documentation

enum CL_MonitorInit_t

Brief

Monitor init enum

enum CL_MonitorState_t

Brief

Monitor state enum

enum CL_MonitorType_t

Brief

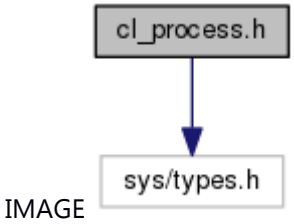
Monitor type enum

cl_process.h File Reference

This file contains the base api,enum and struct of cl_process.

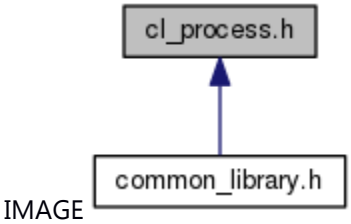
#include <sys/types.h>

Include dependency graph for cl_process.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct CL_ProcessAttr_t
struct CL_ProcessCleanupInfo_t
struct CL_ThreadAttr_t
struct CL_ProcessCreateCgroupAttr_t

Macros

#define **_cl_process_h_**
#define **CL_PROCESSS_ATTR_HOLD_FDS_NUM** 8
#define **CL_INTFY_FILENAME_FORMAT** "/tmp/intfy_%05d"

Enumerations

enum CL_ProcessSchedPolicy_t { CL_PROCESS_SCHED_POLICY_OTHER = ***,
CL_PROCESS_SCHED_POLICY_RR, CL_PROCESS_SCHED_POLICY_FIFO }

Functions

int CL_ProcessInit (void)
int CL_ProcessCreate (const char *file, char *const argv[], char *const envp[], const CL_ProcessAttr_t *attr)
int CL_ProcessCreateAttrInit (CL_ProcessAttr_t *attr)
int CL_ProcessCreateAttrSetName (CL_ProcessAttr_t *attr, const char *name)
int CL_ProcessCreateAttrSetUid (CL_ProcessAttr_t *attr, uid_t uid)

int CL_ProcessCreateAttrSetGid (CL_ProcessAttr_t *attr, gid_t gid)
int CL_ProcessCreateAttrSetSchedule (CL_ProcessAttr_t *attr, CL_ProcessSchedPolicy_t policy, int priority)
int CL_ProcessCreateAttrSetGroup (CL_ProcessAttr_t *attr, int create)
int **CL_ProcessCreateAttrSetCpuAssign** (CL_ProcessAttr_t *attr, int cpu_assign)
int CL_ProcessCreateAttrSetStackSize (CL_ProcessAttr_t *attr, int stack_size)
int CL_ProcessCreateAttrSetHoldFds (CL_ProcessAttr_t *attr, int hold_fds[])
int CL_ProcessCreateAttrSetDisableCloseFds (CL_ProcessAttr_t *attr)
int CL_ProcessCreateAttrSetCgroup (CL_ProcessAttr_t *attr, const char *cgroup_name)
int CL_ProcessTerminate (pid_t pid)
int CL_ProcessTerminateGroup (pid_t pid)
int **CL_ProcessAbort** (pid_t pid)
int **CL_ProcessAbortGroup** (pid_t pid)
int **CL_ProcessEuthanizeGroup** (pid_t pid)
int CL_ProcessCleanup (int sigchld_fd, CL_ProcessCleanupInfo_t *cleanup_info)
int **CL_ThreadCreate** (pthread_t *thread, pthread_attr_t *attr, CL_ThreadAttr_t *cl_attr, void *(*start_routine)(void *), void *arg)
int **CL_ThreadCreateAttrInit** (CL_ThreadAttr_t *attr)
int **CL_ThreadCreateAttrSetName** (CL_ThreadAttr_t *attr, const char *name)
int CL_ProcessCreateCgroupCreate (const char *cgroup_name, CL_ProcessCreateCgroupAttr_t *attr)
int CL_ProcessCreateCgroupAttrInit (CL_ProcessCreateCgroupAttr_t *attr)
int CL_ProcessCreateCgroupAttrSetRtThrottling (CL_ProcessCreateCgroupAttr_t *attr, int runtime_us)
int CL_ProcessCreateCgroupAttrSetCpuShares (CL_ProcessCreateCgroupAttr_t *attr, int cpu_shares)
int CL_ProcessCreateCgroupAttrSetCfsBandwidthControl (CL_ProcessCreateCgroupAttr_t *attr, int cfs_quota_us)
int CL_ProcessCreateCgroupAttrSetMemoryLimit (CL_ProcessCreateCgroupAttr_t *attr, int memory_limit)
int CL_ProcessCreateCgroupAttrSetMemoryUsageNotification (CL_ProcessCreateCgroupAttr_t *attr, int usage_in_bytes, int event_fd)
int CL_ProcessCreateCgroupDelete (const char *cgroup_name)
int CL_ProcessCreateCgroupClassify (const char *cgroup_name, pid_t pid)

Detailed Description

This file contains the base api,enum and struct of cl_process.

Class Documentation

struct CL_ProcessAttr_t

Brief

Class Members:

char	body[148]	

struct CL_ProcessCleanupInfo_t

Brief

Class Members:

int	code	
pid_t	pid	ID
int	status	

struct CL_ThreadAttr_t

Class Members:

char	body[20]	

struct CL_ProcessCreateCgroupAttr_t

Cgroup

Class Members:

char	body[24]	Cgroup

Enumeration Type Documentation

enum CL_ProcessSchedPolicy_t

Brief

Enumerator

CL_PROCESS_SCHED_POLICY_OTHER TSS.

CL_PROCESS_SCHED_POLICY_RR

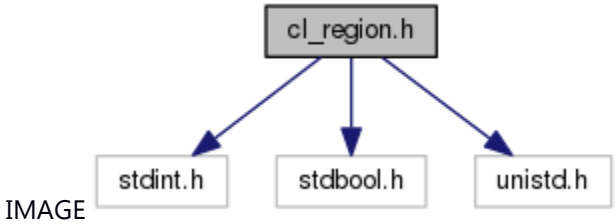
CL_PROCESS_SCHED_POLICY_FIFO FIFO.

cl_region.h File Reference

region manage

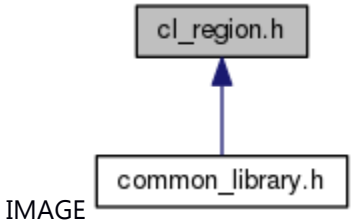
#include <stdint.h>
#include <stdbool.h>
#include <unistd.h>

Include dependency graph for cl_region.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct cl_region_cleanup_s
struct cl_region_large_s
struct cl_region_data_t
struct cl_region_s

Macros

#define **_CL_REGION_H_**
#define **CL_REGION_DEFAULT_SIZE** (4 * 1024)
#define **CL_ALIGNOF**(type) __alignof(type)
#define CL_RegionAlloc(region, type, length) cl_region_alloc(region, sizeof(type) * length, CL_ALIGNOF(type))
#define CL_RegionCleanupAdd(region, type, length) cl_region_cleanup_add(region, sizeof(type) * length, CL_ALIGNOF(type))

Typedefs

typedef void(* **cl_region_cleanup_pt**) (void *data)
typedef struct cl_region_cleanup_s **cl_region_cleanup_t**
typedef struct cl_region_large_s **cl_region_large_t**
typedef struct cl_region_s **cl_region_t**

Functions

cl_region_t * CL_RegionCreate (size_t size)
void CL_RegionDestroy (cl_region_t *region)
void * **cl_region_alloc** (cl_region_t *region, size_t size, size_t align_size)
bool CL_RegionFree (cl_region_t *region, void *p)
cl_region_cleanup_t * **cl_region_cleanup_add** (cl_region_t *region, size_t size, size_t align_size)

Detailed Description

region manage

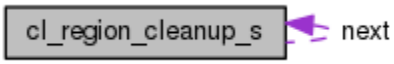
Class Documentation

struct cl_region_cleanup_s

Brief

region cleanup struct

Collaboration diagram for cl_region_cleanup_s:

IMAGE 

Class Members:

void *	data	
cl_region_cleanup_pt	handler	
cl_region_cleanup_t *	next	

struct cl_region_large_s

Brief

region large struct

Collaboration diagram for cl_region_large_s:

IMAGE 

Class Members:

void *	alloc	
cl_region_large_t *	next	
size_t	size	

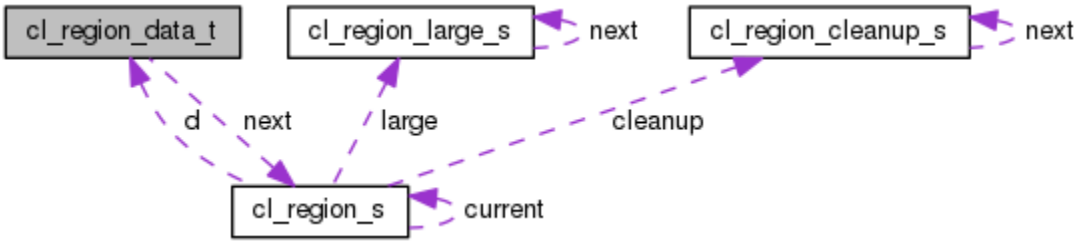
struct cl_region_data_t

Brief

region data struct

Collaboration diagram for cl_region_data_t:

IMAGE



Class Members:

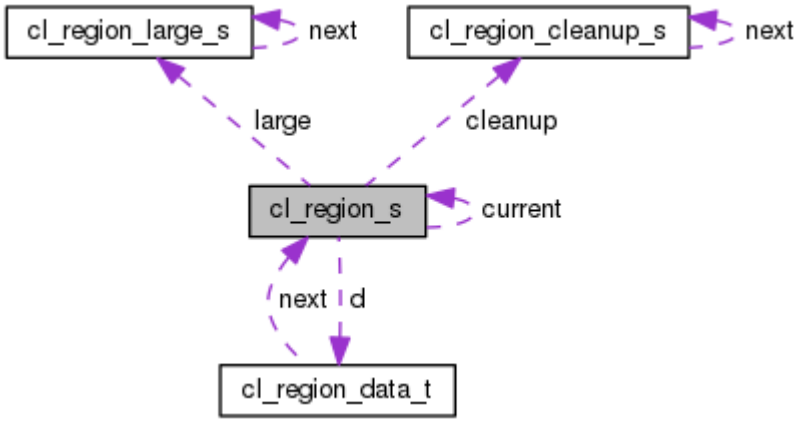
uint8_t *	end	
int	failed	
uint8_t *	last	
cl_region_t *	next	

struct cl_region_s

Brief

region struct

Collaboration diagram for cl_region_s:



IMAGE

Class Members:

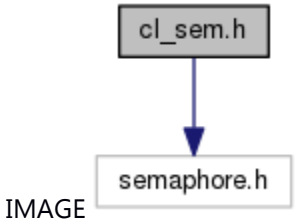
cl_region_cleanup_t *	cleanup	
cl_region_t *	current	
cl_region_data_t	d	
cl_region_large_t *	large	
size_t	max	

cl_sem.h File Reference

This file contains the api of sem wait.

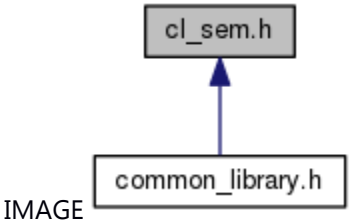
#include <semaphore.h>

Include dependency graph for cl_sem.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

EXT_C int CL_SemWait (sem_t *semid, unsigned int timeout)

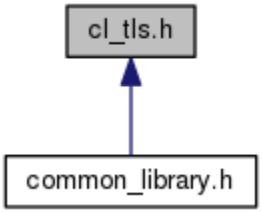
Detailed Description

This file contains the api of sem wait.

cl_tls.h File Reference

This file contains the base api for operatoring CL_TlsData_t.

This graph shows which files directly or indirectly include this file:

IMAGE 

Classes

struct CL_TlsData_t

Functions

void CL_TlsInit (int pno)
CL_TlsData_t * CL_TlsGet (void)
int CL_TlsGetRcvFD (void)
void CL_TlsSetRcvFD (int fd)
int CL_TlsThreadSeqID (void)

Detailed Description

This file contains the base api for operatoring CL_TlsData_t.

Class Documentation

struct CL_TlsData_t

Class Members:

int	id	
int	pno	
int	rcv_fd	

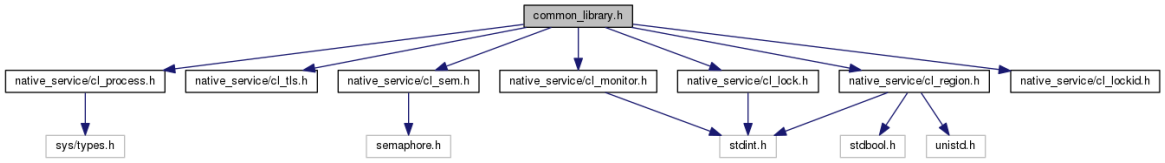
common_library.h File Reference

include all common_library head files

```
#include <native_service/cl_process.h>
#include <native_service/cl_tls.h>
#include <native_service/cl_sem.h>
#include <native_service/cl_monitor.h>
#include <native_service/cl_lock.h>
#include <native_service/cl_region.h>
#include <native_service/cl_lockid.h>
```

Include dependency graph for common_library.h:

IMAGE



Detailed Description

include all common_library head files

framework_unified.h File Reference

include all framework_unified head files

```
#include <native_service/ns_config_parser_if.h>
#include <native_service/ns_eventlogger.h>
#include <native_service/ns_logger_if.h>
#include <native_service/ns_message_center_if.h>
#include <native_service/ns_np_service.h>
#include <native_service/ns_np_service_if.h>
#include <native_service/ns_np_service_nor_persistence.h>
#include <native_service/ns_np_service_notification.h>
#include <native_service/ns_np_service_protocol.h>
#include <native_service/ns_plogger_if.h>
#include <native_service/ns_shared_mem.h>
#include <native_service/ns_shared_mem_if.h>
#include <native_service/ns_sharedmem.h>
#include <native_service/ns_sharedmem_reader.h>
#include <native_service/ns_sharedmem_writer.h>
#include <native_service/ns_ringbuffer.h>
#include <native_service/ns_system_mode.h>
#include <native_service/ns_timer_if.h>
#include <native_service/ns_timer_if.hpp>
#include <native_service/ns_transmit_log.h>
#include <native_service/ns_utility.hpp>
#include <native_service/ns_utility_if.h>
#include <native_service/ns_utility_sys.hpp>
#include <native_service/ns_util_directory.h>
#include <native_service/ns_util_crc.h>
#include <native_service/ns_version_if.h>
#include <native_service/ns_resource_controler.h>
#include <native_service/ns_version.h>
#include <native_service/_CWORD33__service_protocol.h>
#include <native_service/_CWORD33__types.h>
#include <native_service/wpf_legacy.h>
#include <native_service/ns_rcs_data_handler.hpp>
#include <native_service/ns_rcs_logger_types.h>
#include <native_service/ns_rcs_plugin.hpp>
#include <native_service/ns__CWORD77__common.h>
#include <native_service/ns__CWORD77__data_pool_if.h>
#include <native_service/ns__CWORD77__data_pool_key.h>
#include <native_service/ns__CWORD77__data_pool_table.h>
#include <native_service/ns__CWORD77__notifications.h>
```

```
#include <native_service/ns__CWORD77__types.h>
#include <native_service/_CWORD33___CWORD77__service_if.h>
#include <native_service/_CWORD33___CWORD77__session_if.h>
#include <native_service/ns_buildversioncheck.hpp>
#include <native_service/ns_endianess.h>
#include <native_service/nslogutil_cmd_if.h>
#include <native_service/nsfw_message.h>
#include <native_service/_CWORD33__application.h>
#include <native_service/_CWORD33__dispatcher.h>
#include <native_service/_CWORD33__framework_if.h>
#include <native_service/_CWORD33__framework_sync.h>
#include <native_service/_CWORD33__framework_system_if.h>
#include <native_service/_CWORD33__framework_types.h>
#include <native_service/_CWORD33__multithreading.h>
#include <native_service/_CWORD33__service_if.h>
#include <native_service/_CWORD33__thread_priority.h>
#include <native_service/_CWORD33__timer.h>
#include <native_service/_CWORD33__sm_action.h>
#include <native_service/_CWORD33__sm_compositestate.h>
#include <native_service/_CWORD33__sm_conditionconnector.h>
#include <native_service/_CWORD33__sm_deephistorystate.h>
#include <native_service/_CWORD33__sm_dispatcher.h>
#include <native_service/_CWORD33__sm_eventdata.h>
#include <native_service/_CWORD33__sm_eventfactory.h>
#include <native_service/_CWORD33__sm_externaltransition.h>
#include <native_service/_CWORD33__sm_framework_dispatch.h>
#include <native_service/_CWORD33__sm_framework_if.h>
#include <native_service/_CWORD33__sm_framework_types.h>
#include <native_service/_CWORD33__sm_guard.h>
#include <native_service/_CWORD33__sm_historystate.h>
#include <native_service/_CWORD33__sm_hsm.h>
#include <native_service/_CWORD33__sm_hsmframework.h>
#include <native_service/_CWORD33__sm_internaltransition.h>
#include <native_service/_CWORD33__sm_leafstate.h>
#include <native_service/_CWORD33__sm_localtransition.h>
#include <native_service/_CWORD33__sm_multithreading.h>
#include <native_service/_CWORD33__sm_orthogonalstate.h>
#include <native_service/_CWORD33__sm_reaction.h>
#include <native_service/_CWORD33__sm_shallowhistorystate.h>
#include <native_service/_CWORD33__sm_state.h>
#include <native_service/_CWORD33__sm_transition.h>
#include <native_service/XMLParser.h>
#include <native_service/ns_xml_reader.h>
#include <native_service/ns_xml_wrter.h>
```

```
#include <native_service/ns_xmlparser_if.h>
#include <native_service/ns_reader.h>
#include <native_service/ns_writer.h>
```

Detailed Description

include all framework_unified head files

notification_persistent_service.h File Reference

include all notification_persistent_service head files

Detailed Description

include all notification_persistent_service head files

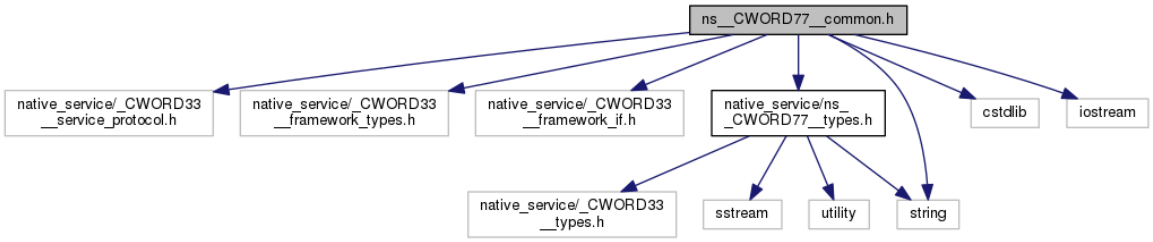
ns__CWORD77__common.h File Reference

This file has templet class declaration of CCallback and CResCallback.

#include <native_service/_CWORD33__service_protocol.h>
#include <native_service/_CWORD33__framework_types.h>
#include <native_service/_CWORD33__framework_if.h>
#include <native_service/ns__CWORD77__types.h>
#include <cstdlib>
#include <iostream>
#include <string>

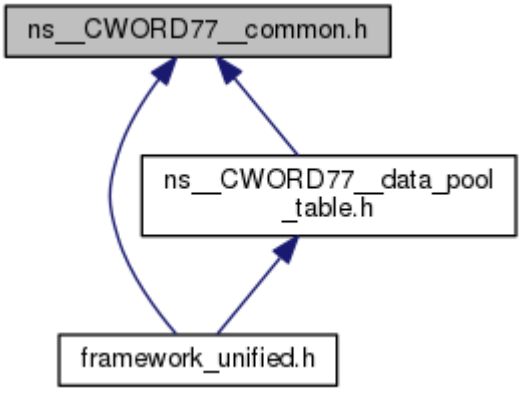
Include dependency graph for ns__CWORD77__common.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

class CCallback< C, M >
class CResCallback< C, M >

Macros

#define **MAX_DATA_SIZE** 512;
#define **DELETEPTR**(p) {if (p != NULL) {delete p; p = NULL;}}

Typedefs

typedef E_CWORD33_Status(* **_CWORD77_FuncPtr**) (HANDLE hApp, BOOL bTImerExpiry)

typedef E_CWORD33_Status(* **ResponseServiceTo_CWORD77_**) (HANDLE hApp,
E_CWORD77_ResponseType ResponseType, std::string ServiceName)
typedef boost::function< E_CWORD33_Status(HANDLE, std::string, std::string, UI_32,
E_CWORD33_Status) > **SessionAckTo_CWORD77_**

Detailed Description

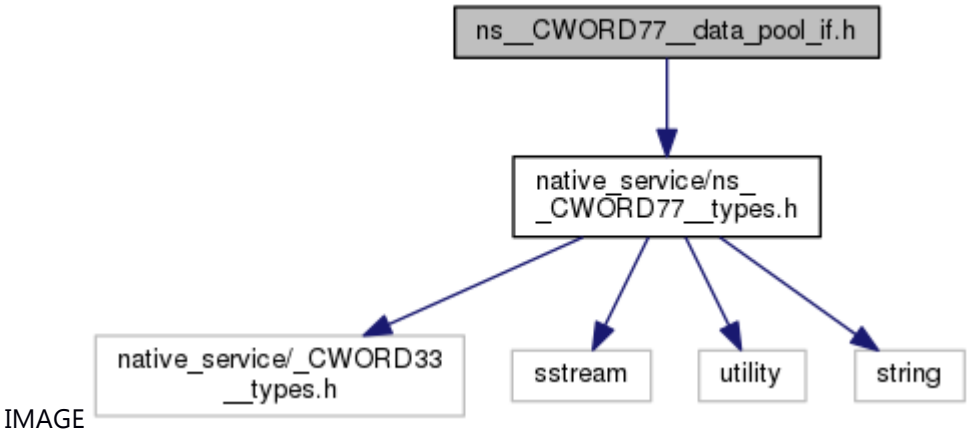
This file has templet class declaration of CCallback and CResCallback.

ns__CWORD77__data_pool_if.h File Reference

This file provides api to operating _CWORD77_DataPool.

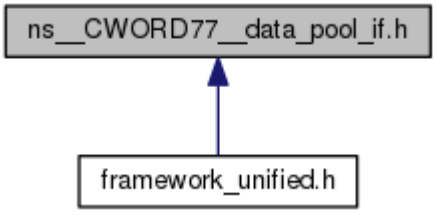
#include <native_service/ns__CWORD77__types.h>

Include dependency graph for ns__CWORD77__data_pool_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

VOID SetReqDataIn_CWORD77_DataPool (const _CWORD77_DataPoolKey &f_DataPoolKey, UI_32 f_uiSize, PVOID f_pData)

E_CWORD33_Status GetRespDataFrom_CWORD77_DataPool (const _CWORD77_DataPoolKey &f_DataPoolKey, UI_32 &f_uiSize, PVOID &f_pData)

VOID SetRespNotfnDataIn_CWORD77_DataPool (const std::string &f_cNotificationName, UI_32 f_uiSize, const PVOID f_pData)

E_CWORD33_Status GetRespNotfnDataFrom_CWORD77_DataPool (const std::string &f_cNotificationName, UI_32 &f_uiSize, PVOID &f_pData)

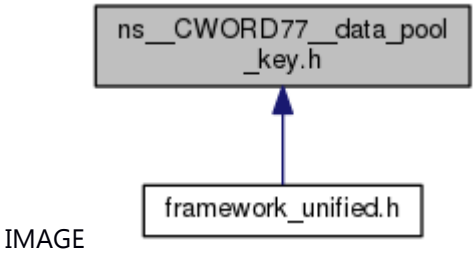
Detailed Description

This file provides api to operating _CWORD77_DataPool.

ns__CWORD77__data_pool_key.h File Reference

Provide function _CWORD77__DATA_POOL_KEY of make_pair.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define _CWORD77__DATA_POOL_KEY(ProtocolId) (std::make_pair(ProtocolId, g_c_CWORD77_ServiceName))

Detailed Description

Provide function _CWORD77__DATA_POOL_KEY of make_pair.

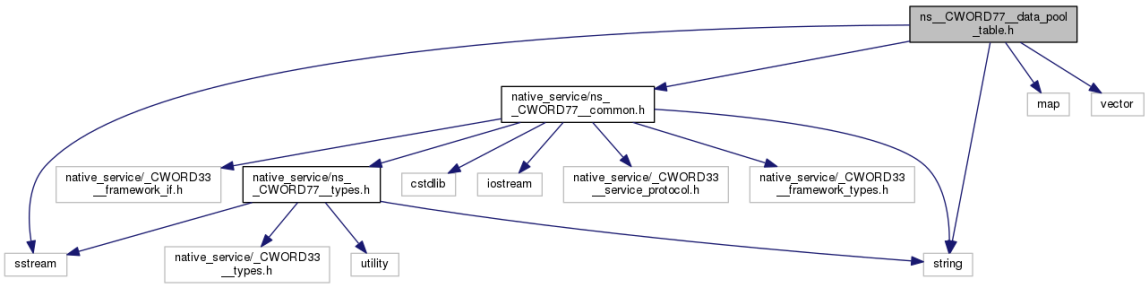
ns__CWORD77__data_pool_table.h File Reference

This file provides API for convert and C_CWORD77_DataPool class.

#include <native_service/ns__CWORD77__common.h>
#include <sstream>
#include <map>
#include <string>
#include <vector>

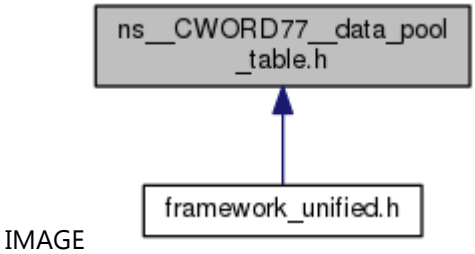
Include dependency graph for ns__CWORD77__data_pool_table.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

struct __CWORD77_Data

CWORD77 Data structure *More...*

class C_CWORD77_DataPool

class C_CWORD77_Data

enum _ETableType { **REQUEST_TABLE** = ***, **RESPONSE_TABLE** }*Enumeration for Type of Table.*

enum _EDataType { **UNKNOWNTYPE** = ***, **SIGNEDINT**, **UNSIGNEDINT**, **FLOAT**, **DOUBLE**, **CHARACTER**, **BOOLEAN**, **STRING** }*Enumeration for Data Type.*

typedef enum _ETableType ETableType

Enumeration for Type of Table.

typedef enum _ETableType * **PETableType**

typedef enum _EDataType EDataType

Enumeration for Data Type.

typedef enum _EDataType * **PEDataType**
typedef struct __CWORD77_Data _CWORD77_Data

CWORD77 Data structure

typedef struct __CWORD77_Data * **P_CWORD77_Data**
typedef std::map< _CWORD77_DataPoolKey, std::vector< CHAR > > **T_CWORD77_DataPool**
C_CWORD77_DataPool **g__CWORD77_DataPool**
UI_32 NoOfElementsinData (std::string Input)
template<typename T > std::string NumberToString (T Number)

Template to Convert Number to String.

template<typename T > T StringToNumber (const std::string &Text)

Template to convert String to Number.

template<class T > std::string ConvertArrayToString (T *Array, int Arraysize)

Template to Convert an Array To String.

template<class T > void ConvertStringToArray (std::string Input, std::vector< T > &Array, UI_32 &Arraysize)

Template to Convert String To Array.

std::string ConvertArrayStringsToString (std::string *strArr, SI_32 Size)
void ConvertStringToArrString (std::string Input, std::string *strArr, UI_32 &ArraySize)
template<class T > void SetRequestArrayData (UI_32 VarName, EDataType DataType, T *Array, UI_32 ArraySize)
template<class T > void SetResponseArrayData (UI_32 VarName, EDataType DataType, T *Array, UI_32 ArraySize)
template<class T > void GetRequestArrayData (UI_32 VarName, T *Array, UI_32 &ArraySize)
template<class T > void GetResponseArrayData (UI_32 VarName, T *Array, UI_32 &ArraySize)
template<class T > void SetRequestData (UI_32 VarName, EDataType DataType, T Array)
template<class T > void SetResponseData (UI_32 VarName, EDataType DataType, T Array)
template<class T > T GetRequestData (UI_32 VarName)
template<class T > T GetResponseData (UI_32 VarName)
void SetRequestArrayStringData (UI_32 VarName, EDataType VarType, std::string DataValue[], UI_32 size)
void SetResponseArrayStringData (UI_32 VarName, EDataType VarType, std::string DataValue[], UI_32 size)
void GetRequestArrayStringData (UI_32 VarName, std::string DataValue[], UI_32 &size)
void GetResponseArrayStringData (UI_32 VarName, std::string DataValue[], UI_32 &size)
void SetRequestStringData (UI_32 VarName, EDataType VarType, std::string DataValue)
void SetResponseStringData (UI_32 VarName, EDataType VarType, std::string DataValue)
std::string GetRequestStringData (UI_32 VarName)
UI_32 GetRequestDataLength (UI_32 Key)
UI_32 GetResponseDataLength (UI_32 Key)
EDataType GetRequestDataType (UI_32 Key)
EDataType GetResponseDataType (UI_32 Key)

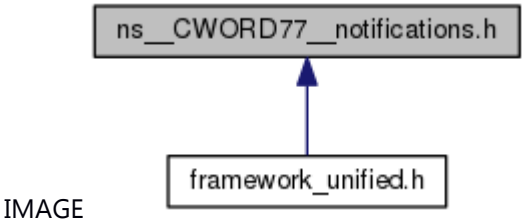
Detailed Description

This file provides API for convert and C_CWORD77_DataPool class.

ns__CWORD77__notifications.h File Reference

This file defines *CWORD77* Service Availability Notification.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define NTFY_REG__CWORD77__Available "_CWORD77_/Available"

CWORD77 Service Availability Notification

Detailed Description

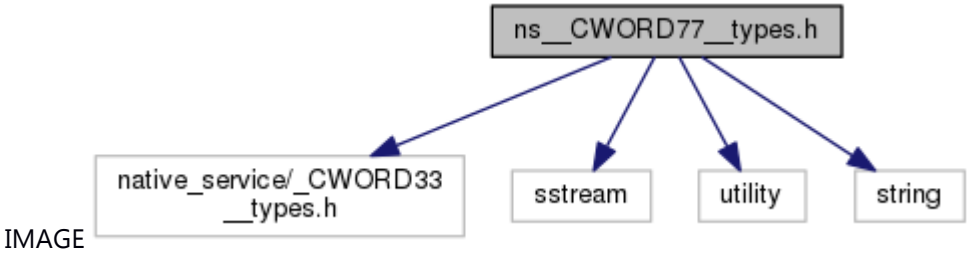
This file defines *CWORD77* Service Availability Notification.

ns__CWORD77__types.h File Reference

This file contains enum of PE_CWORD77_ResponseType.

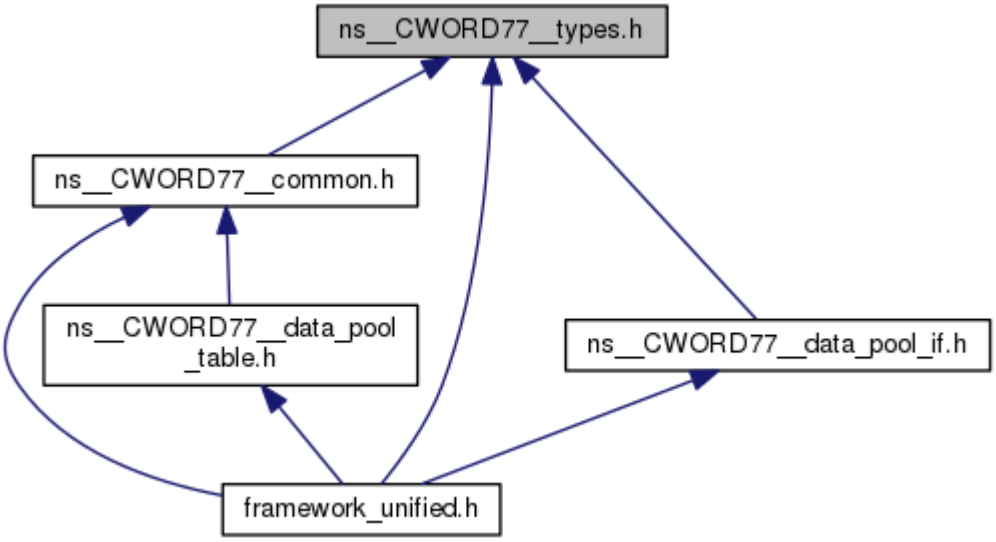
#include <native_service/_CWORD33__types.h>
#include <sstream>
#include <utility>
#include <string>

Include dependency graph for ns__CWORD77__types.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

enum **_E_CWORD77_ResponseType** { **RESPONSE** = ***, **NOTIFICATION**, **EVENT** }
typedef enum _E_CWORD77_ResponseType **E_CWORD77_ResponseType**
typedef enum _E_CWORD77_ResponseType * **PE_CWORD77_ResponseType**
typedef std::pair< UI_32, std::string > **_CWORD77_DataPoolKey**

Detailed Description

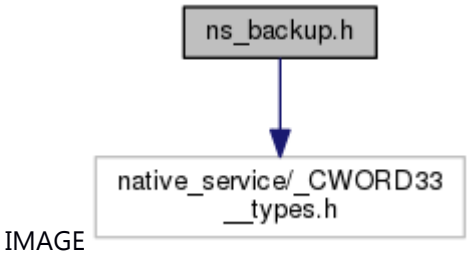
This file contains enum of PE_CWORD77_ResponseType.

ns_backup.h File Reference

backup_manager function header file

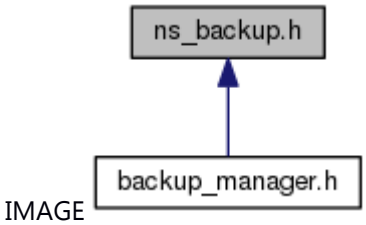
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_backup.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define NTFY_BackupMgr_Availability "NS_BackupMgr/Availability"
#define BKUP_RET_NORMAL 0
#define BKUP_RET_ERROR -1
#define BKUP_RET_ERRPARAM -2
#define BKUP_RET_ERRINIT -3
#define BKUP_RET_ERRTERM -4
#define BKUP_RET_ERRNOENT -5
#define BKUP_RET_ERRSIZE -6

Functions

int32_t Backup_DataRd (PCSTR tag_id, uint32_t ui_offset, void *pv_buf, uint32_t ui_size)
int32_t Backup_DataWt (PCSTR tag_id, void *pv_buf, uint32_t ui_offset, uint32_t ui_size)
int32_t Backup_DataFil (PCSTR tag_id, uint32_t ui_offset, uint8_t uc_pat, uint32_t ui_size)
int32_t Backup_DataSz (PCSTR tag_id, uint32_t *pui_size)
int32_t Backup_DataRdByNumID (uint32_t num_id, uint32_t ui_offset, void *pv_buf, uint32_t ui_size)
int32_t Backup_DataSzByNumID (uint32_t num_id, uint32_t *pui_size)
int32_t Backup_DataChk (PCSTR tag_id)
int32_t Backup_DataDel (PCSTR tag_id)

Detailed Description

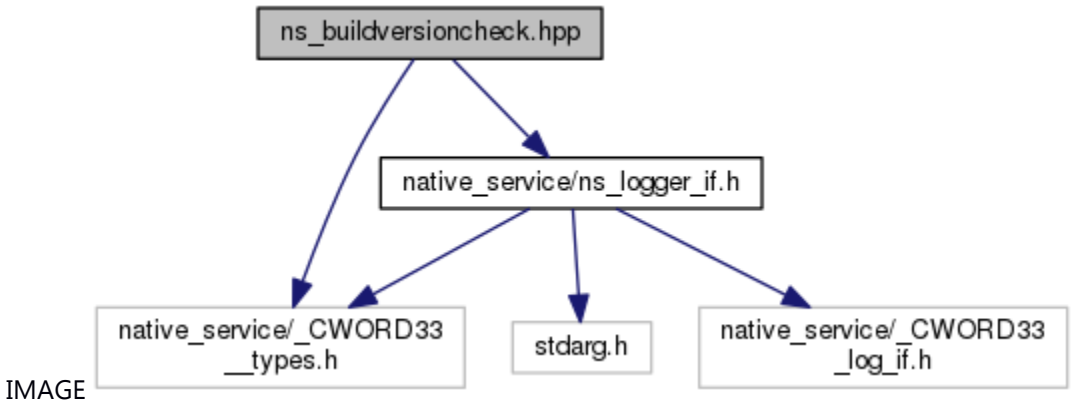
backup_manager function header file

ns_buildversioncheck.hpp File Reference

Provide the APIs to check/get version.

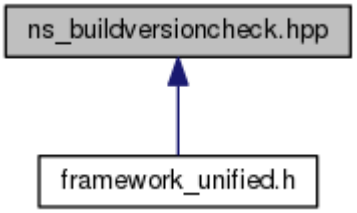
#include <native_service/_CWORD33__types.h>
#include <native_service/ns_logger_if.h>

Include dependency graph for ns_buildversioncheck.hpp:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Namespaces

CWORD33
CWORD33::ns
CWORD33::ns::utility
CWORD33::ns::utility::utility_private

Macros

#define _BUILD_VERSION_ ""
#define WARNING_STRING "\n\!!! WARNING - Build Versions do not match !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n\project build: %s\n\library build: %s\n\environment build: %s\n\!!!"
#define BUILDVERSIONLOG(___log_string___, ...) TEXT(__PRETTY_FUNCTION__, ___log_string___, __VA_ARGS__)
#define BUILDVERSION_NULL_STATEMENT ((void)0)
#define CHECK_BUILD_VERSION()

Functions

BOOL _CWORD33_::ns::utility::buildVersionsMatch (PCSTR build_version)
PCSTR _CWORD33_::ns::utility::getEnvironmentBuildVersion ()
PCSTR _CWORD33_::ns::utility::getLibraryBuildVersion ()
PCSTR _CWORD33_::ns::utility::utility_private::niceBuildVersion (PCSTR build_version)

Detailed Description

Provide the APIs to check/get version.

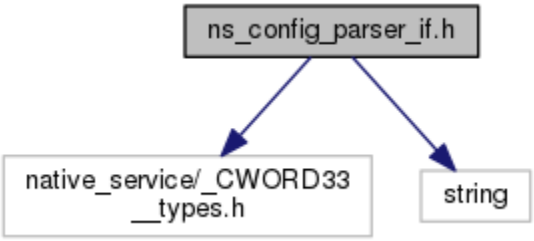
ns_config_parser_if.h File Reference

This file contains declaration of class CNSConfigReader, CNSConfigWriter and CNSConfigParser. API define head file.

#include <native_service/_CWORD33__types.h>
#include <string>

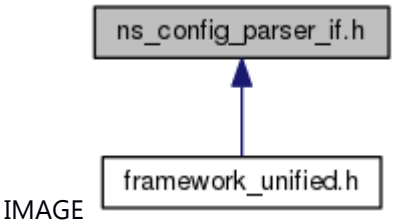
Include dependency graph for ns_config_parser_if.h:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

class CNSConfigReader

Configuration File Read Service Class. class CNSConfigWriter

File Write Service Class. class CNSConfigParser

File both Reade and Write Service Class.

Detailed Description

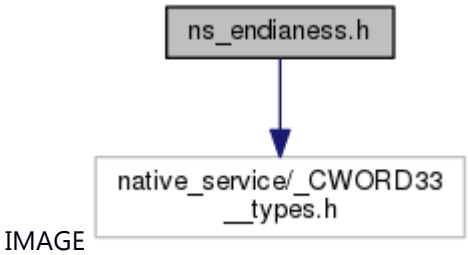
This file contains declaration of class CNSConfigReader, CNSConfigWriter and CNSConfigParser. API define head file.

ns_endianess.h File Reference

Provide api for convert value.

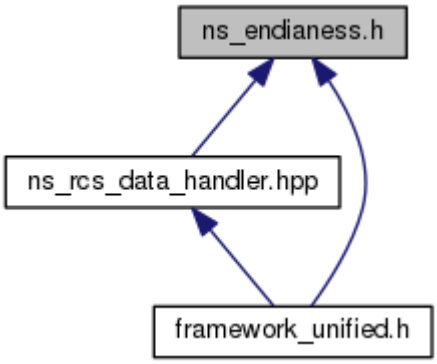
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_endianess.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

UI_16 ConvertEndian_UI16 (UI_16 f_value)
SI_16 ConvertEndian_SI16 (SI_16 f_value)
UI_32 ConvertEndian_UI32 (UI_32 f_value)
SI_32 ConvertEndian_SI32 (SI_32 f_value)
UI_64 ConvertEndian_UI64 (UI_64 f_value)
SI_64 ConvertEndian_SI64 (SI_64 f_value)

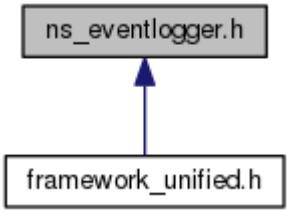
Detailed Description

Provide api for convert value.

ns_eventlogger.h File Reference

structures used by ns_logger.cpp for event and counter logging

This graph shows which files directly or indirectly include this file:

IMAGE 

Classes

struct _stLogEvent_
struct _stLogCount_

Macros

#define **EVENTLOG_MSGQ_NAME** "/EvntLogQue"

Typedefs

typedef enum _Datatype_ **Datatype**
typedef enum _SystemPhase_ **SystemPhase**
typedef enum _NStoSS_LOGGERSERVICEPROTOCOL **NStoSS_loggerserviceprotocol**
typedef struct _stLogEvent_ **st_LogEvent**
typedef struct _stLogCount_ **st_LogCount**

Enumerations

enum **_Datatype_** { **COMMON**, **EVENT_SPECIFIC** }
enum **_SystemPhase_** { **STARTUP** = ***, **NORMAL** = ***, **SHUTDOWN** = *** }
enum **_NStoSS_LOGGERSERVICEPROTOCOL** { **SS_MSG_EVTLOG** = ***, **SS_MSG_LOGGERCNT**, **SS_MSG_LOGGER_CNT_EVTLOG** }

Detailed Description

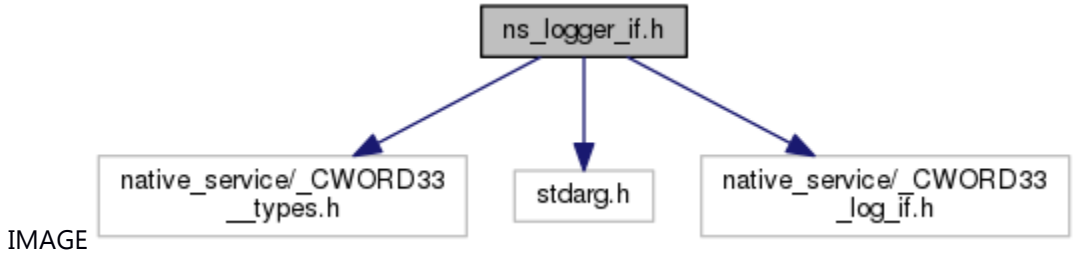
structures used by ns_logger.cpp for event and counter logging

ns_logger_if.h File Reference

Declares the external APIs to Logger.

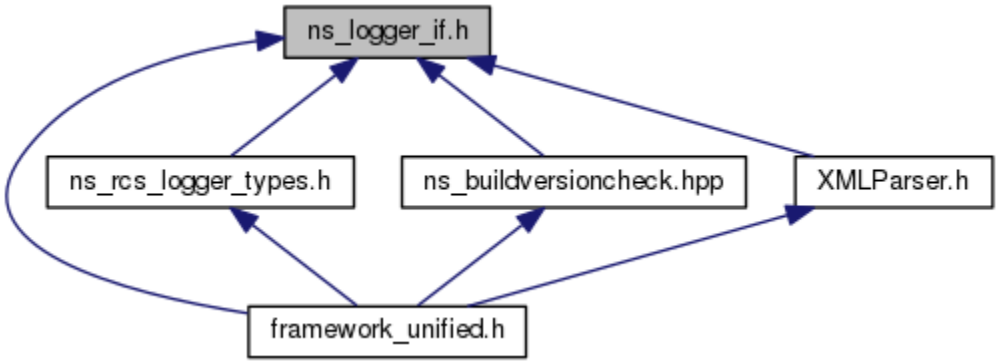
#include <native_service/_CWORD33__types.h>
#include <stdarg.h>
#include <native_service/_CWORD33_log_if.h>

Include dependency graph for ns_logger_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct __CWORD33_LOGPARAM
_CWORD33_LOG parameter *More...*
struct tag_Change_Log_Parameters
struct _TNS_EnableRcsLogging

Macros

#define **_CWORD33_LOG_MSGQ_NAME** "/NSLog"
#define **_CWORD33_LOG_SHAREDMEM_NAME** "/_CWORD33__debug.log.1"
#define **_CWORD33_LOG_SHAREDMEM_SIZE** 5242880
#define **MASTER_ZONE_COUNT** 16U
#define **BITS_IN_BYTE** (8U)
#define **BITS_IN_TZONE** ((UI_32)(sizeof(T_CWORD33_Zone) * BITS_IN_BYTE))
#define **TZONE_COUNT** ((UI_32)MASTER_ZONE_COUNT)
#define **BITS_IN_ZONE_MASK** ((UI_32)(BITS_IN_TZONE * TZONE_COUNT))

#define **DEBUG_ZONE_COUNT** BITS_IN_ZONE_MASK
#define **ZONE_MASK_ARRAY_ELTS** ((UI_32)TZONE_COUNT)
#define **USER_ZONES_COUNT** BITS_IN_ZONE_MASK -10
#define **ZONE_TEXT_SIZE** 24
#define **NS_RCS_LOGGER_PLUGIN_Q** "/NsRcsLoggerPlugin"
#define _CWORD33_LOGZONES {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

extern _CWORD33_LOG parameter

#define _CWORD33__SET_ZONES()
#define **ZONE_END** UINT_MAX
#define **GET_ZONE_INDEX**(zone) (((UI_32)zone) >> 5)
#define **GET_ZONE_BIT_MASK**(zone) ((T_CWORD33_Zone)(1U << (((UI_32)zone) & 0x1F)))
#define ZONEMASK(n) ((T_CWORD33_Zone)(n))
#define IS_ZONE_SET(set_zone) (NsLogIsZoneSet(set_zone) == TRUE)
#define _CWORD33_LOG(zone, funcname, print_fmt, args...) (zone != ZONEMASK(11))?((IS_ZONE_SET(zone))?TEXT_ZONE(zone, funcname, print_fmt, ## args):((void)0)):((void)(0))
#define _**CWORD33_LOG_TIME**(zone, str) ((IS_ZONE_SET(zone))?NsLogTime(zone, str):((void)(0)))
#define _**CWORD33_LOG_DATA**(zone, data, size) ((IS_ZONE_SET(zone))?NsLogData(zone, data, size):((void)(0)))
#define _CWORD33_LOG_EVTCNT(zone, CntId, EvtId, n...) NsLog_EvtCnt(CntId, EvtId, ##n)

_CWORD33_LOG_EVTCNT

#define _CWORD33_LOG_EVT(zone, EvtId, n...) NsLog_Evt(EvtId, ##n)

_CWORD33_LOG_EVT

#define _CWORD33_LOG_CNT(zone, CntId, n...) NsLog_Cnt(CntId, ##n)

_CWORD33_LOG_CNT

#define TEXT(funcname, args...) TEXT_ZONE(BITS_IN_ZONE_MASK, funcname, ## args)

Deprecated API. Not use.

#define **GET_MACRO**(_1, _2, _3, _4, _5, _6, _7, _8, _9, _10, _11, _12, _13, _14, _15, _16, _17, _18, _19, _20, _21, _22, _23, _24, _25, _26, _27, _28, _29, _30, _31, _32, _33, _34, _35, _36, _37, _38, _39, _40, _41, _42, _43, _44, _45, _46, _47, _48, _49, _50, _51, _52, _53, _54, _55, _56, _57, _58, _59, _60, _61, _62, _63, _64, _65, _66, _67, _68, _69, _70, _71, _72, _73, _74, _75, _76, _77, _78, NAME, ...) NAME
#define **NsLogFunc**(...)
#define **TEXT_ZONE**(zone_index, funcname, ...) NsLogFunc(__VA_ARGS__)(__LINE__, zone_index, funcname, __VA_ARGS__)
#define TEXT_EVTCNT(CntId, EvtId, n...) NsLog_EvtCnt(CntId, EvtId, ##n)

Deprecated API. Not use.

#define TEXT_EVT(EvtID, n...) NsLog_Evt(EvtID, ##n)

Deprecated API. Not use.

#define TEXT_CNT(CntId, n...) NsLog_Cnt(CntId, ##n)

Deprecated API. Not use.

#define _CWORD33_LOG0 _CWORD33_LOG

Deprecated API. Use _CWORD33_LOG instead.

#define ZONE_NS_FUNC ZONEMASK(0)

Note: These bits are reserved for Framework logging and debugging.

#define **ZONE_NS_DIS** ZONEMASK(1)
#define **ZONE_NS__CWORD77_** ZONEMASK(2)
#define **ZONE_NS_IMP_INFO** ZONEMASK(3)
#define **ZONE_NS_SM_DEV_INFO** ZONEMASK(4)
#define **ZONE_NS_SM_USR_INFO** ZONEMASK(5)
#define **ZONE_NS_INFO** ZONEMASK(6)
#define **ZONE_NS_DEBUG_DUMP** ZONEMASK(7)
#define **ZONE_NS_WAR** ZONEMASK(8)
#define **ZONE_NS_ERR** ZONEMASK(9)
#define **ZONE_DEBUG_DUMP** ZONE_NS_DEBUG_DUMP
#define **ZONE_TEXT_0** "NS_Func"
#define **ZONE_TEXT_1** "NS_Dispatcher"
#define **ZONE_TEXT_2** "NS__CWORD77_"
#define **ZONE_TEXT_3** "NS_Reserved"
#define **ZONE_TEXT_4** "NS_StateMachine_Dev"
#define **ZONE_TEXT_5** "NS_StateMachine_Usr"
#define **ZONE_TEXT_6** "NS_Info"
#define **ZONE_TEXT_7** "NS_Debug_Dump"
#define **ZONE_TEXT_8** "NS_Warning"
#define **ZONE_TEXT_9** "NS_Error"
#define ZONE_PERFORMANCE ZONEMASK(64)

Note: These bits are reserved for special ZONE.

#define **ZONE_SCREEN_TRANS** ZONEMASK(65)
#define **ZONE_SWDL** ZONEMASK(66)
#define **ZONE_SYSTEMDATA** ZONEMASK(67)
#define **ZONE_LOG_SH** ZONEMASK(68)
#define **ZONE_LOG_SYS** ZONEMASK(69)
#define **ZONE_LOG__CWORD52_** ZONEMASK(70)
#define **ZONE_SECURE** ZONEMASK(71)
#define **ZONE_CAN_FILTER** ZONEMASK(72)
#define **ZONE_COMM_PROCESS** ZONEMASK(73)
#define **ZONE_CONNECT_DEVICE** ZONEMASK(74)
#define **ZONE_SP_ZONE_75** ZONEMASK(75)
#define **ZONE_SP_ZONE_76** ZONEMASK(76)
#define **ZONE_SP_ZONE_77** ZONEMASK(77)
#define **ZONE_SP_ZONE_78** ZONEMASK(78)
#define **ZONE_SP_ZONE_79** ZONEMASK(79)
#define **ZONE_SP_ZONE_80** ZONEMASK(80)
#define **ZONE_SP_ZONE_81** ZONEMASK(81)
#define **ZONE_SP_ZONE_82** ZONEMASK(82)
#define **ZONE_SP_ZONE_83** ZONEMASK(83)
#define **ZONE_SP_ZONE_84** ZONEMASK(84)
#define **ZONE_SP_ZONE_85** ZONEMASK(85)
#define **ZONE_SP_ZONE_86** ZONEMASK(86)
#define **ZONE_SP_ZONE_87** ZONEMASK(87)

#define **ZONE_SP_ZONE_88** ZONEMASK(88)
#define **ZONE_SP_ZONE_89** ZONEMASK(89)
#define **ZONE_SP_ZONE_90** ZONEMASK(90)
#define **ZONE_SP_ZONE_91** ZONEMASK(91)
#define **ZONE_SP_ZONE_92** ZONEMASK(92)
#define **ZONE_SP_ZONE_93** ZONEMASK(93)
#define **ZONE_SP_ZONE_94** ZONEMASK(94)
#define **ZONE_SP_ZONE_95** ZONEMASK(95)
#define ZONE_COMM_SYS ZONEMASK(96)

Note: These bits are reserved for special ZONE.

#define **ZONE_COMM_SH** ZONEMASK(97)
#define **ZONE_COMM_USB** ZONEMASK(98)
#define **ZONE_SP_ZONE_99** ZONEMASK(99)
#define **ZONE_SP_ZONE_100** ZONEMASK(100)
#define **ZONE_SP_ZONE_101** ZONEMASK(101)
#define **ZONE_SP_ZONE_102** ZONEMASK(102)
#define **ZONE_SP_ZONE_103** ZONEMASK(103)
#define **ZONE_SP_ZONE_104** ZONEMASK(104)
#define **ZONE_SP_ZONE_105** ZONEMASK(105)
#define **ZONE_SP_ZONE_106** ZONEMASK(106)
#define **ZONE_SP_ZONE_107** ZONEMASK(107)
#define **ZONE_SP_ZONE_108** ZONEMASK(108)
#define **ZONE_SP_ZONE_109** ZONEMASK(109)
#define **ZONE_SP_ZONE_110** ZONEMASK(110)
#define **ZONE_SP_ZONE_111** ZONEMASK(111)
#define **ZONE_SP_ZONE_112** ZONEMASK(112)
#define **ZONE_SP_ZONE_113** ZONEMASK(113)
#define **ZONE_SP_ZONE_114** ZONEMASK(114)
#define **ZONE_SP_ZONE_115** ZONEMASK(115)
#define **ZONE_SP_ZONE_116** ZONEMASK(116)
#define **ZONE_SP_ZONE_117** ZONEMASK(117)
#define **ZONE_SP_ZONE_118** ZONEMASK(118)
#define **ZONE_SP_ZONE_119** ZONEMASK(119)
#define **ZONE_SP_ZONE_120** ZONEMASK(120)
#define **ZONE_SP_ZONE_121** ZONEMASK(121)
#define **ZONE_SP_ZONE_122** ZONEMASK(122)
#define **ZONE_SP_ZONE_123** ZONEMASK(123)
#define **ZONE_SP_ZONE_124** ZONEMASK(124)
#define **ZONE_SP_ZONE_125** ZONEMASK(125)
#define **ZONE_SP_ZONE_126** ZONEMASK(126)
#define **ZONE_SP_ZONE_127** ZONEMASK(127)

Typedefs

typedef UI_32 T_CWORD33_Zone

typedef of ZONE

typedef T_CWORD33_Zone T_CWORD33_ZoneMask[ZONE_MASK_ARRAY_ELTS]

Array of mask options.

typedef enum _T_CWORD33_LoggerSeverity **T_CWORD33_LoggerSeverity**
typedef struct __CWORD33_LOGPARAM _CWORD33_LOGPARAM
_CWORD33_LOG parameter
typedef struct __CWORD33_LOGPARAM * **LP_CWORD33_LOGPARAM**
typedef enum _NS_LOGGER_METHODS NS_LoggingMethod_t
_CWORD33_LOG output direction
typedef struct tag_Change_Log_Parameters **CHANGELOGPARAMS**
typedef enum _NS_RCS_LOGGER_SETTINGS **NS_RCS_LOGGER_SETTINGS**
typedef struct _TNS_EnableRcsLogging **TNS_EnableRcsLogging**

Enumerations

enum _T_CWORD33_LoggerSeverity { **_CWORD33__SEVERITY_LOW** = ***,
_CWORD33__SEVERITY_DEBUG2, _CWORD33__SEVERITY_DEBUG1,
_CWORD33__SEVERITY_INFO, _CWORD33__SEVERITY_WARN, _CWORD33__SEVERITY_ERROR,
_CWORD33__SEVERITY_FATAL, _CWORD33__SEVERITY_ALWAYS }
enum _NS_LOGGER_METHODS { **LPRINT** = ***, LMSGQ = ***, LSLOGGER = ***,
LSHAREDMEM = *** }*_CWORD33_LOG output direction*
enum **_NS_RCS_LOGGER_SETTINGS** { **NS_RCS_LOGGER_PLUGIN_ADD_APPNAME** = *** }

Functions

void NsLog (const UI_16 p_lLine_i, const UI_16 f_uiZoneIndex, PCSTR p_pstrClassName_i, PCSTR lpszFormat,...)
void NsLog0 (const UI_16 p_lLine_i, const UI_16 f_uiZoneIndex, PCSTR p_pstrClassName_i, PCSTR lpszFormat)
void NsLogTime (const UI_16 f_uiZoneIndex, PCSTR lpszFormat)
void NsLogData (const UI_16 f_uiZoneIndex, PCSTR data, UI_32 size)
void NsLogSet_CWORD33_LogParams (_CWORD33_LOGPARAM *p__CWORD33_LogParams)
void NsLogSetProcessName (PCSTR p_strProcessName_i)
void NsLogSetControlMask (T_CWORD33_ZoneMask p_ulNSLogControl_i)
BOOL NsLogIsZoneSet (UI_32 set_zone)
void NsLogGetControlMask (T_CWORD33_ZoneMask p_Zonemask_i)
void NsLogSetLogMethod (UI_8 p_eMethod_i)
UI_8 NsLogGetLogMethod (void)
void NsLogInitialize (void)
void NsLogSet_CWORD33_LogFlag (UI_8 flag_id, UI_8 mode)
E_CWORD33_Status NsLogGet_CWORD33_LogFlag (UI_8 flag_id, UI_8 *mode)
void NsLogSetRealtimeLog (UI_8 mode)
void NsLogGetRealtimeLog (UI_8 *mode)
void NsLogSetSeverity (T_CWORD33_LoggerSeverity p_eLogSeverity_i)
T_CWORD33_LoggerSeverity NsLogGetSeverity (void)
void NsLog_EvtCnt (UI_16 Cnt_Id, UI_16 Evt_Id, UI_8 n,...)
void NsLog_Evt (UI_16 Evt_Id, UI_8 nu,...)
void NsLog_Cnt (UI_16 Cnt_Id, UI_8 nu,...)
UI_8 NsLogDetermineLogMethod (PCSTR output_type)
VOID NsLogGetZoneTextList (CHAR f_cZoneList[][ZONE_TEXT_SIZE])
void NsLogSetZones (UI_32 f_uiZoneCount,...)

void **NsLogParseZones** (_CWORD33_LOGPARAM *p__CWORD33_LogParams, UI_32 f_uiZoneCount,...)
UI_32 NSLogGet_CWORD33_logFileTotalNum (void)
PCSTR NSLogGet_CWORD33_logFileName (UI_32 index)
int NSLogGet_CWORD33_logIndex (PCSTR filename)
VOID NsForceClose (void)

Detailed Description

Declares the external APIs to Logger.

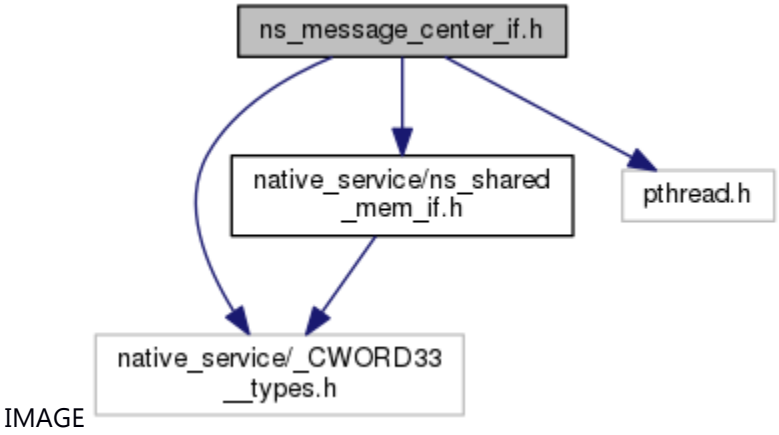
Declares the external APIs to Logger.

ns_message_center_if.h File Reference

APIs to Open/Close and Send/Receive on message queues and shared memory.

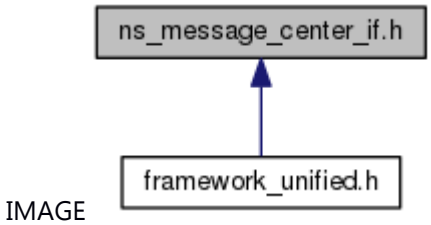
#include <native_service/_CWORD33__types.h>
#include <native_service/ns_shared_mem_if.h>
#include <pthread.h>

Include dependency graph for ns_message_center_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

HANDLE McOpenReceiver (PCSTR name)
HANDLE McOpenReceiverNotBlocked (PCSTR name)
HANDLE McOpenSyncReceiver (PCSTR name)
HANDLE McOpenSender (PCSTR name)
HANDLE McOpenSenderNotBlocked (PCSTR name)
HANDLE McOpenSyncSender (PCSTR name)
HANDLE **McOpenSenderChild** (PCSTR name, pthread_t childid)
E_CWORD33_Status **McJoinChild** (HANDLE hChildApp)
E_CWORD33_Status **McGetChildThreadPriority** (HANDLE hChildApp, PSI_32 threadPrio)
E_CWORD33_Status McReceive (HANDLE hMessage, PSTR source, UI_32 *cmd, UI_32 length, PVOID data)
E_CWORD33_Status McReceiveWithSession (HANDLE hMessage, PSTR source, UI_32 *cmd, UI_32 *sessionid, UI_32 length, PVOID data)
UI_32 McGetLength (PVOID data)

PVOID McGetDataPointer (PVOID data)
E_CWORD33_Status McGetDataOfSize (PVOID data, PVOID to, UI_32 uiSize)
E_CWORD33_Status McGetDataOfSizeWithSMRetain (PVOID data, PVOID to, UI_32 uiSize)
E_CWORD33_Status McGetSysInfoData (PVOID data, PVOID to)
E_CWORD33_Status McClearData (PVOID data)
E_CWORD33_Status McSend (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 length, PCVOID data)
E_CWORD33_Status McSendWithSession (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 length, PCVOID data, UI_32 sessionid)
E_CWORD33_Status McSendWithPriority (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 length, PCVOID data, E_CWORD33_MessagePriorties priority, UI_32 sessionid)
E_CWORD33_Status McInvokeSync (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 msgLength, PCVOID msgData, UI_32 sessionid, HANDLE hRcvMessage, UI_32 responseLength, PVOID responseData, UI_32 *receivedLength)
E_CWORD33_Status McSendSyncResponse (HANDLE hMessage, PCSTR source, UI_32 cmd, UI_32 seq_id, E_CWORD33_Status ret_val, UI_32 length, PCVOID data)
E_CWORD33_Status McCreateInvokerName (PCSTR source, UI_32 sessionid, PSTR invokerName, UI_32 size)
E_CWORD33_Status McClose (HANDLE hMessage)
TMemID McGetDataUSID (PVOID pData)
PCSTR McGetMsgSrc (PVOID data)
E_CWORD33_Status McForward (HANDLE hMessage, PCSTR source, UI_32 iCmd, TMemID USID)
void McFlushReceiver (HANDLE hMessage)
PCSTR McGetQueueName (HANDLE hMessage)
int McGetQueueFD (HANDLE hMessage)
E_CWORD33_Status McTranslateError (int error)
E_CWORD33_Status McZcSetParam (HANDLE handle, UI_32 cmd, UI_32 length)
PVOID McZcGetBuf (HANDLE handle)
E_CWORD33_Status McZcSend (HANDLE hMessage)
HANDLE McZcOpenSender (PCSTR source)
E_CWORD33_Status McZcClose (HANDLE handle)

Detailed Description

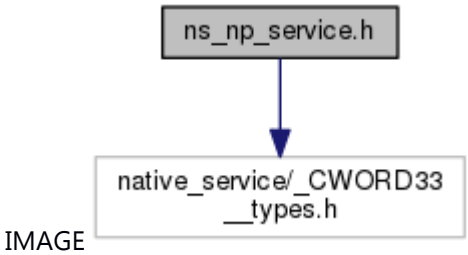
APIs to Open/Close and Send/Receive on message queues and shared memory.

ns_np_service.h File Reference

Header for message structures for notification_persistent_service.

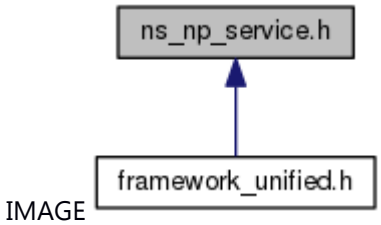
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_np_service.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _NC_register_notif_msg
struct _NC_register_multiple_notif_msg
struct _NC_register_immediate_notif_msg
struct _NC_register_multiple_immediate_notif_msg
struct _NC_unregister_notif_msg
struct _NC_unregister_multiple_notif_msg
struct _NC_savepersdata_ack
struct _NC_subscribe_multiple_notif_msg
struct _NC_unsubscribe_multiple_notif_msg
struct __NC_get_persdata_failed_ack
struct _NC_User
struct _NC_RegisterPersistentFileMsg
struct _NC_LoadPersistedFileMsg
struct _NC_ReleasePersistentFileMsg
struct _NC_RegisterPersistentFolderMsg
struct _NC_LoadPersistedFolderMsg
struct _NC_ReleasePersistentFolderMsg
struct _NC_LoadPersistedAck
struct _NC_NorPersistentData
struct _NC_ClearPersistedDataReq
struct _NC_ClearPersisteDatadAck

struct _NC_StopMsgData
struct _NC_SetPersistType
struct _NC_ImmediateWriteAck

Typedefs

typedef struct _NC_register_notif_msg NC_register_notif_msg
typedef struct _NC_register_multiple_notif_msg NC_register_multiple_notif_msg
typedef struct _NC_register_immediate_notif_msg NC_register_immediate_notif_msg
typedef struct _NC_register_multiple_immediate_notif_msg
NC_register_multiple_immediate_notif_msg
typedef struct _NC_unregister_notif_msg NC_unregister_notif_msg
typedef struct _NC_unregister_notif_msg **NC_subscribe_msg**
typedef struct _NC_unregister_notif_msg **NC_unsubscribe_frm_notif_msg**
typedef struct _NC_unregister_notif_msg **NC_get_pers_data_msg**
typedef struct _NC_unregister_multiple_notif_msg NC_unregister_multiple_notif_msg
typedef struct _NC_savepersdata_ack NC_savepersdata_ack
typedef struct _NC_subscribe_multiple_notif_msg NC_subscribe_multiple_notif_msg
typedef struct _NC_unsubscribe_multiple_notif_msg NC_unsubscribe_multiple_notif_msg
typedef struct __NC_get_persdata_failed_ack NC_get_persdata_failed_ack
typedef struct _NC_User NC_User
typedef struct _NC_RegisterPersistentFileMsg NC_RegisterPersistentFileMsg
typedef struct _NC_LoadPersistedFileMsg NC_LoadPersistedFileMsg
typedef struct _NC_ReleasePersistentFileMsg NC_ReleasePersistentFileMsg
typedef struct _NC_RegisterPersistentFolderMsg NC_RegisterPersistentFolderMsg
typedef struct _NC_LoadPersistedFolderMsg NC_LoadPersistedFolderMsg
typedef struct _NC_ReleasePersistentFolderMsg NC_ReleasePersistentFolderMsg
typedef struct _NC_LoadPersistedAck NC_LoadPersistedAck
typedef struct _NC_LoadPersistedAck **NC_ReleasePersistedAck**
typedef struct _NC_NorPersistentData NC_NorPersistentData
typedef struct _NC_ClearPersistedDataReq NC_ClearPersistedDataReq
typedef struct _NC_ClearPersisteDatadAck NC_ClearPersisteDatadAck
typedef struct _NC_StopMsgData NC_StopMsgData
typedef struct _NC_SetPersistType NC_SetFilePersistType
typedef struct _NC_SetPersistType **NC_SetFolderPersistType**
typedef struct _NC_ImmediateWriteAck NC_ImmediateWriteAck

Detailed Description

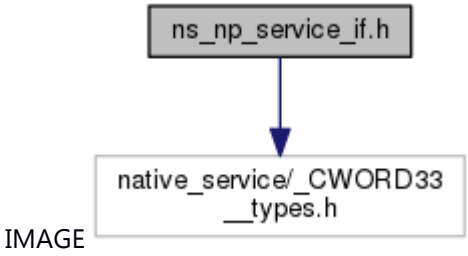
Header for message structures for notification_persistent_service.

ns_np_service_if.h File Reference

Declares the external APIs to Notification and Persistence Service.

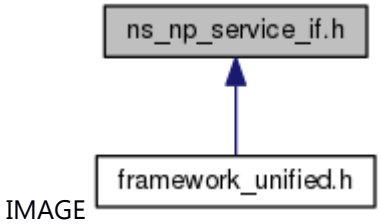
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_np_service_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _NotificationInfo
struct _ImmediateNotificationInfo
struct _SubscribeInfo

Typedefs

typedef struct _NotificationInfo NotificationInfo
typedef struct _ImmediateNotificationInfo ImmediateNotificationInfo
typedef struct _SubscribeInfo SubscribeInfo

Functions

E_CWORD33_Status NPRegisterNotifications (HANDLE hNPMsgQ, PCSTR pPublisherName, UI_32 numNotifications, NotificationInfo *pNotificationArray)
E_CWORD33_Status NPRegisterImmediateNotifications (HANDLE hNPMsgQ, PCSTR pPublisherName, UI_32 numNotifications, ImmediateNotificationInfo *pNotificationArray)
E_CWORD33_Status NPRegisterNotification (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR notif_name, const UI_32 max_length, const E_CWORD33_NotificationType perstype)
E_CWORD33_Status NPUnRegisterNotification (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pNotification)
E_CWORD33_Status NPSetPersistentNotfnType (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pNotification, E_CWORD33_PersistCategory ePersistCategory)

E_CWORD33_Status NPSetPersistNotfnDefaultValue (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pNotification, PCVOID pData, const UI_32 iLength)
E_CWORD33_Status NPUnRegisterNotifications (HANDLE hNPMsgQ, PCSTR pPublisherName, UI_32 numNotifications, NotificationInfo *pNotificationArray)
E_CWORD33_Status NPPublishNotification (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pNotification, PCVOID pData, const UI_32 iLength)
E_CWORD33_Status NPSubscribeToNotification (HANDLE hNPMsgQ, PCSTR pSubscriberName, PCSTR notif_name)
E_CWORD33_Status NPSubscribeToNotifications (HANDLE hNPMsgQ, PCSTR pSubscriberName, UI_32 numNotifications, SubscribeInfo *pSubscribeInfoArray)
E_CWORD33_Status NPUnsubscribeFromNotification (HANDLE hNPMsgQ, PCSTR pSubscriberName, PCSTR pNotification)
E_CWORD33_Status NPUnsubscribeFromNotifications (HANDLE hNPMsgQ, PCSTR pUnSubscriberName, UI_32 numNotifications, SubscribeInfo *pUnSubscribeInfoArray)
E_CWORD33_Status NPReadPersistedData (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR notification)
E_CWORD33_Status NPSavePersistentData (HANDLE hNPMsgQ, PCSTR pPublisherName)
E_CWORD33_Status NPRegisterPersistentFile (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pTag, BOOL bIsUserFile)
E_CWORD33_Status NPSetFilePersistentType (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pTag, E_CWORD33_PersistCategory ePersistCategory)
E_CWORD33_Status NPLoadPersistentFile (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pDstFilePath, PCSTR pTag, HANDLE hUser)
E_CWORD33_Status NPReleasePersistentFile (HANDLE hNPMsgQ, PCSTR pPublisherName, E_CWORD33_ReleaseType eReleaseType, PCSTR pTag, PCSTR pFullFilePath, HANDLE hUser)
E_CWORD33_Status NPPersistentSync (PCSTR SrcName, HANDLE hNPMsgQ, UI_32 sessionid, PCSTR pPublisherName)
E_CWORD33_Status NPSetPersonality (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pUserName)
E_CWORD33_Status NPChangePersonality (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pUserName)
E_CWORD33_Status NPGetReadyStatusOfNPP (HANDLE hNPMsgQ, PCSTR pRequesterName)
E_CWORD33_Status NPClearPersistedData (HANDLE hNPMsgQ, PCSTR pRequesterName, E_CWORD33_ClearPersistence e_CWORD33_ClearPersistenceScope)
E_CWORD33_Status NPRegisterPersistentFolder (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pTag, BOOL bIsUserFolder)
E_CWORD33_Status NPSetFolderPersistentType (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pTag, E_CWORD33_PersistCategory ePersistCategory)
E_CWORD33_Status NPLoadPersistentFolder (HANDLE hNPMsgQ, PCSTR pPublisherName, PCSTR pDstFolderPath, PCSTR pTag, HANDLE hUser)
E_CWORD33_Status NPReleasePersistentFolder (HANDLE hNPMsgQ, PCSTR pPublisherName, E_CWORD33_ReleaseType e_CWORD33_ReleaseType, PCSTR pTag, PCSTR pFullFolderPath, HANDLE hUser)

Detailed Description

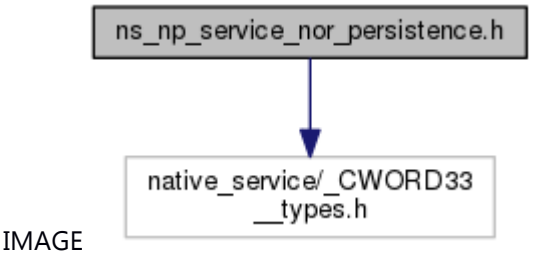
Declares the external APIs to Notification and Persistence Service.

ns_np_service_nor_persistence.h File Reference

This file contains the declaration for methods to retrieve and store data for NOR storage.

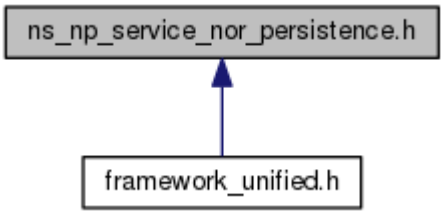
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_np_service_nor_persistence.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

E_CWORD33_Status NPSynchronousReadPersistentData (PCSTR pAppName, PCSTR notif_name, PVOID pData, UI_32 uiDataSize, const E_CWORD33_PersistCategory ePersistCategory=e_CWORD33_UserData)

E_CWORD33_Status NPSynchronousWritePersistentData (PCSTR pAppName, PCSTR notif_name, PVOID pData, const UI_32 uiDataSize, const E_CWORD33_PersistCategory ePersistCategory=e_CWORD33_UserData)

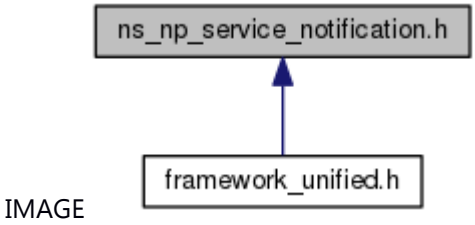
Detailed Description

This file contains the declaration for methods to retrieve and store data for NOR storage.

ns_np_service_notification.h File Reference

Notifications for notification service messages used by senders and receivers.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **NTFY_NPPService_UserChange** "NPPService/UserChange"

Detailed Description

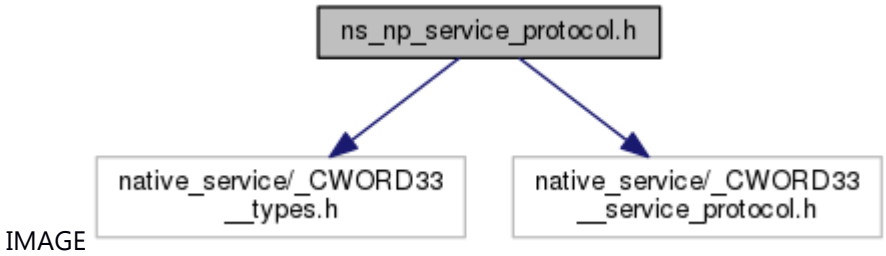
Notifications for notification service messages used by senders and receivers.

ns_np_service_protocol.h File Reference

API Header for Notification Service messages used by senders and recievers.

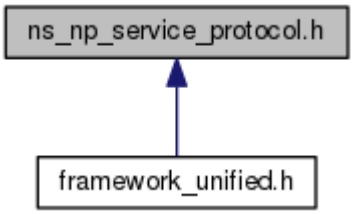
#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__service_protocol.h>

Include dependency graph for ns_np_service_protocol.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Typedefs

typedef enum _NS_NPServiceProtocol NS_NPServiceProtocol
typedef enum _NS_NPServiceEvent NS_NPServiceEvent

Enumerations

enum _NS_NPServiceProtocol { **NPS_REGISTER_EV_REQ** = PROTOCOL__CWORD33__BASE_CMD + 0, NPS_UNREGISTER_EV_REQ, NPS_PUBLISH_EV_REQ, NPS_NOTIFY_EV_REQ, NPS_SUBSCRIBE_TO_EV_REQ, NPS_UNSUBSCRIBE_FROM_EV_REQ, NPS_SET_PERSIST_FILE_PATH_REQ, NPS_GET_PERS_DATA_REQ, NPS_GET_PERS_DATA_ACK, NPS_GET_PERS_FILE_REQ, NPS_GET_PERS_FILE_ACK, NPS_SAVE_PERS_DATA_REQ, NPS_SAVE_PERS_DATA_ACK, NPS_RELEASE_PERS_FILE_REQ, NPS_NPP_STOP_REQ, NPS_TST_WAKEUP, NPS_BATCH_SUBSCRIBE_TO_EV_REQ, NPS_GET_PERS_DATA_FAILED_ACK, NPS_SET_PERSONALITY_REQ, NPS_CHANGE_PERSONALITY_REQ, NPS_USER_CHANGE_REQ, NPS_SET_PERSIST_FOLDER_PATH_REQ, NPS_GET_PERS_FOLDER_REQ, NPS_RELEASE_PERS_FOLDER_REQ, NPS_GET_PERS_FOLDER_ACK, NPS_NPP_STOP_ACK, NPS_BATCH_UNSUBSCRIBE_FROM_EV_REQ, NPS_GET_READYSTATUS_REQ, NPS_GET_READYSTATUS_ACK, NPS_REGISTER_NOR_EV_REQ, NPS_DELETE_PERSISTED_DATA_REQ, NPS_DELETE_PERSISTED_DATA_ACK, NPS_SET_DEFAULT_PERS_DATA, NPS_SET_NOTFN_PERSISTENT_TYPE, NPS_SET_FILE_PERSISTENT_TYPE, NPS_SET_FOLDER_PERSISTENT_TYPE,

NPS_SYNCHRONOUS_WRITE_NOTIFY_REQ, NPS_IMMEDIATE_WRITE_ACK,
NPS_NPP_SYNC_REQ }
enum _NS_NPServiceEvent { **NPS_NPP_READY_EVENT** = *** }

Detailed Description

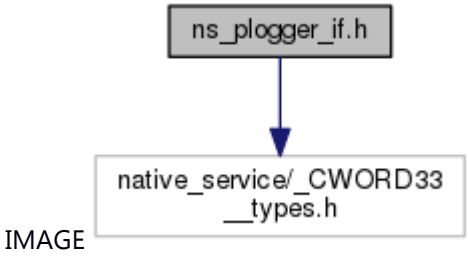
API Header for Notification Service messages used by senders and recievers.

ns_plogger_if.h File Reference

Declares functions and macros related to performance logging.

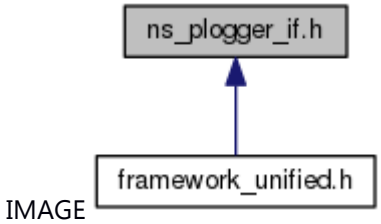
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_plogger_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct __CWORD33_LOGPPARAM

Performance LOG parameter. *More...*

struct __CWORD33_LOGSYSEVENTPARAM

System event LOG parameter. *More...*

#define **NS_PLOGQ** "/NSplog"

#define **NS_SYSEVENTLOGQ** "/NSSysEventLog"

#define **PLOG_TEXT**(args...) NSLogPrintPerformanceLog(__LINE__, __FUNCTION__, ## args)

#define **SYSEVENTLOG_TEXT**(args...) NSLogSysEvent(__LINE__, __FUNCTION__, ## args)

#define **IS_PLOG_ENABLED**() NsLogIsPLogEnabled()

#define **IS_SYSEVENTLOG_ENABLED**() NSLogIsSysEventLogEnabled()

#define _CWORD33_LOG_PERFORMANCE_DEBUG(print_fmt, args...) ((IS_PLOG_ENABLED()) ? PLOG_TEXT(print_fmt, ## args): ((void)0))

_CWORD33_LOG_PERFORMANCE_DEBUG

#define _CWORD33_LOG_PERFORMANCE(print_fmt, args...) ((IS_PLOG_ENABLED()) ? PLOG_TEXT(print_fmt, ## args): ((void)0))

_CWORD33_LOG_PERFORMANCE

#define _CWORD33_LOG_SYSTEMEVENT(print_fmt, args...)

enum _EPLOG_TIME_FORMAT { EPLOG_TIME_FORMAT_USEC, EPLOG_TIME_FORMAT_MSEC, EPLOG_TIME_FORMAT_SEC }*Performance logging.*

enum _EPLOG_OUTPUT_OPTION { EPLOG_SLOG = ***, EPLOG_MSGQ = ***, EPLOG_TRACEEVENT = *** }
typedef enum _EPLOG_TIME_FORMAT EPLOG_TIME_FORMAT

Performance logging.

typedef enum _EPLOG_OUTPUT_OPTION **EPLOG_OUTPUT_OPTION**
typedef struct __CWORD33_LOGPPARAM _CWORD33_LOGPPARAM

Performance LOG parameter.

typedef struct __CWORD33_LOGPPARAM * **L_CWORD33_LOGPPARAM**
typedef struct __CWORD33_LOGSYSEVENTPARAM _CWORD33_LOGSYSEVENTPARAM

System event LOG parameter.

typedef struct __CWORD33_LOGSYSEVENTPARAM * **L_CWORD33_LOGSYSEVENTPARAM**
_CWORD33_LOGPPARAM **g__CWORD33_LogPParams**
_CWORD33_LOGSYSEVENTPARAM **g__CWORD33_LogSysEventParams**
VOID NSLogPrintPerformanceLog (const UI_16 f_ui16Line, PCSTR f_cFuncName, PCSTR __format,...)
VOID NSLogEnablePLog (BOOL f_bEnable)
BOOL NsLogIsPLogEnabled (void)
VOID NSLogSetPlogTimeFormat (EPLOG_TIME_FORMAT f_ePlogTimeFormat)
VOID NSLogSetPlogOutputOptions (UI_8 f_uiPlogOutputOption)
VOID NSLogSysEvent (const UI_16 f_ui16Line, PCSTR f_cFuncName, PCSTR __format,...)
VOID NSLogEnableSysEventLog (BOOL f_bEnable)
BOOL NSLogIsSysEventLogEnabled (void)

Detailed Description

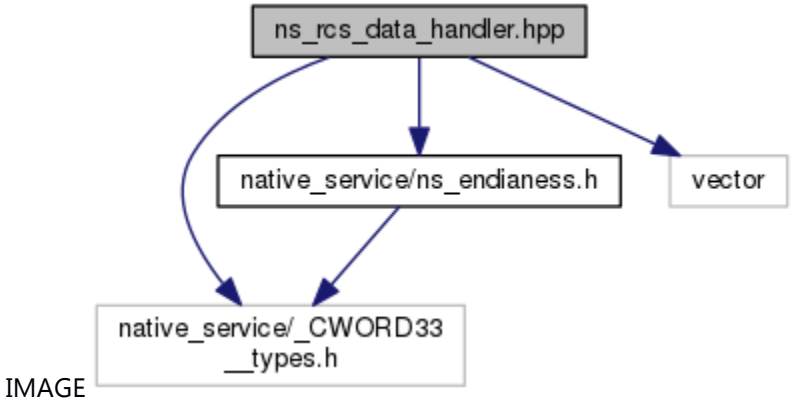
Declares functions and macros related to performance logging.

ns_rcs_data_handler.hpp File Reference

This file contains declaration of CPassThruInDataHandler class.

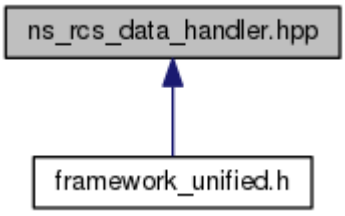
#include <native_service/_CWORD33__types.h>
#include <native_service/ns_endianess.h>
#include <vector>

Include dependency graph for ns_rcs_data_handler.hpp:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class CPassThruInDataHandler

Class: CPassThruInDataHandler. class CPassThruOutDataHandler

***Data Handler.* Macros**

#define **SIZE_OF_PASSTHRU_DATATYPE_IDENTIFIER** 1
#define **SIZE_OF_PASSTHRU_PARAM_BUFFERTYPE_HEADER** 3
#define **PASSTHRU_DATA_HEADER_LEN** 8

Typedefs

typedef UI_8 **NSRCS_BOOL**

Detailed Description

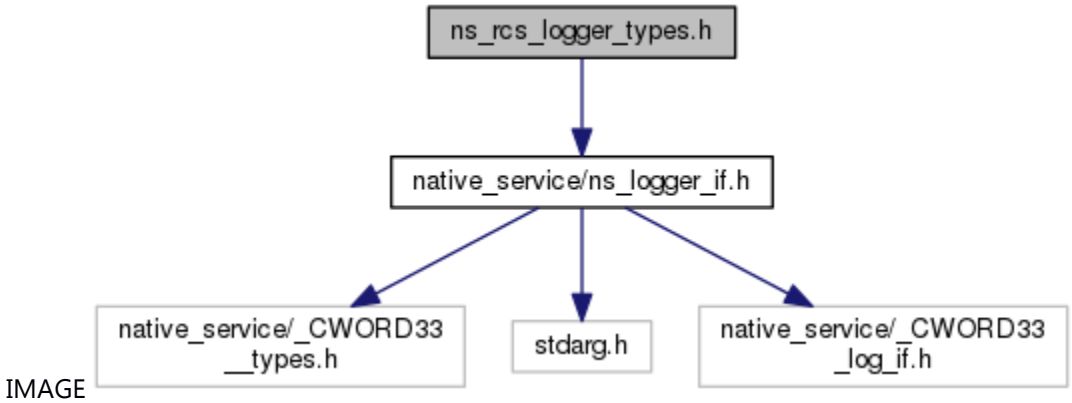
This file contains declaration of CPassThruInDataHandler class.

ns_rcs_logger_types.h File Reference

This file provide logger type's structures.

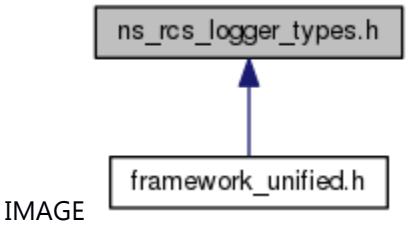
#include <native_service/ns_logger_if.h>

Include dependency graph for ns_rcs_logger_types.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _TNSRCS_SetLogSettingsReq
struct _TNSRCS_SetLogSettingsResp
struct _TNSRCS_GetLogSettingsReq
struct _TNSRCS_GetLogSettingsResp
struct _TNSRCS_SetLogMaskResp
struct _TNSRCS_SetLogOutOptResp
struct _TNSRCS_SetLogSeverityResp

Typedefs

typedef struct _TNSRCS_SetLogSettingsReq **TNSRCS_SetLogSettingsReq**
typedef struct _TNSRCS_SetLogSettingsResp **TNSRCS_SetLogSettingsResp**
typedef struct _TNSRCS_GetLogSettingsReq **TNSRCS_GetLogSettingsReq**
typedef struct _TNSRCS_GetLogSettingsResp **TNSRCS_GetLogSettingsResp**
typedef struct _TNSRCS_SetLogMaskResp **TNSRCS_SetLogMaskResp**
typedef struct _TNSRCS_SetLogOutOptResp **TNSRCS_SetLogOutOptResp**
typedef struct _TNSRCS_SetLogSeverityResp **TNSRCS_SetLogSeverityResp**

Detailed Description

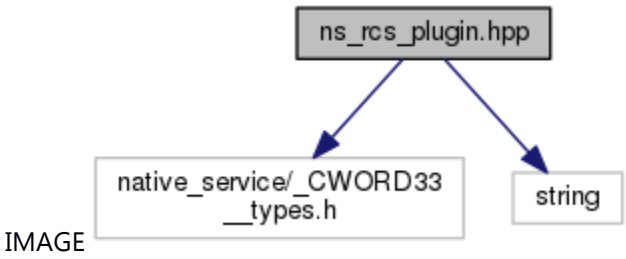
This file provide logger type's structures.

ns_rcs_plugin.hpp File Reference

This file contains the declaration of CNSRcsPlugin class.

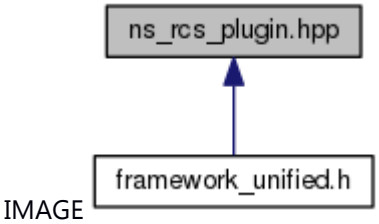
#include <native_service/_CWORD33__types.h>
#include <string>

Include dependency graph for ns_rcs_plugin.hpp:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class CNSRcsPlugin

***CNSRcsPlugin*. Typedefs**

typedef E_CWORD78_Status(* **TFPNSRcsSendPassthruData**) (UI_8 f_ui8clientId, PVOID f_data, UI_16 f_ui16PayloadLength)

Detailed Description

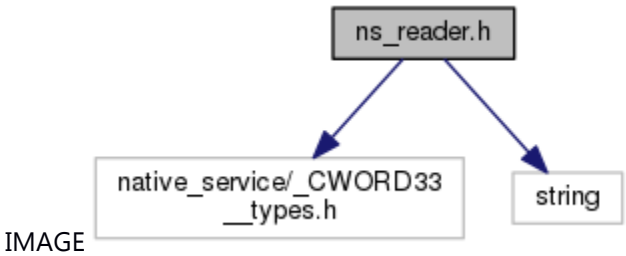
This file contains the declaration of CNSRcsPlugin class.

ns_reader.h File Reference

This file contains declaration of class IConfigReader.

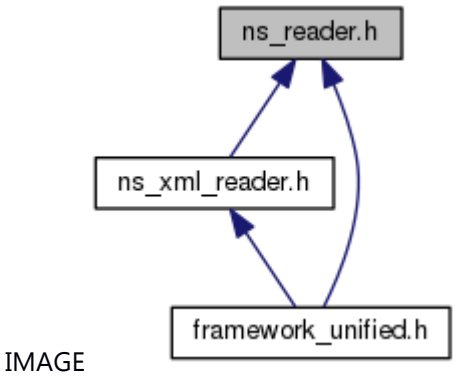
#include <native_service/_CWORD33__types.h>
#include <string>

Include dependency graph for ns_reader.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class IConfigReader
IConfigReader.

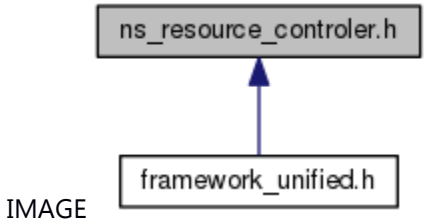
Detailed Description

This file contains declaration of class IConfigReader.

ns_resource_controler.h File Reference

This file provides api for operating Resource.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **_CWORD33__RES_ABNMLMON** "_CWORD33__RES_ABNMLMON"
#define **_CWORD33__RES_TIMER** "_CWORD33__RES_TIMER"

Functions

int _CWORD33_GetResource (const char *mod, const char *key, long *resource)
int _CWORD33_AcquireResouce (const char *mod, const char *key, long *resource)
int _CWORD33_ReleaseResouce (const char *mod, const char *key)
int _CWORD33_SearchResourseKey (const char *mod, long resource, const char **key)
int _CWORD33_RegistResouce (const char *mod, const char *key, long resource, int init_counter)
int _CWORD33_UnregistResouce (const char *mod, const char *key)

Detailed Description

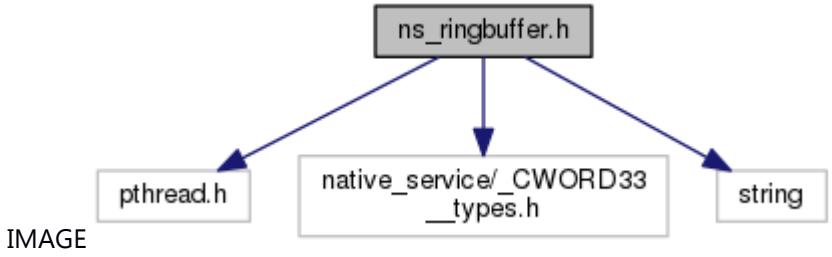
This file provides api for operating Resource.

ns_ringbuffer.h File Reference

This file contains implementation of class CNSRingBuffer. This class provides API to open, read, write and close ring buffer.

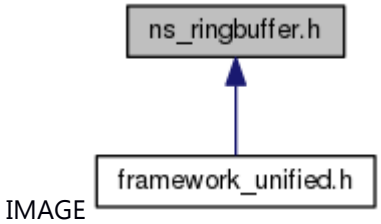
#include <pthread.h>
#include <native_service/_CWORD33__types.h>
#include <string>

Include dependency graph for ns_ringbuffer.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _NSRingBufferHdr
struct _NSRingBufferMtx
class CNSRingBuffer

***CNSRingBuffer*. Typedefs**

typedef struct _NSRingBufferHdr **NSRingBufferHdr**
typedef struct _NSRingBufferMtx **NSRingBufferMtx**

Detailed Description

This file contains implementation of class CNSRingBuffer. This class provides API to open, read, write and close ring buffer.

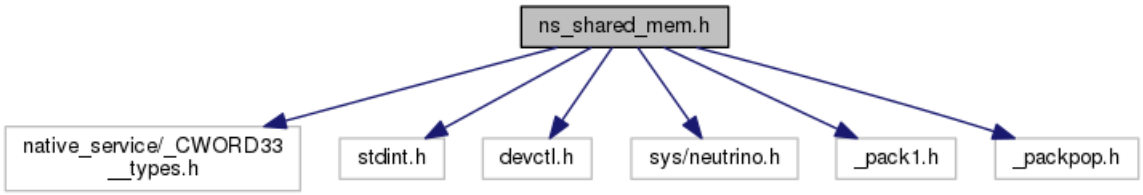
ns_shared_mem.h File Reference

Shared interface file for SharedMemory Service Detail description of interface and usage http://ptc_CWORD33_accbuild2/TikiWiki/tiki-index.php?page=Shared+Memory http://ptc_CWORD33_aspsql01/sites/ptcsweng/SW%20Engineering%20Document%20Library/ Projects/Connected%20Color%20Radio%20(CCR)/Architecture/Framework/CCR%20Messaging.vsd.

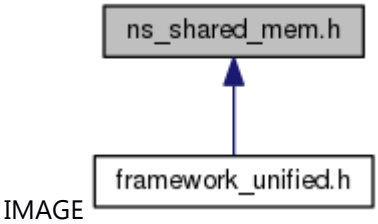
```
#include <native_service/_CWORD33__types.h>
#include <stdint.h>
#include <devctl.h>
#include <sys/neutrino.h>
#include <_pack1.h>
#include <_packpop.h>
```

Include dependency graph for ns_shared_mem.h:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct SMsgHeader
struct SReplyHeader
#define **NS_SHAREDMEM_NAME** "NS_SharedMem"
#define **NS_SHAREDMEM_SOCK_QUEUE** 10

enum ESharedMemCommand { **eSharedMemCommandWrite** = ***, **eSharedMemCommandRead** = ***, **eSharedMemCommandSize** = ***, **eSharedMemCommandDelete** = *** }

Header for system-provided messages

typedef struct _pulse_t TPosixHdr

< *Header file for Template*

DEFINE_HANDLE_TYPE (SSHMemID, TShMemID)

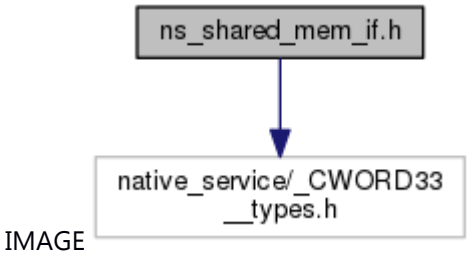
Detailed Description

ns_shared_mem_if.h File Reference

API Header for Logger. Declares the external APIs to Logger.

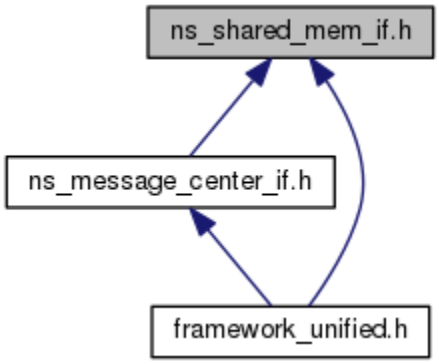
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_shared_mem_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **BAD_MEM_ID** (0)

Typedefs

typedef unsigned int TMemID

< *Standard CWORD3 Types*

Functions

TMemID SetDataToShared (const void *data, UI_32 dataBytes, const char *from, const char *to)
E_CWORD33_Status GetDataFromShared (TMemID id, void *data, UI_32 dataMaxBytes)
UI_32 GetLengthOfDataFromShared (TMemID id)
E_CWORD33_Status DiscardDataFromShared (TMemID id)

Detailed Description

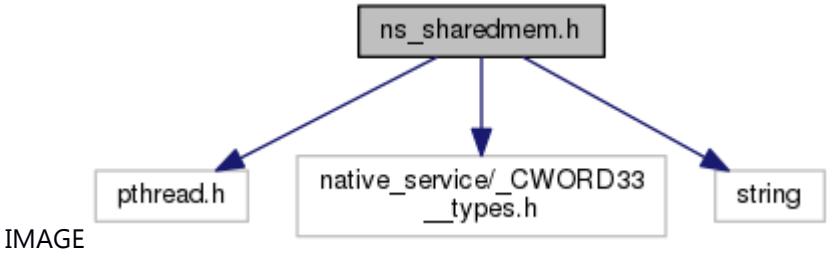
API Header for Logger. Declares the external APIs to Logger.

ns_sharedmem.h File Reference

This file contains implementation of class CNSSharedMem. This class provides API to open, read, write and close shared memory.

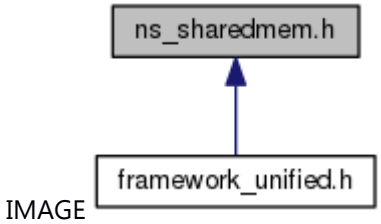
#include <pthread.h>
#include <native_service/_CWORD33__types.h>
#include <string>

Include dependency graph for ns_sharedmem.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _NSSharedBuffer
class CNSSharedMem
this file has the C_CWORD33_Guard class definitions #define **NSTEST_FAIL_SHAREDMEM_OPEN** "NSTEST_FAIL_SHAREDMEM_OPEN"
typedef struct _NSSharedBuffer **NSSharedBufferHdr**

Detailed Description

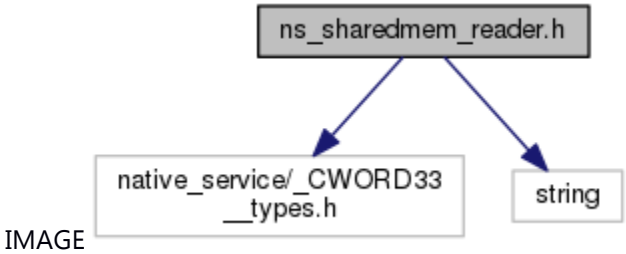
This file contains implementation of class CNSSharedMem. This class provides API to open, read, write and close shared memory.

ns_sharedmem_reader.h File Reference

This file contains implementation of class CNSSharedMemReader. This class provides API to open, close and perform read operation on shared memory.

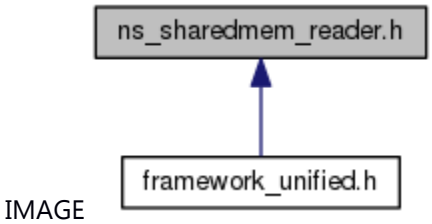
#include <native_service/_CWORD33__types.h>
#include <string>

Include dependency graph for ns_sharedmem_reader.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class CNSSharedMemReader
read shared memory

Detailed Description

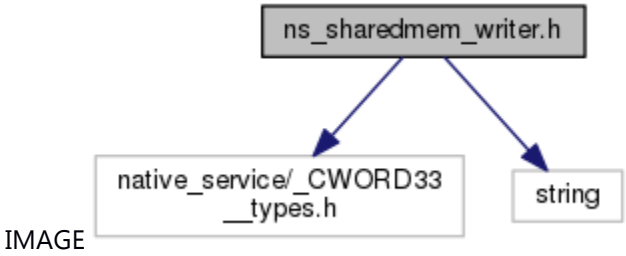
This file contains implementation of class CNSSharedMemReader. This class provides API to open, close and perform read operation on shared memory.

ns_sharedmem_writer.h File Reference

This file contains implementation of class CNSSharedMemWriter. This class provides API to open, close and perform write operation on shared memory.

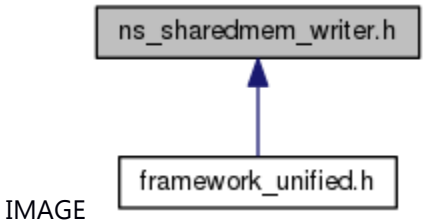
#include <native_service/_CWORD33__types.h>
#include <string>

Include dependency graph for ns_sharedmem_writer.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class CNSSharedMemWriter

write shared memory

Detailed Description

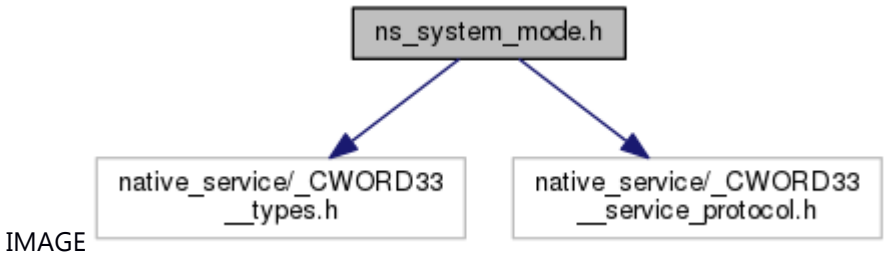
This file contains implementation of class CNSSharedMemWriter. This class provides API to open, close and perform write operation on shared memory.

ns_system_mode.h File Reference

The file defines enum SystemModeProtocol.

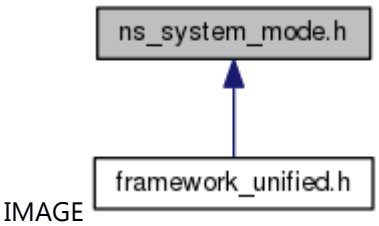
#include <native_service/_CWORD33__types.h>
#include <native_service/_CWORD33__service_protocol.h>

Include dependency graph for ns_system_mode.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

enum _SystemModeProtocol { SYSTEM_ON_INITIALIZATION = PROTOCOL_THREAD_INITIALIZATION, SYSTEM_ON_WAKEUP = PROTOCOL_THREAD_WAKEUP, SYSTEM_ON_SHUTDOWN = PROTOCOL_THREAD_SHUTDOWN, SYSTEM_ON_DESTROY = PROTOCOL_THREAD_DESTROY }*System Mode Protocol.*

typedef enum _SystemModeProtocol SystemModeProtocol

System Mode Protocol.

Detailed Description

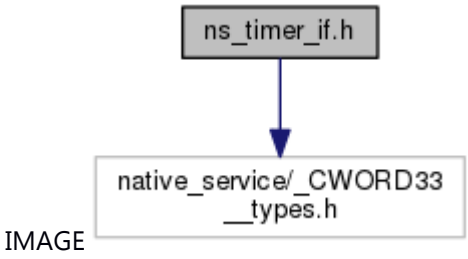
The file defines enum SystemModeProtocol.

ns_timer_if.h File Reference

APIs to create, delete and use Native Service timers .

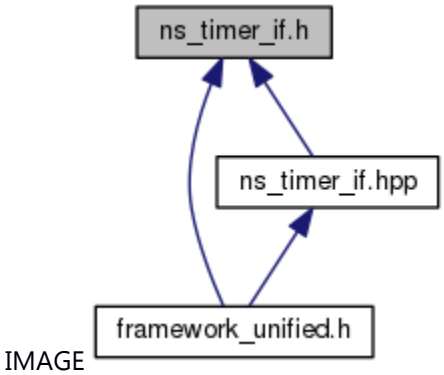
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_timer_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _NSTimerInfo

Timer info, defines the initial start of a timer, the repeat timer values and the cmd id for a timer. More...

Macros

#define **TIMER_QUE** "TIMER"
#define **MAX_SERVICE_NAME** 15

Typedefs

typedef void(* TimerCb) (UI_16 cmd)

Timer Function Pointer definition Detailed description of the class.

typedef struct _NSTimerInfo NSTimerInfo

Timer info, defines the initial start of a timer, the repeat timer values and the cmd id for a timer.

typedef enum _NSTimerCallbackMechanism eNSTimerCallbackMechanism

Enum Types for valid Callback Mechanisms for a NS_Timer.

Enumerations

enum _NSTimerCallbackMechanism { **CALLBACK_MESSAGE** }*Enum Types for valid Callback Mechanisms for a NS_Timer.*

Functions

time_t WholeSeconds (UI_32 ms)

Helper methods that convert time provided in MS. mseconds.

UI_32 RemainderMs (UI_32 ms)
UI_64 MSToNS (UI_32 ms)
HANDLE NS_TimerCreate (NSTimerInfo timer_info, eNSTimerCallbackMechanism cbMech, HANDLE sndMqHndl)
E_CWORD33_Status NS_TimerDelete (HANDLE hTimer)
E_CWORD33_Status NS_TimerSetTime (HANDLE hTimer, NSTimerInfo timer_info)
E_CWORD33_Status NS_TimerGetTime (HANDLE hTimer, NSTimerInfo *timer_info)
void NS_TimerDebugOn (BOOL FlagState)

Detailed Description

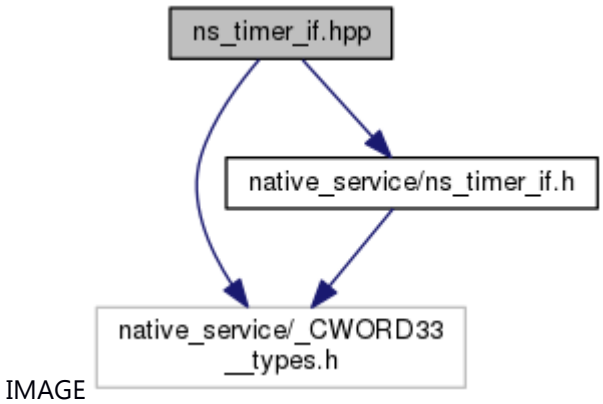
APIs to create, delete and use Native Service timers .

ns_timer_if.hpp File Reference

Header file for Timer class.

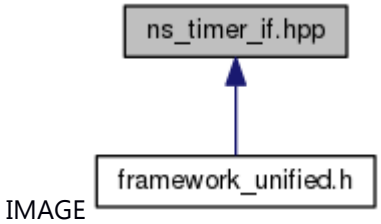
#include <native_service/_CWORD33__types.h>
#include <native_service/ns_timer_if.h>

Include dependency graph for ns_timer_if.hpp:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class NSTimer

Handle Timer.

Detailed Description

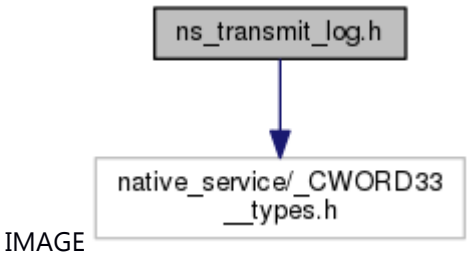
Header file for Timer class.

ns_transmit_log.h File Reference

This file contains declaration of wrapper C type API's for class CNSSharedMem to read and write transmit log to shared memory.

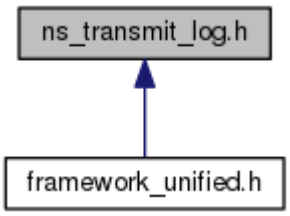
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_transmit_log.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

#define **TRANSMIT_LOG_SHAREDMEM_NAME** "/TransmitLogShBuf"
#define **TRANSMIT_LOG_SHAREDMEM_SIZE** 5242880
E_CWORD33_Status NSSharedMemTransmitLogOpen ()
E_CWORD33_Status NSSharedMemTransmitLogClose ()
SI_32 NSSharedMemReadTransmitLog (PSTR f_pBuffer, const UI_32 f_uiLength, const BOOL f_bBlock)
SI_32 NSSharedMemWriteTransmitLog (PCSTR f_pBuffer, const UI_32 f_uiLength)
E_CWORD33_Status NSSharedMemDumpTransmitLogToFile (PCSTR f_pPath, PUI_32 f_puiDumpSize)
BOOL NSSharedMemTransmitLogIsOpen ()

Detailed Description

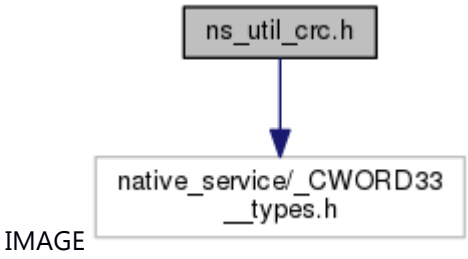
This file contains declaration of wrapper C type API's for class CNSSharedMem to read and write transmit log to shared memory.

ns_util_crc.h File Reference

This file contains declaration of APIs to calcuate 16-bit and 32-bit CRC checksum of file.

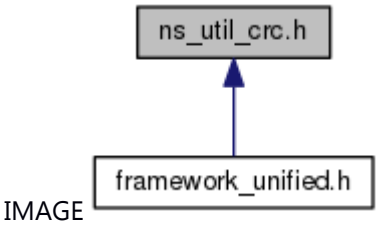
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_util_crc.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Functions

E_CWORD33_Status CalculateCRC16 (PCSTR f_c_file_name, UI_16 &f_ui_check_sum)
E_CWORD33_Status CalculateCRC32 (PCSTR f_c_file_name, UI_32 &f_ui_check_sum)

Detailed Description

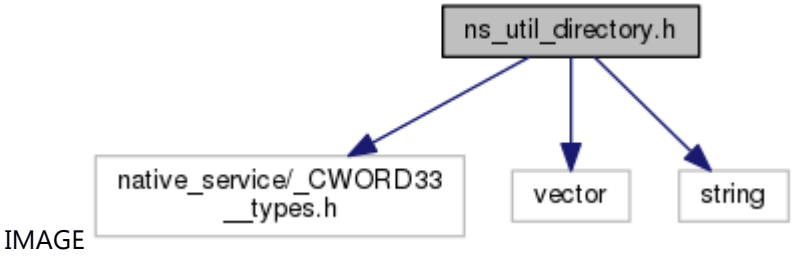
This file contains declaration of APIs to calcuate 16-bit and 32-bit CRC checksum of file.

ns_util_directory.h File Reference

This file contains declaration of ns_util_directory.

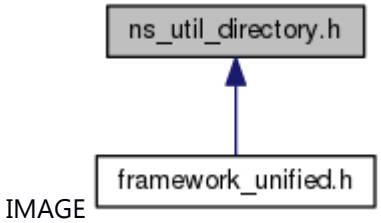
#include <native_service/_CWORD33__types.h>
#include <vector>
#include <string>

Include dependency graph for ns_util_directory.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Typedefs

typedef std::vector< std::string > TFileList
typedef TFileList::iterator TFileListIterator

Functions

E_CWORD33_Status GetFileList (TFileList *f_pv_tfile_list, PCSTR f_pc_path)
BOOL DoesDirecotryExist (std::string f_c_dir_path)
E_CWORD33_Status CreateDirectory (std::string f_c_dir_path)

Detailed Description

This file contains declaration of ns_util_directory.

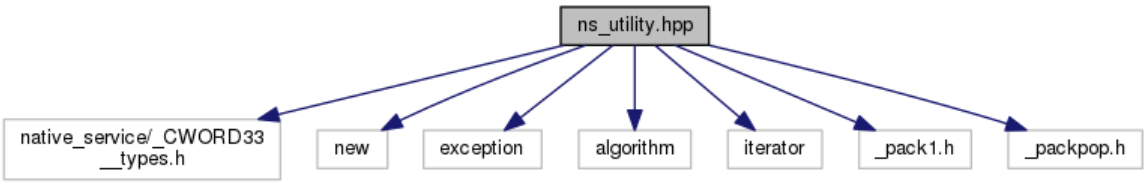
ns_utility.hpp File Reference

This file contains declaration of common APIs for NS_UtilityCenter.

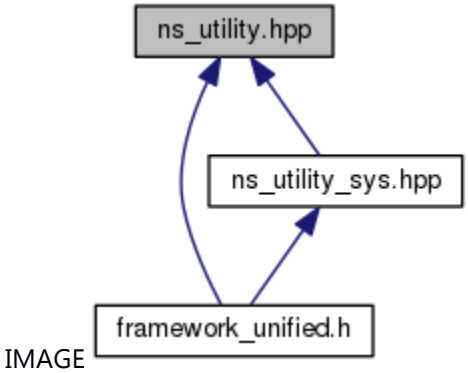
#include <native_service/_CWORD33__types.h>
#include <new>
#include <exception>
#include <algorithm>
#include <iterator>
#include <_pack1.h>
#include <_packpop.h>

Include dependency graph for ns_utility.hpp:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct Int2Type< v >
struct Type2Type< T >
struct TList< T, N >
struct TList< T, 1 >
struct FSig< Sig >
struct FSig< R(*)() >
struct FSig< R(*)(T1) >
struct FSig< R(*)(T1, T2) >
struct FSig< R(*)(T1, T2, T3) >
struct FSig< R(*)(T1, T2, T3, T4) >
struct FSig< R(*)(T1, T2, T3, T4, T5) >

struct FSig< R(*)(T1, T2, T3, T4, T5, T6) >
struct FSig< R(*)(T1, T2, T3, T4, T5, T6, T7) >
class Accumulator< T >
Accumultor type. class MemTraits< T >
class RaiseExceptionPolicy< RsrcTraits >
class CheckForErrorPolicy< RsrcTraits >
class ResourceMgr< T, RsrcTraits, ErrorPolicy >
class IFunctor< R >
class CFunctor0< TFn >
class CFunctor1< TFn, Arg1 >
class CFunctor2< TFn, Arg1, Arg2 >
class CFunctor3< TFn, Arg1, Arg2, Arg3 >
class CFunctor4< TFn, Arg1, Arg2, Arg3, Arg4 >
class CFunctor5< TFn, Arg1, Arg2, Arg3, Arg4, Arg5 >
class CFunctor6< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6 >
class CFunctor7< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7 >
#define **DISALLOW_COPY_AND_ASSIGN**(TypeName)
#define **DEFINE_EXCEPTION**(name, base)
template<class T , UI_32 N> const T * **ArrBeg** (const T(&arr)[N])
template<class T , UI_32 N> const T * **ArrEnd** (const T(&arr)[N])
template<class O , class I > O **SimpleCast** (I i)
template<class R , class B > R **UnalignedRet** (B *b)
template<class Cont , class Gen > Cont **genRange** (UI_32 N, Gen genFn)
template<class OCont , class IIter , class MapFn > OCont **mapRange** (const IIter &begin, const IIter &end, MapFn fn)
template<typename T > Accumulator< T > MakeAccumulator (T n)
Accumulator utility function.
template<class TFn > CFunctor0< TFn > **functor** (TFn fn)
template<class TFn , class Arg1 > CFunctor1< TFn, Arg1 > **functor** (TFn fn, Arg1 arg1)
template<class TFn , class Arg1 , class Arg2 > CFunctor2< TFn, Arg1, Arg2 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2)
template<class TFn , class Arg1 , class Arg2 , class Arg3 > CFunctor3< TFn, Arg1, Arg2, Arg3 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3)
template<class TFn , class Arg1 , class Arg2 , class Arg3 , class Arg4 > CFunctor4< TFn, Arg1, Arg2, Arg3, Arg4 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4)
template<class TFn , class Arg1 , class Arg2 , class Arg3 , class Arg4 , class Arg5 > CFunctor5< TFn, Arg1, Arg2, Arg3, Arg4, Arg5 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4, Arg5 arg5)
template<class TFn , class Arg1 , class Arg2 , class Arg3 , class Arg4 , class Arg5 , class Arg6 > CFunctor6< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4, Arg5 arg5, Arg6 arg6)
template<class TFn , class Arg1 , class Arg2 , class Arg3 , class Arg4 , class Arg5 , class Arg6 , class Arg7 > CFunctor7< TFn, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7 > **functor** (TFn fn, Arg1 arg1, Arg2 arg2, Arg3 arg3, Arg4 arg4, Arg5 arg5, Arg6 arg6, Arg7 arg7)

Detailed Description

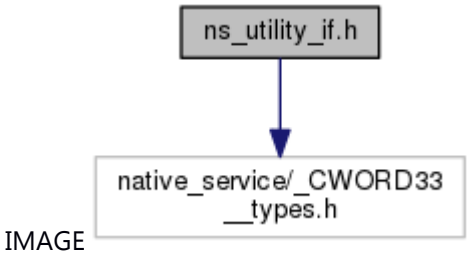
This file contains declaration of common APIs for NS_UtilityCenter.

ns_utility_if.h File Reference

This file contains declaration of common APIs for NS_UtilityCenter.

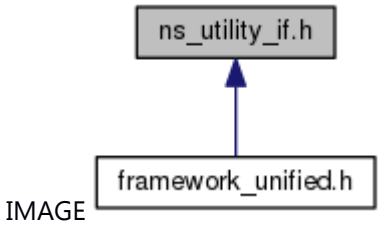
#include <native_service/_CWORD33__types.h>

Include dependency graph for ns_utility_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define NS_INVALID_RETURN -1
#define NS_SETBIT(x, i) ((x) |= (1 << (i)))
#define NS_CLEARBIT(x, i) ((x) &= ~(1 << (i)))
#define NS_TOGGLEBIT(x, i) ((x) ^= (1 << (i)))
#define NS_ISBITSET(x, i) (((x) & (1 << (i))) != ***)

Detailed Description

This file contains declaration of common APIs for NS_UtilityCenter.

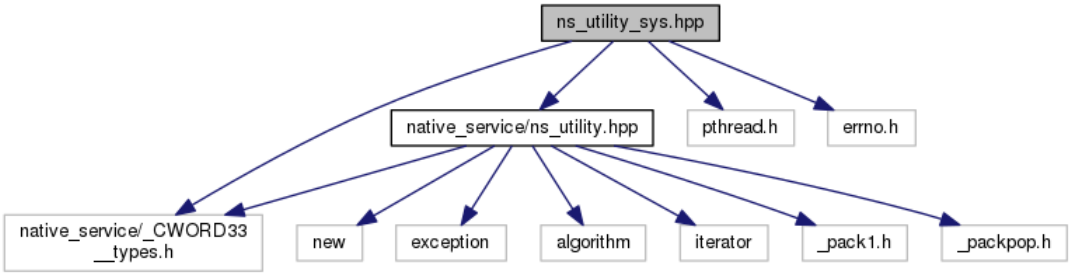
ns_utility_sys.hpp File Reference

This file contains declaration of common APIs for NS_UtilityCenter.

#include <native_service/_CWORD33__types.h>
#include <native_service/ns_utility.hpp>
#include <pthread.h>
#include <errno.h>

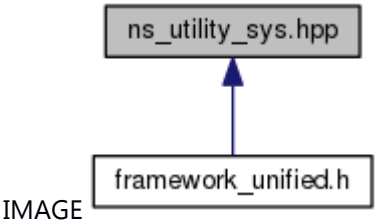
Include dependency graph for ns_utility_sys.hpp:

IMAGE



This graph shows which files directly or indirectly include this file:

IMAGE



Classes

class CMutex

Mutex Lock Class. class CRWLock

***Read/Write Lock Class.* Functions**

DEFINE_EXCEPTION (lock_error, std::runtime_error)
DEFINE_EXCEPTION (lock_creation_error, lock_error)
DEFINE_EXCEPTION (lock_acquireread_error, lock_error)
DEFINE_EXCEPTION (lock_acquirewrite_error, lock_error)
DEFINE_EXCEPTION (lock_release_error, lock_error)

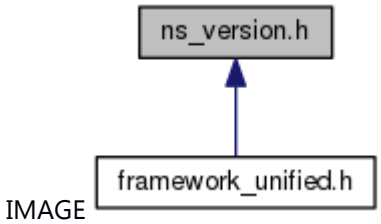
Detailed Description

This file contains declaration of common APIs for NS_UtilityCenter.

ns_version.h File Reference

This file provides version define.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **MAJORNO** 0x02
#define **MINORNO** 0x02
#define **REVISION** 0x05

Detailed Description

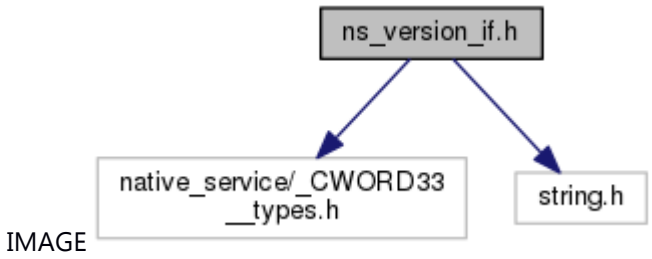
This file provides version define.

ns_version_if.h File Reference

Header for C_CWORD33_Version class.

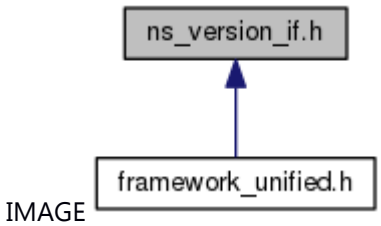
#include <native_service/_CWORD33__types.h>
#include <string.h>

Include dependency graph for ns_version_if.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class C_CWORD33_Version

***Version Info Class.* Macros**

#define _CWORD33__GET_VERSION() "undefined_undefined_00.00.00"
#define _CWORD33__APP_VERSION_MAJOR() (0)
#define _CWORD33__APP_VERSION_MINOR() (0)
#define _CWORD33__APP_VERSION_REVISION() (0)
#define _CWORD33__APP_VERSION_BUILDVER() "undefined"
#define _CWORD33__APP_VERSION_PRODUCTID() "undefined"

Detailed Description

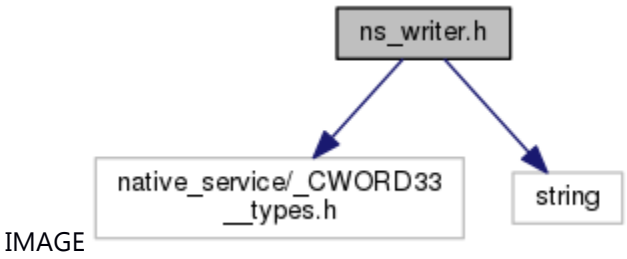
Header for C_CWORD33_Version cl

ns_writer.h File Reference

This file contains declaration of class IConfigWriter.

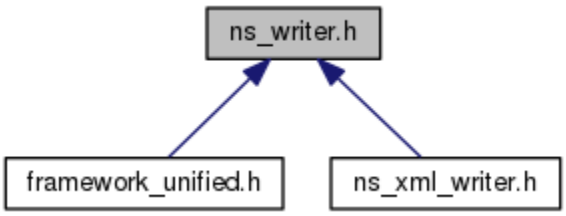
#include <native_service/_CWORD33__types.h>
#include <string>

Include dependency graph for ns_writer.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class IConfigWriter

Config writer abstract class.

Detailed Description

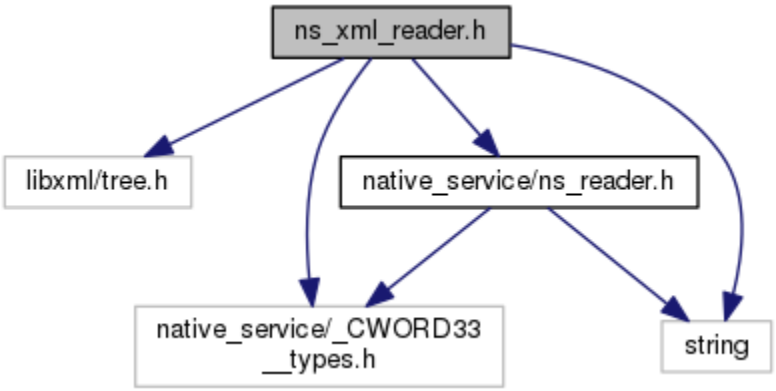
This file contains declaration of class IConfigWriter.

ns_xml_reader.h File Reference

This file contains declaration of class CXMLReader.

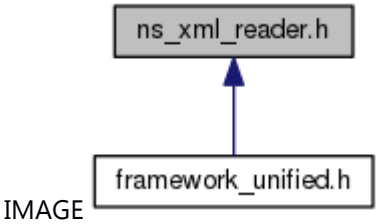
#include <libxml/tree.h>
#include <native_service/_CWORD33__types.h>
#include <native_service/ns_reader.h>
#include <string>

Include dependency graph for ns_xml_reader.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class CXMLReader

***This file contains declaration of class CXMLReader*. Functions**

CXMLReader * GetCXMLReaderObject (CHAR *f_cFilePath)

Detailed Description

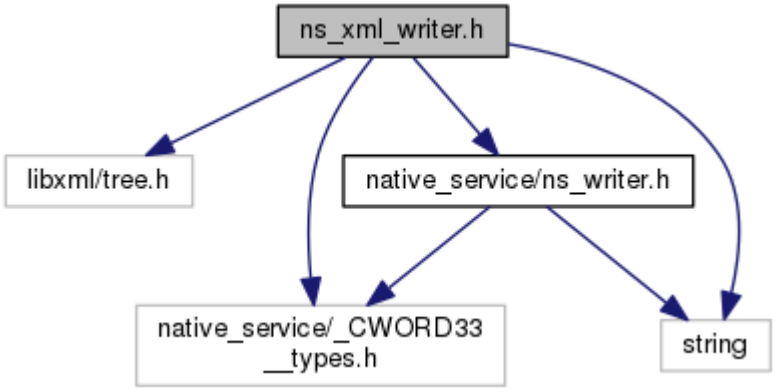
This file contains declaration of class CXMLReader.

ns_xml_writer.h File Reference

This file contains declaration of class CXMLWriter.

#include <libxml/tree.h>
#include <native_service/_CWORD33__types.h>
#include <native_service/ns_writer.h>
#include <string>

Include dependency graph for ns_xml_writer.h:



IMAGE

Classes

class CXMLWriter

This file contains declaration of class CXMLWriter. **Functions**

CXMLWriter * GetCXMLWriterObject (CHAR *f_cFilePath)
CXMLWriter * GetCXMLWriterObjectNoParam ()

Detailed Description

This file contains declaration of class CXMLWriter.

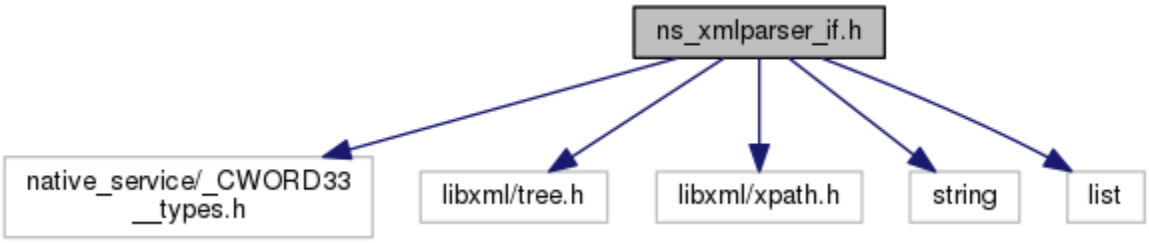
ns_xmlparser_if.h File Reference

This file contains declaration of class CXmlParser, CXmlNode and CXmlAttr. This file provides the interface for Parsing XML file, API for operation on node of an xml structure.

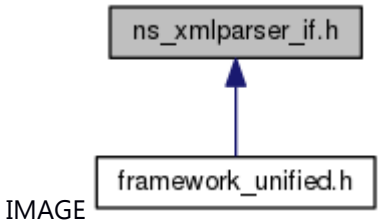
#include <native_service/_CWORD33__types.h>
#include <libxml/tree.h>
#include <libxml/xpath.h>
#include <string>
#include <list>

Include dependency graph for ns_xmlparser_if.h:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class CXmlNode

This class represents node of an XML. class CXmlAttr

This class represents the attribute of an xml node. class CXmlParser

This class represents the XML parser. **Typedefs**

typedef enum _E_CWORD33_XmlNodeTypes E_CWORD33_XmlNodeTypes

Include Files.

typedef std::list< CXmlNode > **TNodeList**
typedef TNodeList::iterator **TNodeListIterator**
typedef std::list< CXmlAttr > **TAttrList**
typedef std::list< CXmlAttr >::iterator **TAttrListIterator**

Enumerations

enum _E_CWORD33_XmlNodeTypes { _CWORD33__XML_NODE_NONE = ***, _CWORD33__XML_ELEMENT_NODE = ***, _CWORD33__XML_ATTRIBUTE_NODE = ***, _CWORD33__XML_TEXT_NODE = ***, _CWORD33__XML_CDATA_SECTION_NODE = ***, _CWORD33__XML_ENTITY_REF_NODE = ***, _CWORD33__XML_ENTITY_NODE = ***, _CWORD33__XML_PI_NODE = ***, _CWORD33__XML_COMMENT_NODE = ***, _CWORD33__XML_DOCUMENT_NODE = ***, _CWORD33__XML_DOCUMENT_TYPE_NODE = ***, _CWORD33__XML_DOCUMENT_FRAG_NODE = ***, _CWORD33__XML_NOTATION_NODE = ***, _CWORD33__XML_HTML_DOCUMENT_NODE = ***, _CWORD33__XML_DTD_NODE = ***, _CWORD33__XML_ELEMENT_DECL = ***, _CWORD33__XML_ATTRIBUTE_DECL = ***, _CWORD33__XML_ENTITY_DECL = ***, _CWORD33__XML_NAMESPACE_DECL = ***, _CWORD33__XML_XINCLUDE_START = ***, _CWORD33__XML_XINCLUDE_END = ***, _CWORD33__XML_DOCB_DOCUMENT_NODE = *** }*Include Files.*

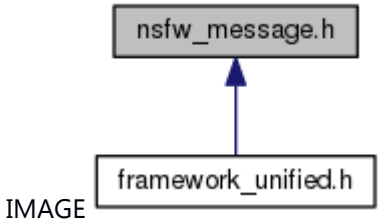
Detailed Description

This file contains declaration of class CXmlParser, CXmlNode and CXmlAttr. This file provides the interface for Parsing XML file, API for operation on node of an xml structure.

nsfw_message.h File Reference

The file definition of NSFW message for NSFW independent module reference.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **_NSFW_MSG_LEN_** (40)
#define **_NSFW_SYSINFO_FLAG_** (4)
#define **_NSFW_SYSINFO_SIZE_** (64)
#define **NSFW_GET_MESSAGE**(msg)

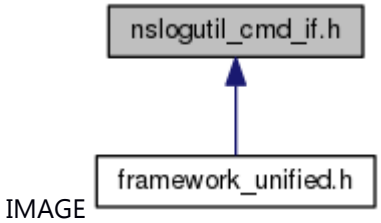
Detailed Description

The file definition of NSFW message for NSFW independent module reference.

nslogutil_cmd_if.h File Reference

Declares COMMANDS related to Log Utility.

This graph shows which files directly or indirectly include this file:



IMAGE

Macros

#define **NTFY_LOGGER_SETCONTROLMASK** "LoggerService/SetControlMask"

Typedefs

typedef enum _COMMANDS COMMANDS

Logger Utility Commands.

Enumerations

enum _COMMANDS { **SET_LOG_MASK** = ***, **GET_LOG_MASK**, **SET_LOG_OUT_OPT**, **GET_LOG_OUT_OPT**, **GET_LOG_MASK_RESPONSE**, **GET_LOG_OUT_OPT_RESPONSE**, **SET_MSGTX_RX_ON_AND_TX_OFF**, **SET_MSGTX_RX_OFF_AND_TX_ON**, **SET_MSGTX_RX_AND_TX_ON**, **SET_MSGTX_RX_AND_TX_OFF**, **kDebugDumpRequest**, **SET_LOG_SEVERITY**, **GET_LOG_SEVERITY**, **GET_LOG_SEVERITY_RESPONSE**, **NSRCS_SET_LOG_SETTINGS_REQ**, **NSRCS_SET_LOG_SETTINGS_RESP**, **NSRCS_GET_LOG_SETTINGS_REQ**, **NSRCS_GET_LOG_SETTINGS_RESP**, **NSRCS_SET_LOG_MASK_REQ**, **NSRCS_SET_LOG_MASK_RESP**, **NSRCS_SET_LOG_OUT_OPT_REQ**, **NSRCS_SET_LOG_OUT_OPT_RESP**, **NSRCS_SET_LOG_SEVERITY_REQ**, **NSRCS_SET_LOG_SEVERITY_RESP** }*Logger Utility Commands.*

Detailed Description

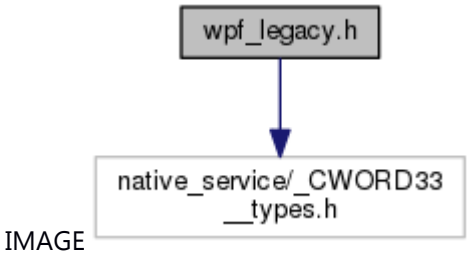
Declares COMMANDS related to Log Utility.

wpf_legacy.h File Reference

This file define primary types to use.

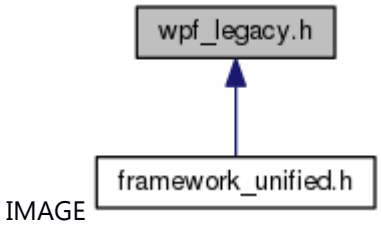
#include <native_service/_CWORD33__types.h>

Include dependency graph for wpf_legacy.h:



IMAGE

This graph shows which files directly or indirectly include this file:



IMAGE

Classes

struct _SYSTEMTIME

Macros

#define **CALLBACK**
#define **INVALID_HANDLE_VALUE** ((void*)(-1))
#define **INFINITE** 0xFFFFFFFF
#define **VOID** void
#define **__readableTo**(extent)
#define **__nullterminated** __readableTo(sentinel(0))
#define **wsprintf** sprintf
#define **DECLARE_HANDLE**(name) struct name##__ { int unused; }; typedef struct name##__ *name

Typedefs

typedef int **INT**
typedef unsigned int **UINT**
typedef unsigned int * **PUINT**
typedef char **CHAR**
typedef short **SHORT**
typedef long **LONG**
typedef SHORT * **PSHORT**

typedef LONG * **PLONG**
typedef unsigned long **ULONG**
typedef ULONG * **PULONG**
typedef unsigned short **USHORT**
typedef USHORT * **PUSHORT**
typedef unsigned char **UCHAR**
typedef UCHAR * **PUCHAR**
typedef unsigned long **DWORD**
typedef unsigned char **BYTE**
typedef unsigned short **WORD**
typedef float **FLOAT**
typedef FLOAT * **PFLOAT**
typedef BOOL * **PBOOL**
typedef BOOL * **LPBOOL**
typedef BYTE * **PBYTE**
typedef BYTE * **LPBYTE**
typedef int * **PINT**
typedef int * **LPINT**
typedef WORD * **PWORD**
typedef WORD * **LPWORD**
typedef long * **LPLONG**
typedef DWORD * **PDWORD**
typedef DWORD * **LPDWORD**
typedef void * **PVOID**
typedef void * **LPVOID**
typedef const void * **LPCVOID**
typedef wchar_t **WCHAR**
typedef size_t **SIZE_T**
typedef DWORD **COLORREF**
typedef DWORD * **LPCOLORREF**
typedef CHAR * **PCHAR**
typedef CHAR * **LPCH**
typedef CHAR * **PCH**
typedef CONST CHAR * **LPCCH**
typedef CONST CHAR * **PCCH**
typedef CHAR * **NPSTR**
typedef CHAR * **LPSTR**
typedef CHAR * **PSTR**
typedef PSTR * **PZPSTR**
typedef CONST PSTR * **PCZPSTR**
typedef CONST CHAR * **LPCSTR**
typedef CONST CHAR * **PCSTR**
typedef PCSTR * **PZPCSTR**
typedef void * **HGDIOBJ**
typedef int **INT_PTR**
typedef int * **PINT_PTR**
typedef unsigned int **UINT_PTR**
typedef unsigned int * **PUINT_PTR**

typedef long **LONG_PTR**
typedef long * **PLONG_PTR**
typedef unsigned long **ULONG_PTR**
typedef unsigned long * **PULONG_PTR**
typedef unsigned long **ULONGLONG**
typedef char **CCHAR**
typedef DWORD **LCID**
typedef PDWORD **PLCID**
typedef WORD **LANGID**
typedef long **LONGLONG**
typedef WORD **ATOM**
typedef pthread_mutex_t **CRITICAL_SECTION**
typedef CRITICAL_SECTION * **LPCRITICAL_SECTION**
typedef __nullterminated WCHAR * **NWPSTR**
typedef __nullterminated WCHAR * **LPWSTR**
typedef __nullterminated WCHAR * **PWSTR**
typedef __nullterminated PWSTR * **PZPWSTR**
typedef __nullterminated CONST PWSTR * **PCZPWSTR**
typedef __nullterminated CONST WCHAR * **LPCWSTR**
typedef __nullterminated CONST WCHAR * **PCWSTR**
typedef __nullterminated PCWSTR * **PZPCWSTR**
typedef LPSTR **PTSTR**
typedef LPSTR **LPTSTR**
typedef LPSTR **PUTSTR**
typedef LPSTR **LPUTSTR**
typedef LPCSTR **PCTSTR**
typedef LPCSTR **LPCTSTR**
typedef LPCSTR **PCUTSTR**
typedef LPCSTR **LPCUTSTR**
typedef WCHAR * **LPWCH**
typedef WCHAR * **PWCHAR**
typedef struct _SYSTEMTIME **SYSTEMTIME**
typedef struct _SYSTEMTIME * **PSYSTEMTIME**
typedef struct _SYSTEMTIME * **LPSYSTEMTIME**

Detailed Description

This file define primary types to use.

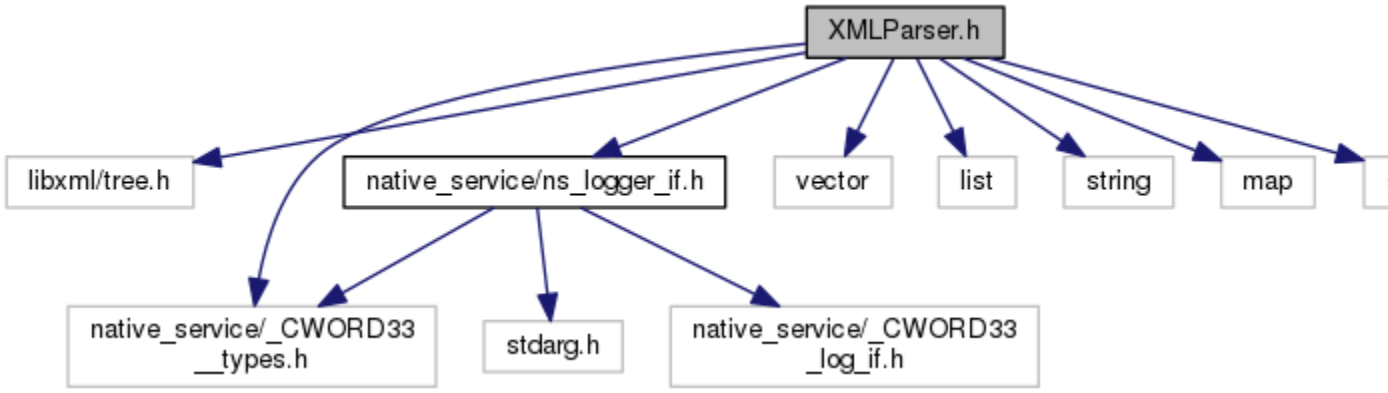
XMLParser.h File Reference

Xml parser used to extract and update data from Xml file.

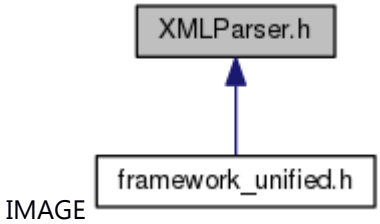
```
#include <libxml/tree.h>
#include <native_service/_CWORD33__types.h>
#include <native_service/ns_logger_if.h>
#include <vector>
#include <list>
#include <string>
#include <map>
#include <sstream>
```

Include dependency graph for XMLParser.h:

IMAGE



This graph shows which files directly or indirectly include this file:



IMAGE

Classes

class CTestCaseData

CTestCaseData : Class used to fill testcase data from XML. class XMLParser

---

Detailed Description

Xml parser used to extract and update data from Xml file.