MAGONIA API Specification (Distributed Processing Base)

Version 1.0

Nippon Telegraph and Telephone Corporation

February 19, 2015

INDEX

1. Introduction

1.1. About this document

This document contains opened API specifications for application of the VNF(distributed processing base) platform.

Incidentally, as a premise, distributed processing base platform works on high availability middleware(below is called server base), but API specifications of server base is out of this document.

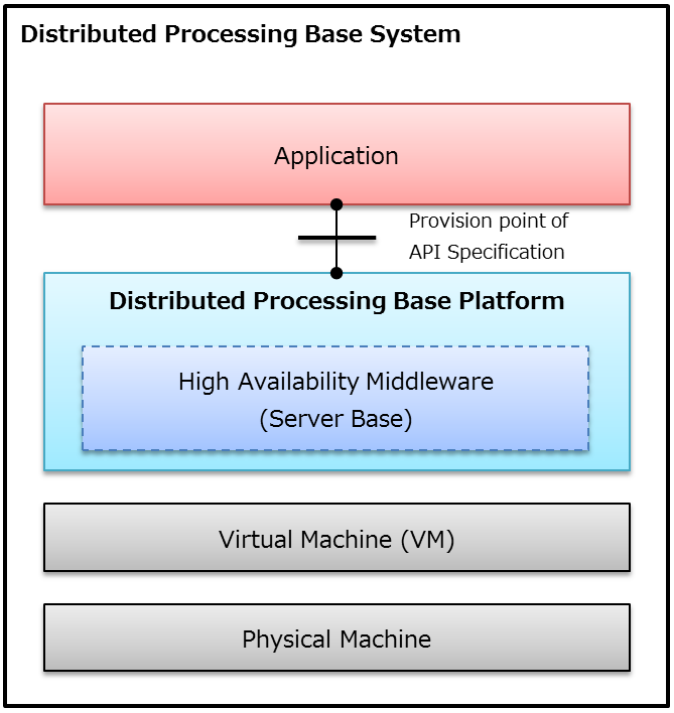


Fig. 1-1 The provision point of API specification of VNF(distributed processing base)

1.2. Constitution

In chapter 2 Namespace, name-space opened to application by distributed processing base platform and that which defined in each name-space (namespace, class, enumeration, variable and so on) are described.

In chapter3 Class, class opened to application by distributed processing base platform and that in each class (opened method, enumeration and so on) are described.

In chapter4 File, header file opened to application developer by distributed processing base platform and that defined in each header file (include files, namespace, class, functions of exception class, macro definition and so on) are described.

2. Namespace

2.1. xnode

Basic name-space of distributed processing base system

2.1.1. Namespace List

distributedprocessingbase	Name-space of distributed processing base.

2.1.2. Class List

Nothing

2.1.3. Enumeration

Nothing

2.1.4. Variable

Nothing

2.2. xnode::distributedprocessingbase

Namespace of distributed processing base

2.2.1. Namespace List

cmnp	Namespace of common parts *FB*
comc	Namespace of communication control *FB*
common	Namespace of common
dtct	Namespace of data container *FB*
gmbc	Namespace of member control *FB*
logm	Namespace of log management *FB*
msdp	Namespace of dispatch control *FB*
mspr	Namespace of execution control *FB*
odtm	Namespace of data original management *FB*

2.2.2. Class List

class DispatchKey	Class containing dispatch key data
class Socket	Socket class for concealing buffering processing

2.2.3. Enumeration

Nothing

2.2.4. Variable

Nothing

2.3. xnode::distributedprocessingbase::cmnp

Namespace of common parts *FB*

2.3.1. Namespace List

Nothing

2.3.2. Class List

class *BufferPool*	Buffer pool class
class *BufferPoolMng*	Buffer pool manager class
class *ThreadPool*	Thread pool class
class *ThreadPoolMng*	Thread pool manager class
class *TimerMng*	Time manager class

2.3.3. Enumeration

enum ***xnode::distributedprocessingbase::cmnp::ERegisterKind***

Each kind of registants

value of enumeration	
registerApl	APL
registerMdl	MDL
registerInvalid	void

2.3.4. Variable

Nothing

2.4. xnode::distributedprocessingbase::comc

Namespace of communication control *FB*

2.4.1. Namespace List

Nothing

2.4.2. Class List

class TcpConnectionMng	TCP connection management class
class UdpControl	UDP control class

2.4.3. Enumeration

Nothing

2.4.4. Variable

Nothing

2.5. xnode::distributedprocessingbase::common

Namespace of common

2.5.1. Namespace List

Nothing

2.5.2. Class List

class DistributedException	Common exception class for application thrown by distributed processing base.

2.5.3. Enumeration

enum ***xnode::distributedprocessingbase::common::FunctionBlock***

Enumeration showing *FB*

Value of enumeration	
fbApplication	Application
fbClusteringManager	Cluster configuration management
fbGenericMemberController	Member control
fbCoordinatorController	Coordinator control
fbMutualWatchKeeper	Other member monitoring
fbEventDistributor	Event control
fbMessageDispatcher	Dispatch control
fbMessageProcessor	Execution control
fbDataContainer	Data container
fbOriginalDataManager	Data original management
fbReplicaDataManager	Replicated data management
fbDataRelocator	Data relocating control
fbLockManager	Lock management
fbCommunicationCtrl	Communication control
fbCommonParts	Common parts

enum ***xnode::distributedprocessingbase::common::SvApiNumber***

Enumeration showing server base API numbers

Value of enumeration	
svApiNum_init_a_level	Level A initialization

Value of enumeration	
svApiNum_init_b_level	Level B initialization
svApiNum_init_c_level	Level C initialization
svApiNum_init_d_level	Level D initialization
svApiNum_init_e_level	Level E initialization
svApiNum_svRsmRequestStopServer	Request for stopping server
svApiNum_svRsmRegisterServerStopEntry	Registaring Callback when noticed stopping
svApiNum_svThrCreateThread	Creating thread
svApiNum_svThrStartThread	Starting thread
svApiNum_svThrCreateStartThread	Creating and starting thread
svApiNum_svThrExitDeleteThread	Thread finalization
svApiNum_svThrTerminateThread	Other thread forced termination
svApiNum_svThrGetSelfThreadId	Getting thread-ID
svApiNum_svThrCreateThreadPool	Creating thread pool
svApiNum_svThrDeleteThreadPool	Deleting thread pool
svApiNum_svThrStartPoolThread	Starting thread by thread pool
svApiNum_svThrGetThreadNumber	Getting the number of thread usable for thread pool
svApiNum_svThrGetThreadPoolId	Getting thread pool ID belonging self-thread
svApiNum_svThrRegisterMazeObservation	Self-thread maze observation registration
svApiNum_svThrDeleteMazeObservation	Self-thread maze observation release
svApiNum_svThrRegisterOneCallInitParameter	Resource registration for self-thread individual initial configuration
svApiNum_svThrReferOneCallInitParameter	Resource reference for self-thread individual initial configuration
svApiNum_svThrDeleteOneCallInitParameter	Resource registration-cancellation for self-thread individual initial configuration
svApiNum_svThrRegisterMazeParameter	Resorce registration for self-thread maze detection
svApiNum_svThrReferMazeParameter	自スレッドメーズ検出時用リソース参照 Resource reference for self-thread maze detection
svApiNum_svThrDeleteMazeParameter	Resource registration release for self-thread maze detection
svApiNum_svThrRegisterOneCallInitEscalation	Callback registragion when individual initial configuration escalationing
svApiNum_svThrRegisterMazeEscalation	Callback registragion when mazu escalationing
svApiNum_svTcpCreateSocket	TCP socket creating
svApiNum_svTcpBindSocket	TCP socket bind
svApiNum_svTcpConnectSocket	TCP socket connection
svApiNum_svTcpListenSocket	Waiting for connecting TCP socket
svApiNum_svTcpAcceptSocket	TCP connetion acception
svApiNum_svTcpSend	TCP transmission(simple)
svApiNum_svTcpSendMulti	TCP transmission(plural)
svApiNum_svTcpReceive	TCP reception
svApiNum_svTcpShuntDownSocket	TCP shutdown

Value of enumeration	
svApiNum_svTcpCloseSocket	TCP close
svApiNum_svTcpSelect	TCP selection
svApiNum_svTcpSetSocketOption	TCP socket optional configuration
svApiNum_svTcpGetSocketOption	Getting TCP socket option
svApiNum_svUdpCreateSocket	Creating UDP socket
svApiNum_svUdpJoinSocketMulticast	UDP socket multicast JOIN
svApiNum_svUdpDropSocketMulticast	UDP socket multicast DROP
svApiNum_svUdpSetSocketMulticastIF	UDP multicast IF configuration
svApiNum_svUdpSendSocketSingle	UDP transmission(simple)
svApiNum_svUdpSendSocketMulti	UDP transmission(plural)
svApiNum_svUdpReceiveSeocket	UDP reception
svApiNum_svUdpReceiveSeocketAssist	UDP reception(including ancillary data)
svApiNum_svUdpSendSocketReplace	UDP transmission(source address replacement, simple)
svApiNum_svUdpCloseSocket	UDP close
svApiNum_svUdpSelectSocket	UDP selection
svApiNum_svUdpSetSocketOption	UDP socket optional configuration
svApiNum_svUdpGetSocketOption	Getting UDP socket option
svApiNum_svPhcRegisterHealthCheck	Health check registration
svApiNum_svPhcStartHealthCheck	Starting health check
svApiNum_svPhcStopHealthCheck	Stopping health check
svApiNum_svPhcDeleteHealthCheck	Deleting health check
svApiNum_svTmrRegisterTimer	Timer registration
svApiNum_svTmrCancelTimer	Timer release
svApiNum_svTmrCancelAllTimer	All timer release
svApiNum_svTmrGetTimerList	Showing timer list
svApiNum_svTmrGetTimerInfo	Timer reference(specified ID)
svApiNum_svTmrGetPoolThreadNumber	Getting the number of thread pool available threads for timer
svApiNum_svClmRegisterHeartbeatInformation	Heartbeat transmission information registration
svApiNum_svClmStartHeartbeat	Starting heartbeat transmission
svApiNum_svClmStopHeartbeat	Stopping heartbeat transmission
svApiNum_svClmDeleteHeartbeatInformation	Deleting heartbeat transmission information
svApiNum_svClmSendHeartbeat	Heartbeat mmediate transmission
svApiNum_svClmRegisterHeartbeatCallback	Callback registration when receving heartbeat
svApiNum_svSrmGetCpuUtilizationInfo	Getting CPU utilization information
svApiNum_svSrmGetMemoryUtilizationInfo	Getting memory unilization information
svApiNum_svClrGetClockStatus	Getting clock status
svApiNum_svClrRegisterNoticeCallback	Callback registration for clock status notification
svApiNum_svCfmRegisterConfig	Config registration
svApiNum_svCfmGetIntegerValue	Config reference(integer)
svApiNum_svCfmGetLongValue	Config reference(Long)
svApiNum_svCfmGetStringValue	Config reference(String)

Value of enumeration	
svApiNum_svCfmGetBoolValue	Config reference(Bool)
svApiNum_svCfmGetIPValue	Config reference(IP address)
svApiNum_svCfmGetTimeValue	Config reference(Time)
svApiNum_svTcmStartSurveillance	Starting TCP connection monitoring
svApiNum_svTcmStopSurveillance	Stopping TCP connection monitoring
svApiNum_svTcmAcceptSurveillance	Starting TCP connection monitoring reception
svApiNum_svTcmRequestStopSurveillance	TCP connection monitoring stopping request

2.5.4. Variable

Nothing

2.6. xnode::distributedprocessingbase::dtct

Namespace of data container *FB*

2.6.1. Namespace List

Nothing

2.6.2. Class List

class Ddc	A class that represents management data of distributed processing base, *DistributedDataContainer*

2.6.3. Enumeration

Nothing

2.6.4. Variable

const std::string xnode::distributedprocessingbase::dtct::classNameBool = "+bool"[static]

Type of attribute data: boolean

const std::string xnode::distributedprocessingbase::dtct::classNameDouble = "+double"[static]

Type of attribute data: double

const std::string xnode::distributedprocessingbase::dtct::classNameFloat = "+float"[static]

Type of attribute data: float

const std::string xnode::distributedprocessingbase::dtct::classNameInt = "+int"[static]

Type of attribute data: int

const std::string xnode::distributedprocessingbase::dtct::classNameLong = "+long"[static]

Type of attribute data: long

const std::string xnode::distributedprocessingbase::dtct::classNameLongLong = "+longLong"[static]

Type of attribute data: long long

const std::string xnode::distributedprocessingbase::dtct::classNameString = "+string"[static]

Type of attribute data: std::string

const std::string xnode::distributedprocessingbase::dtct::classNameVecBool = "+vector_bool"[static]

Type of attribute data: std::vector<bool>

const std::string xnode::distributedprocessingbase::dtct::classNameVecDouble = "+vector_double"[static]

Type of attribute data: std::vector<double>

const std::string xnode::distributedprocessingbase::dtct::classNameVecFloat = "+vector_float"[static]

Type of attribute data: std::vector<float>

const std::string xnode::distributedprocessingbase::dtct::classNameVecInt = "+vector_int"[static]

Type of attribute data: std::vector<int>

const std::string xnode::distributedprocessingbase::dtct::classNameVecLong = "+vector_long"[static]

Type of attribute data: std::vector<long>

const std::string xnode::distributedprocessingbase::dtct::classNameVecLongLong = "+vector_longLong"[static]

Type of attribute data: std::vector<long long>

const std::string xnode::distributedprocessingbase::dtct::classNameVecString = "+vector_string"[static]

Type of attribute data: std::vector<std::string>

2.7. xnode::distributedprocessingbase::gmbc

Namespace of member control *FB*

2.7.1. Namespace List

Nothing

2.7.2. Class List

class MemberControlInterface	An interface class for member control *FB APL*.

2.7.3. Enumeration

Nothing

2.7.4. Variable

Nothing

2.8. xnode::distributedprocessingbase::logm

Namaspace of log management *FB*

2.8.1. Namespace List

Nothing

2.8.2. Class List

class LogMng	Log management class

2.8.3. Enumeration

*enum **xnode::distributedprocessingbase::logm::ELogApiKind***

API classification enumeration

Value of enumeration	
logApiApl	APL
logApiMdl	MDL

*enum **xnode::distributedprocessingbase::logm::ELogAplTrcLvl***

APL trace level enumeration

Value of enumeration	
logAplTrcLvl1	Level1
logAplTrcLvl2	Level2

*enum **xnode::distributedprocessingbase::logm::ELogDbgLvl***

Debug trace level enumeration

Value of enumeration	
logDbgLvl1	Level1(STlevel)
logDbgLvl2	Level2(ITlevel)
logDbgLvl3	Level3(CTlevel)

*enum **xnode::distributedprocessingbase::logm::ELogDirection***

Transmission and reception (translocation) identifier enumeration

Enumeration value	
logDirectRcv	Reception data
logDirectSnd	Transmission data
logDirectFwd	Translocation data ※out of use in APL

enum xnode::distributedprocessingbase::logm::ELogEvtKind

Event identifier enumeration

Value of enumeration	
logEvtCri	Critical Error
logEvtErr	Error
logEvtWarn	Warning
logEvtInfo	Information
logEvtDbg	Debug

enum xnode::distributedprocessingbase::logm::ELogPointKind

API starting/stopping identifier enumeration

Value of enumeration	
logPointApiSta	API start
logPointApiEnd	API stop

enum xnode::distributedprocessingbase::logm::ELogSigKind

Enumeration based on signals

Value of enumeration	
logSigMaint	Facing maintenance mechanism
logSigCntrl	Control event ※Out of use in APL
logSigDudp	Dsignal(UDP) ※Out of use in APL
logSigDtcp	Dsignal(TCP) ※Out of use in APL
logSigPudp	Psignal(UDP)
logSigPtcp	Psignal(TCP)

2.8.4. Variable

Nothing

2.9. xnode::distributedprocessingbase::msdp

Namespace of distribution control *FB*

2.9.1. Namespace List

Nothing

2.9.2. Class List

class DispatcherInterface	Interface class for APL of distribution control *FB*

2.9.3. Enumeration

Nothing

2.9.4. Variable

Nothing

2.10. xnode::distributedprocessingbase::mspr

Namespace of execution *FB*

2.10.1. Namespace List

Nothing

2.10.2. Class List

class IAsyncTaskListener	Listener base class for asynchronous task execution
class IApplicationListener	Listener base class for D- and PS-APL processing
class IFactoryListener	Listener base class for creating
class IDataListener	Listener base class for DDC data APL processing
class DefaultDataListener	Default implementation class of IDataListener
class ProcessorInterface	Interface class for APL of execution control *FB*

2.10.3. Enumeration

Nothing

2.10.4. Variable

Nothing

2.11. xnode::distributedprocessingbase::odtm

Namespace of original data management *FB*

2.11.1. Namespace List

Nothing

2.11.2. Class List

class LocalDdcIdIterator	Iterator control class for DDCID acquisition
class OriginalDataInterface	Interface class for APL of original data management *FB.*

2.11.3. Enumeration

enum xnode::distributedprocessingbase::odtm::ReplicationMode

Replication mode

Value of enumeration	
async	Asynchronous mode (It responds regardless of success in copying)
sync1Phase	1 phase synchronous mode (It responds after all copying are successful. If one of them is failure, “unconformity” is responded.)
sync2Phase	2 phase synchronous mode (It responds after all copying are successful. If one of them is failure, all of them are rewinded.)

2.11.4. Variable

Nothing

3. Class

3.1. xnode::distributedprocessingbase::cmnp::BufferPool

BufferPoolClass

”Defined in “BufferPool.hpp”

3.1.1. Public Method List

*void alloc (void *&aBufferPtr) throw (common::DistributedException)*
*void free (void *aBufferPtr) throw (common::DistributedException)*

3.1.2. Static Public Method List

Nothing

3.1.3. Enumeration

Nothing

3.1.4. Method

void xnode::distributedprocessingbase::cmnp::BufferPool::alloc (void *& aBufferPtr) throw common::DistributedException)

Buffer-acquisition

Acquire empty buffer of one block from bufferpool generated by bufferpool manager class.

Argument:

aBufferPtr	OUT:buffer pointer

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

: Supplementary explanation

- If there is no empty buffer, exception (internalError) is thrown.

void xnode::distributedprocessingbase::cmnp::BufferPool::free (void * aBufferPtr) throw common::DistributedException)

Buffer release

Buffer of one block acquired by alloc is released.

Argument:

aBufferPtr	IN:Buffer pointer

Exception:

invalidArgument	Parameter abnormality
internalError	Processing abnormality occurred

Return value:

Nothing

: Supplementary explanation

- If Buffer out of management release is attempted, exception(internalError) is thrown.

3.2. xnode::distributedprocessingbase::cmnp::BufferPoolMng

Bufferpool manager class defined in “BufferPoolMng.hpp”.

3.2.1. Public Method List

*void create (int aBufferNum, size_t aBufferSize, const void *aClassOidPtr, BufferPool *&aBufferPoolPtr) throw (common::DistributedException)*
*void release (BufferPool *aBufferPoolPtr) throw (common::DistributedException)*

3.2.2. Static Public Method List

static BufferPoolMng & getInstance (void)

3.2.3. Enumeration

Nothing

3.2.4. Method

void xnode::distributedprocessingbase::cmnp::BufferPoolMng::create (int aBufferNum, size_t aBufferSize, const void * aClassOidPtr, BufferPool *& aBufferPoolPtr) throw common::DistributedException)

Creating Bufferpool instance

Create and return bufferpool class instance, and manage in this class.

Argument:

aBufferNum	IN:the number of buffer block
aBufferSize	IN:size of buffer of one block
aClassOidPtr	IN:origin class instance
aBufferPoolPtr	OUT:bufferpool class instance

Exception:

invalidArgument	Parameter abnormality
internalError	Processing abnormality occurred

Return value:

Nothing

Supplementary explanation:

- Reserve memory area (“the number of block” x ”1 block size”) at one time, and manege with bufferpool class.

Caution:

- Release with release method bufferclass instance generated with this method (delete command cannot be used).
- aClassOidPtris used as for fault analysis. (set this pointer)

*static **BufferPoolMng**& xnode::distributedprocessingbase::cmnp::BufferPoolMng::getInstance (void)[static]*

Bufferpool manager class instance acquisition

Acquire singular bufferpool manager class instance.

Argument:

Nothing

Return value:

Bufferpool manager class instance

*void xnode::distributedprocessingbase::cmnp::BufferPoolMng::release (**BufferPool** * aBufferPoolPtr) throw **common::DistributedException**)*

Bufferpool instance release

Release bufferpool class instance managed in this class.

Argument:

aBufferPoolPtr	IN:Bufferpool class instance

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing.

Supplementary explanation:

- If bufferpool class instance out of management is designated, exception (internalError) is thrown.

3.3. xnode::distributedprocessingbase::dtct::Ddc

Class showing DistributedDataContainer, that is managaement data of distributed processing base defined in “datacontainer.h”

3.3.1. Public Method List

virtual void getDispatchKey (DispatchKey &dispatchKey)=0
virtual void getKeyList (std::vector< std::string > &keyList)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, int value)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, long value)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, long long value)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, float value)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, double value)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, bool value)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, const std::string &value, bool checkBinaryDiff=false)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, const std::vector< int > &value, bool checkBinaryDiff=false)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, const std::vector< long > &value, bool checkBinaryDiff=false)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, const std::vector< long long > &value, bool checkBinaryDiff=false)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, const std::vector< float > &value, bool checkBinaryDiff=false)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, const std::vector< double > &value, bool checkBinaryDiff=false)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, const std::vector< bool > &value, bool checkBinaryDiff=false)=0 throw (common::DistributedException)
virtual void setAttribute (const std::string &key, const std::vector< std::string > &value, bool checkBinaryDiff=false)=0 throw (common::DistributedException)
virtual void setSerializedData (const std::string &key, const std::string &value, const std::string &className, bool checkBinaryDiff=false)=0 throw (common::DistributedException)
virtual void removeAttribute (const std::string &key)=0 throw (common::DistributedException)
virtual void getClassName (const std::string &key, std::string &className)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, int &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, long &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, long long &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, float &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, double &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, bool &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, std::string &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, std::vector< int > &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, std::vector< long > &value)=0 throw (common::DistributedException)

virtual void getAttribute (const std::string &key, std::vector< long long > &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, std::vector< float > &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, std::vector< double > &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, std::vector< bool > &value)=0 throw (common::DistributedException)
virtual void getAttribute (const std::string &key, std::vector< std::string > &value)=0 throw (common::DistributedException)
virtual void getSerializedData (const std::string &key, std::string &value)=0 throw (common::DistributedException)
virtual void addOneTimeTask (std::string &taskId, const std::string &className, int delay, const std::string &extraData)=0 throw (common::DistributedException)
virtual void addCyclicTask (std::string &taskId, const std::string &className, int delay, bool fixedRate, const std::vector< int > &interval, const int repeat, const std::string &extraData)=0 throw (common::DistributedException)
virtual void removeTask (const std::string &taskId)=0 throw (common::DistributedException)
virtual void getTaskIdList (std::vector< std::string > &taskIdList)=0 throw (common::DistributedException)
virtual ~Ddc ()=0

3.3.2. Static Public Method List

Nothing

3.3.3. Enumeration

Nothing

3.3.4. Method

virtual xnode::distributedprocessingbase::dtct::Ddc::~Ddc ()[pure virtual]

Destructor

Argument:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::addCyclicTask (std::string & taskId, const std::string & className, int delay, bool fixedRate, const std::vector< int > & interval, const int repeat, const std::string & extraData) throw common::DistributedException) [pure virtual]

Asynchronous task (rotation) registration (relief)

Register asynchronous task executed cyclically and repeatedly.

Argument:

taskId	OUT:Asynchronous task ID (For storing return value)
className	IN:Listener class name for asynchronous task execution)
delay	IN:An execution standby time (in milliseconds) until first expiration.(0 means instantaneous execution)
fixedRate	IN:Operation mode (true:fixed frequency, false: fixed delay)
interval	IN:List of expiration interval. (in milliseconds)
repeat	IN:Repeat count of list of expiration interval. (“-1” means repeating infinitely)
extraData	IN:Additional information required for creating instance of task execution class for APL.

Exception:

invalidArgument	Parameter abnormality
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::addOneTimeTask (std::string & taskId, const std::string & className, int delay, const std::string & extraData) throw common::DistributedException) [pure virtual]

Asynchronous task (single) registration (relief)

Register asynchronous task executed only once.

Argument:

taskId	OUT:Asynchronous task ID(For storing return value)
className	IN:Listener class name for executing asynchronous task)
delay	IN: An execution standby time (in milliseconds) until first expiration.(0 means instantaneous execution)
extraData	IN:Additional information required for creating instance of task execution class for APL.

Exception:

invalidArgument	Parameter abnormality
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, int & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type)

Acquire an int value from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, long & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type)Acquire a long value from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, long long & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type)

Acquire a long long value from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, float & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type)

Acquire a float value form DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, double & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type)Acquire a double value from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, bool & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type)Acquire a bool value from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, std::string & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type)Acquire a string value from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, std::vector< int > & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type).

Acquire an int value from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, std::vector< long > & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type).

Acquire a long list from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, std::vector< long long > & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type) Acquire a long long list from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, std::vector< float > & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type).

Acquire a float list from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

.Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, std::vector< double > & value) throw common::DistributedException) [pure virtual]

Attribute acquisition Acquire a double list from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, std::vector< bool > & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type) Acquire a bool list from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getAttribute (const std::string & key, std::vector< std::string > & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (primitive type) Acquire a string list from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getClassName (const std::string & key, std::string & className) throw common::DistributedException) [pure virtual]

Holding data class name acquisition

Acquire class name when executing setAttribute/setSerializedData.

Argument:

key	IN:Acquisition key
className	OUT:Class name

Exception:

keyNotExist	Specified key registration does not exist.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getDispatchKey (DispatchKey & dispatchKey)[pure virtual]

.Dispatch key acquisition

Release a dispatch key storing in DDC.

Argument:

dispatchKey	OUT:For value release

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getKeyList (std::vector< std::string > & keyList) throw common::DistributedException) [pure virtual]

Key list acquisition

Release attribute key list storing in DDC.

Argument:

keyList	OUT:For value release

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

Supplementary explanation:

- If there is no attribute, release 0 lists.

virtual void xnode::distributedprocessingbase::dtct::Ddc::getSerializedData (const std::string & key, std::string & value) throw common::DistributedException) [pure virtual]

Attribute acquisition (Serialized data).

Acquire serialized data from DDC.

Argument:

key	IN:Acquisition key
value	OUT:Acquisition value

Exception:

keyNotExist	Specified key registration does not exist.
invalidType	Value type corresponding to specified key is invalid.
internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::getTaskIdList (std::vector< std::string > & taskIdList) throw common::DistributedException) [pure virtual]

Execution waiting asynchronous task list acquisition

Search execution waiting asynchronous task registerd in this DDC, and acquire the task name list.

Argument:

taskIdList	OUT:execution waiting task name list

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

- Supplementary explanation: If there is no asynchronous task, a list with element 0 is returned.

virtual void xnode::distributedprocessingbase::dtct::Ddc::removeAttribute (const std::string & key) throw common::DistributedException) [pure virtual]

Attribute delete

Delete attribute in DDC.

Argument:

key	IN:delete key

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

- Supplementary explanation:Even if there is no specified key, exit successfully.

virtual void xnode::distributedprocessingbase::dtct::Ddc::removeTask (const std::string & taskId) throw common::DistributedException) [pure virtual]

Asynchronous task registration release (persistent failover)

Release registered asynchronous task.

Argument:

taskId	IN:Asynchronous task ID delivered when registering asynchronous task

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

Caution:

- Task already executing cannot be canceled.
- Even if there is no specified key, exit successfully.

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, int value) throw common::DistributedException) [pure virtual]

Attribute setting (primitive type)

Set int data in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, long value) throw common::DistributedException) [pure virtual]

Attribute setting (primitive type) Set long data in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, long long value) throw common::DistributedException) [pure virtual]

Attribute setting (primitive type)Set long long data in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, float value) throw common::DistributedException) [pure virtual]

Attribute setting (primitive type)Set float data in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, double value) throw common::DistributedException) [pure virtual]

Attribute setting (primitive type)Set double data in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, bool value) throw common::DistributedException) [pure virtual]

Attribute setting (Primitive type)

Set bool data in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, const std::string & value, bool checkBinaryDiff = `false`) throw common::DistributedException) `[pure virtual]`

Attribute setting (Primitive type).

Set string data in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value
checkBinaryDiff	IN:Whether to use replication of a difference of binary data or not (true:enable, false:disable [default])

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, const std::vector< int > & value, bool checkBinaryDiff = `false`) throw common::DistributedException) `[pure virtual]`

Attribute setting (Primitive type).

Set an int list in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value
checkBinaryDiff	IN: Whether to use replication of a difference of binary data or not (true:enable, false:disable [default])

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, const std::vector< long > & value, bool checkBinaryDiff = `false`) throw common::DistributedException) `[pure virtual]`

Attribute setting (primitive type)

Set a long list in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value
checkBinaryDiff	IN: Whether to use replication of a difference of binary data or not (true:enable, false:disable [default])

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, const std::vector< long long > & value, bool checkBinaryDiff = `false`) throw common::DistributedException) `[pure virtual]`

Attribute setting (primitive type)

Set a long long list in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value
checkBinaryDiff	IN: Whether to use replication of a difference of binary data or not (true:enable, false:disable [default])

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, const std::vector< float > & value, bool checkBinaryDiff = `false`) throw common::DistributedException) `[pure virtual]`

Attribute setting (primitive type)

Set a float list in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value
checkBinaryDiff	IN: Whether to use replication of a difference of binary data or not (true:enable, false:disable [default])

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, const std::vector< double > & value, bool checkBinaryDiff = `false`) throw common::DistributedException) `[pure virtual]`

Attribute setting (primitive type)

Set a double list in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value
checkBinaryDiff	IN: Whether to use replication of a difference of binary data or not (true:enable, false:disable [default])

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, const std::vector< bool > & value, bool checkBinaryDiff = `false`) throw common::DistributedException) `[pure virtual]`

Attribute setting (primitive type)

Set a bool list in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value
checkBinaryDiff	IN: Whether to use replication of a difference of binary data or not (true:enable, false:disable [default])

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setAttribute (const std::string & key, const std::vector< std::string > & value, bool checkBinaryDiff = `false`) throw common::DistributedException) `[pure virtual]`

Attribute setting (primitive type)

Set a string list in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value
checkBinaryDiff	IN: Whether to use replication of a difference of binary data or not (true:enable, false:disable [default])

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual void xnode::distributedprocessingbase::dtct::Ddc::setSerializedData (const std::string & key, const std::string & value, const std::string & className, bool checkBinaryDiff = `false`) throw common::DistributedException) `[pure virtual]`

Attribute setting (Selialized data)

Set an APL object (selialized data) in DDC.

Argument:

key	IN:Registration key
value	IN:Registration value (Selialized data)
className	IN:Class name for identifying registered object.
checkBinaryDiff	IN: Whether to use replication of a difference of binary data or not (true:enable, false:disable [default])

Exception:

internalError	Processing abnormality occurred
invalidArgument	Parameter abnormality

Return value:

Nothing

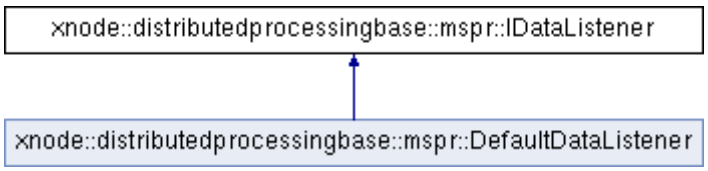
Supplement explanation:

- Class name registered in thin API can be acquired by getClassName().

3.4. xnode::distributedprocessingbase::mspr::DefaultDataListener

IDataListener default implementation class defined in “processor.h”

xnode::distributedprocessingbase::mspr::Succession graph to DefaultDataListener



3.4.1. Public Method List

virtual bool notifyExpired (dtct::Ddc &ddc) throw (common::DistributedException)
virtual bool notifyReplicaExpired (dtct::Ddc &ddc) throw (common::DistributedException)
*virtual bool notifyOneThreadInit (const std::vector< dtct::Ddc * > &ddc, std::map< std::string, void * > &resourceMap) throw (common::DistributedException)*
virtual void getOriginalDataStateNotify (bool &isNotify) throw (common::DistributedException)
virtual void notifyUpOriginalData (const std::string &ddcId) throw (common::DistributedException)
virtual void notifyTransOriginalData (dtct::Ddc &ddc, bool &isUpdated) throw (common::DistributedException)
virtual ~DefaultDataListener ()

3.4.2. Static Public Method List

Nothing

3.4.3. Enumeration

Nothing

3.4.4. Method

virtual xnode::distributedprocessingbase::mspr::DefaultDataListener::~DefaultDataListener ()[virtual]

Destructor

Argument:

Nothing

virtual void xnode::distributedprocessingbase::mspr::DefaultDataListener::getOriginalDataStateNotify (bool & isNotify) throw common::DistributedException) [virtual]

Acquire whether or not to notify original data statement

The default implementation sets"isNotify" "false"

Argument:

isNotify	OUT: Whether or not to notify original data statement (true:Notification is required, false:Notification is not required)

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

virtual bool xnode::distributedprocessingbase::mspr::DefaultDataListener::notifyExpired (dtct::Ddc & ddc) throw common::DistributedException) [virtual]

Floating data notification (origin)

The default implementation always returns true and deletes DDC.

Argument:

ddc	IN:DDC that can float

Exception:

internalError	Processing abnormality occurred

Return value:

Whether or not to delete DDC (true: required delete, false: not required delete)

*virtual bool xnode::distributedprocessingbase::mspr::DefaultDataListener::notifyOneThreadInit (const std::vector< dtct::Ddc * > & ddc, std::map< std::string, void * > & resourceMap) throw common::DistributedException) [virtual]*

Individual initial setting notification

The default implementation always returns true and deletes DDC.

Argument:

ddc	IN:DDC list in progress
resourceMap	IN:Resource registration map

Exception:

internalError	Processing abnormality occurred

Return value:

Whether or not to delete DDC (true: required delete, false: not required delete)

*virtual bool xnode::distributedprocessingbase::mspr::DefaultDataListener::notifyReplicaExpired (**dtct::Ddc** & ddc) throw **common::DistributedException**) [virtual]*

Floating data notification (replication)

The default implementation always returns true and deletes DDC.

Argument:

ddc	IN:DDC that can float

Exception:

internalError	Processing abnormality occurred

Return value:

Whether or not to delete DDC (true: required delete, false: not required delete)

*virtual void xnode::distributedprocessingbase::mspr::DefaultDataListener::notifyTransOriginalData (**dtct::Ddc** & ddc, bool & isUpdated) throw **common::DistributedException**) [virtual]*

Original data transfer notification

The default implementation does not do anything and always sets “isUpdated” false.

Argument:

ddc	IN:DDC replication object
isUpdated	OUT:Whether or not to update DDC (true: updated, false: not updated)

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

*virtual void xnode::distributedprocessingbase::mspr::DefaultDataListener::notifyUpOriginalData (const std::string & ddcId) throw **common::DistributedException**) [virtual]*

Original data promotion notification

The default implementation does not do anything.

Argument:

ddcId	IN:DDCID of original data

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

3.5. xnode::distributedprocessingbase::msdp::DispatcherInterface

Interface class for APL of dispatch control *FB* defined in “dispatcher.h”
”

3.5.1. Public Method List

void loggingRequest (DispatchKey &dispatchKey, std::string &message, long dataId) throw (common::DistributedException)

3.5.2. Static Public Method List

static DispatcherInterface & getInstance ()

3.5.3. Enumeration

Nothing

3.5.4. Method

static DispatcherInterface& xnode::distributedprocessingbase::msdp::DispatcherInterface::getInstance ()[static]

Acquire interface class instance of dispatch control *FB*

Argument:

Nothing

Return value:

Unique instance in this class

void xnode::distributedprocessingbase::msdp::DispatcherInterface::loggingRequest (DispatchKey & dispatchKey, std::string & message, long dataId) throw common::DistributedException)

Output request for logs for signal tracking

Signal log information (the outline is in accordance with APL) and dispatch key are passed, and the logs are output after adding information which server the signal is dispatched to to logs.

Argument:

dispatchKey	IN:Dispatch key
message	IN:Log output contents
dataId	IN:Log output destination ID

Exception:

internalError	Processing abnormality occurred
invalidArgument	Parameter abnormality

Return value:

Nothing

Supplement explanation:

- If this API is executed when processing PS, not do anything and respond.

Caution:

- Log output destination ID is not-supported parameter. Default ID should be “1”.

3.6. xnode::distributedprocessingbase::DispatchKey

Class retaining dispatch key data defined in "distributedprocessingbase.h"

.

3.6.1. Public Method List

DispatchKey ()
DispatchKey (std::string &ddcId, uint64_t hash, time_t creationTime, std::string &clusterId, std::string &fileVersion, bool hashPresent=true)
void setDdcId (const std::string &ddcId)
*const char * getDdcId ()*
void setHash (const uint64_t hash)
uint64_t getHash ()
bool isHashPresent ()
void setCreationTime (const time_t creationTime)
time_t getCreationTime ()
void setClusterId (const std::string &clusterId)
*const char * getClusterId ()*
void setFileVersion (const std::string &fileVersion)
*const char * getFileVersion ()*

3.6.2. Static Public Method List

Nothing

3.6.3. Enumeration

Nothing

3.6.4. Method

xnode::distributedprocessingbase::DispatchKey::DispatchKey ()

Default constructor

Argument:

Nothing

Supplement explanation:

- Each field is regarded as unspecified.

xnode::distributedprocessingbase::DispatchKey::DispatchKey (std::string & ddcId, uint64_t hash, time_t creationTime, std::string & clusterId, std::string & fileVersion, bool hashPresent = true)

Constructor

Argument:

ddcId	IN:DDCID(If this is unspecified, an empty string is specified)
hash	IN:Hash value(If this is unspecified, “hashPresent” is set to false.)
creationTime	IN:Session start time(If this cannot be acquired, the value of 0 is returned.)
clusterId	IN:Cluster identifier(If this cannot be acquired, an empty string is specified)
fileVersion	IN:File version(If this cannot be acquired, an empty string is specified)
hashPresent	IN:Hash value setting(true:specified, false:unspecified)

const char xnode::distributedprocessingbase::DispatchKey::getClusterId ()*

Cluster identifier acquisition

Argument:

Nothing

Return value:

Cluster identifier

time_t xnode::distributedprocessingbase::DispatchKey::getCreationTime ()

Session start time acquisition

Argument:

Nothing

Return value:

Session start time

const char xnode::distributedprocessingbase::DispatchKey::getDdcId ()*

DDCID acquisition

Argument:

Nothing

Return value:

DDCID.

const char xnode::distributedprocessingbase::DispatchKey::getFileVersion ()*

File version acquisition

Argument:

Nothing

Return value:

File version

uint64_t xnode::distributedprocessingbase::DispatchKey::getHash ()

Hash value acquisition

Argument:

Nothing

Return value:

Hash value

bool xnode::distributedprocessingbase::DispatchKey::isHashPresent ()

Hash value setting state acquisition

Argument:

Nothing

Return value:

Hash value setting state (true:setting, false:non-setting).

void xnode::distributedprocessingbase::DispatchKey::setClusterId (const std::string & clusterId)

Cluster identifier setting

Argument:

clusterId	IN:Cluster identifier

Return value:

Nothing

void xnode::distributedprocessingbase::DispatchKey::setCreationTime (const time_t creationTime)

Session star time setting

Argument:

creationTime	IN:Session start time

Return value:

Nothing

void xnode::distributedprocessingbase::DispatchKey::setDdcId (const std::string & ddcId)

DDCIDsetting

Argument:

ddcId	IN:DDCID

Return value:

Nothing

void xnode::distributedprocessingbase::DispatchKey::setFileVersion (const std::string & fileVersion)

File version setting

Argument:

fileVersion	IN:File version

Return value:

Nothing

void xnode::distributedprocessingbase::DispatchKey::setHash (const uint64_t hash)

Hash value setting

Argument:

hash	IN:Hash value

Return value:

Nothing

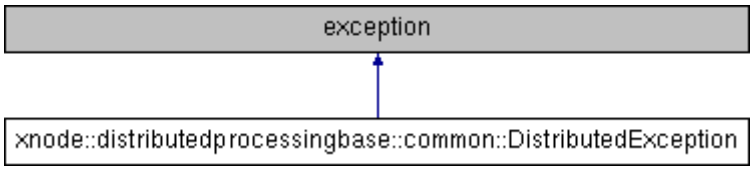
Supplement explanation:

- If this API is executed, “hashPresent” is changed to “true”.

3.7. xnode::distributedprocessingbase::common::DistributedException

Common exception class for APL thrown by distributed processing base,defined in "DistributedException.hpp"

xnode::distributedprocessingbase::common:: Succession graph to DistributedException



3.7.1. Public Method List

DistributedException (const enum CauseCode cause, const enum FunctionBlock fb, const std::string &message)
DistributedException (const enum CauseCode cause, const int detailCause1, const int detailCause2, const enum FunctionBlock fb, const std::string &message)
CauseCode getCause (void) const
FunctionBlock getFunctionBlock (void) const
*virtual const char * what (void) const throw ()*
void getDetailCause (int &detailCause1, int &detailCause2)
virtual ~DistributedException (void) throw ()

3.7.2. Static Public Method List

Nothing

3.7.3. Enumeration

enum ***xnode::distributedprocessingbase::common::DistributedException::CauseCode***

Exception error number for APL

Enumeration value	
invalidArgument	Parameter abnormality
alreadyExist	Data already exists
dataNotExist	Original data does not exist
dataRegisterFailed	Original data fails to be created
dataUpdateFailed	Original data fails to be updated
dataGetFailed	Original data fails to be acquired
dataRemoveFailed	Original data fails to be removed
replicationFailed	Replecation failure
lockTimeout	Time out of lock acquisition
notLocked	Target DDC isn't locked
taskRegisterFailed	Asynchronous task fails to be operated
timerRegisterFailed	Floating check timer fails to be operated
keyNotExist	Specified key registration does not exist
invalidType	Value type corresponding to specified key is invalid.

Enumeration value	
ioError	IO error
dispatchKeyNotFound	Dispatch key cannot be found
balancerNotFound	B address corresponding to specified port does not exist
invalidThread	Execution on those except distributed processing base
notStarted	Execution (before service starts/after service stops)
internalError	Process abnormality occurance
serverApiError	Error occurance in server base API
connectionNotExist	Specified connection registration does not exist

3.7.4. Method

xnode::distributedprocessingbase::common::DistributedException::DistributedException (const enum CauseCode cause, const enum FunctionBlock fb, const std::string & message)

Constructor

Argument:

cause	IN:Error number
fb	IN:FB of the source of exception
message	IN:Detailed error information

xnode::distributedprocessingbase::common::DistributedException::DistributedException (const enum CauseCode cause, const int detailCause1, const int detailCause2, const enum FunctionBlock fb, const std::string & message)

Constructor

Argument:

cause	IN:Error number
detailCause1	IN:Detailed error number 1
detailCause2	IN:Detailed error number 2
fb	IN:*FB* of the source of exception
message	IN:Detailed error information

virtual xnode::distributedprocessingbase::common::DistributedException::~DistributedException (void) throw) [virtual]

Destructor

Argument

Nothing

***CauseCode** xnode::distributedprocessingbase::common::DistributedException::getCause (void) const*

Error number acquisition

Argument:

Nothing

Return value:

Error number

void xnode::distributedprocessingbase::common::DistributedException::getDetailCause (int & detailCause1, int & detailCause2)

Detailed error number acquisition

Argument:

detailCause1	OUT:Detailed error number 1
detailCause2	OUT:Detailed error number 2

Return value:

Nothing

Supplement explanation:

- If detailed error number is non-used, “INVALID_DETAIL_CAUSE (-1)” is returned as detailed error number.

***FunctionBlock** xnode::distributedprocessingbase::common::DistributedException::getFunctionBlock (void) const*

***FB* of the source of exception**

Argument:

Nothing

Return value:

FB of the source of exception

virtual const char* xnode::distributedprocessingbase::common::DistributedException::what (void) const throw) [virtual]

Detailed error information acquisition

Argument:

Nothing

Return value:

Detailed error information

3.8. xnode::distributedprocessingbase::mspr::IApplicationListener

Listener base class for processing D and PS-APL, defined in "processor.h".

3.8.1. Public Method List

virtual void getDispatchKey (Socket &socket, DispatchKey &dispatchKey, bool &initialSignal, bool &readOnly, std::vector< DispatchKey > &list)=0 throw (common::DistributedException)
*virtual void getDispatchKey (const char *data, int size, const std::string &srcAddr, const int srcPort, DispatchKey &dispatchKey, bool &initialSignal, bool &readOnly, std::vector< DispatchKey > &list)=0 throw (common::DistributedException)*
virtual void doRead (Socket &socket)=0 throw (common::DistributedException)
virtual void dispatch (Socket &socket, DispatchKey &dispatchKey)=0 throw (common::DistributedException)
*virtual void dispatch (const char *data, int size, const std::string &srcAddr, const int srcPort, const int recvPort, DispatchKey &dispatchKey)=0 throw (common::DistributedException)*
virtual ~IApplicationListener ()=0

3.8.2. Static Public Method List

Nothing

3.8.3. Enumeration

Nothing

3.8.4. Method

virtual xnode::distributedprocessingbase::mspr::IApplicationListener::~IApplicationListener ()[pure virtual]

Destructor

Argumet:

Nothing

*virtual void xnode::distributedprocessingbase::mspr::IApplicationListener::dispatch (**Socket** & socket, **DispatchKey** & dispatchKey) throw **common::DistributedException**) [pure virtual]*

PS processing call (TCP)

Implement APL processing to PS

Argument:

socket	INReceiving side TCP signal
dispatchKey	IN:Dispatch key

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

Supplement explanation:

- If TCP is not used, dummy implementation can be done.

*virtual void xnode::distributedprocessingbase::mspr::IApplicationListener::dispatch (const char * data, int size, const std::string & srcAddr, const int srcPort, const int recvPort, **DispatchKey** & dispatchKey) throw **common::DistributedException**) [pure virtual]*

PS processing call (UDP)

Implement APL processing to PS

Argument:

data	IN:Reception UDP data
size	IN:Reception UDP data size
srcAddr	IN:Transmission source address
srcPort	IN:Transmission source port number
recvPort	IN:Reception (P) port number
dispatchKey	IN:Dispatch key

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

Supplement explanation:

- If UDP is not used, dummy implementation can be done.

virtual void xnode::distributedprocessingbase::mspr::IApplicationListener::doRead (Socket & socket) throw common::DistributedException) [pure virtual]

TCP signal readout

Readout from stream to a final edge of signal

Argument:

socket	IN:Reception side TCP signal

Exception:

ioError	IOerror

Return value:

Nothing

Supplement explanation

- If TCP is not used, dummy implementation can be done.

virtual void xnode::distributedprocessingbase::mspr::IApplicationListener::getDispatchKey (Socket & socket, DispatchKey & dispatchKey, bool & initialSignal, bool & readOnly, std::vector< DispatchKey > & list) throw common::DistributedException) [pure virtual]

Dispatch key acquisition (TCP)

Acquire or create dispatch key from signal

Argument:

socket	IN:Reception side TCP signal
dispatchKey	OUT:Dispatch key
initialSignal	OUT:Signal classification (true: initial signal, false: following signal)
readOnly	OUT:Operation classification (true:READONLY, false:READWRITE)
list	OUT:List of dispatch key of update object (If multiple DDC are not updated, do not set up. And this mustn't be overlapped to dispatchKey argument)

Exception:

ioError	IO—error
dispatchKeyNotFound	Dispatch key cannot be found

Return value:

Nothing

Supplement explanation:

- If TCP is not used, dummy implementation can be done.
- After creating instances of dispatch key and list at the distributed processing base, execute this API.If hash value of dispatch key or DDCID are not set, it is treated that DDC is not accessed when PS processing and lock et cetera is not done.
- If hash value of dispatch key is not set, sorting destination when D processing is the own server. However, if D is the only server, sorting destination is determined at random.

Caution:

- If the signal classification of some protocol cannot be discriminated, signal classification is specified to following singal.

*virtual void xnode::distributedprocessingbase::mspr::IApplicationListener::getDispatchKey (const char * data, int size, const std::string & srcAddr, const int srcPort, DispatchKey & dispatchKey, bool & initialSignal, bool & readOnly, std::vector< DispatchKey > & list) throw common::DistributedException) [pure virtual]*

Dispatch key acquisition (UDP)

Acquire or create dispatch key from signal

Argument:

data	IN:Reception UDP data
size	IN:Reception UDP data size
srcAddr	IN:Transmission source address
srcPort	IN:Port number
dispatchKey	OUT:Dispatch key
initialSignal	OUT:Signal classification (true:initial signal, false:following signal)
readOnly	OUT:Operation classification (true:READONLY, false:READWRITE)
list	OUT: List of dispatch key of update object (If multiple DDC are not updated, do not set up. And this mustn't be overlapped to dispatchKey argument)

Exception:

dispatchKeyNotFound	Dispatch key cannot be found

Return value:

Nothing

Supplement explanation:

- If UDP is not used, dummy implementation can be done.
- After creating instances of dispatch key and list at the distributed processing base, execute this API. Discarding including dispatch key registered in the list is done at distributed processing base
- If hash value of dispatch key or DDCID are not set, it is treated that DDC is not accessed when PS processing and lock et cetera is not done.
- If hash value of dispatch key is not set, sorting destination when D processing is the own server. However, if D is the only server, sorting destination is determined at random.

Caution:

- If the signal classification of some protocol cannot be discriminated, signal classification is specified to following singal.

3.9. xnode::distributedprocessingbase::mspr::IAsyncTaskListener

Listener base class for asynchronous task execution defined in "processor.h"

3.9.1. Public Method List

virtual void executeAsyncTask (const std::string &taskId, const std::string &ddcId, const std::string &extraData)=0 throw (common::DistributedException)
virtual ~IAsyncTaskListener ()=0

3.9.2. Static Public Method List

Nothing

3.9.3. Enumeration

Nothing

3.9.4. Method

virtual xnode::distributedprocessingbase::mspr::IAsyncTaskListener::~IAsyncTaskListener ()[pure virtual]

Destructor

Argument:

Nothing

virtual void xnode::distributedprocessingbase::mspr::IAsyncTaskListener::executeAsyncTask (const std::string & taskId, const std::string & ddcId, const std::string & extraData) throw common::DistributedException) [pure virtual]

Asynchronous task processing execution

Execute asynchronous task processing registered in DDC.

Argument:

taskId	IN:Asynchronous task ID
ddcId	IN:DDC in which asynchronous task is registered ID
extraData	IN:Object set when registering task

Exception:

internalError	Processing abnormality occurred

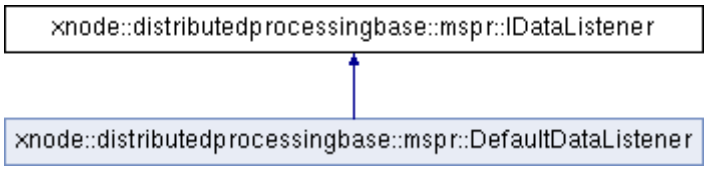
Return value:

Nothing

3.10. xnode::distributedprocessingbase::mspr::IDataListener

Listner base class for processing APL about DDC data, defined in"processor.h".

xnode::distributedprocessingbase::mspr::Sucession graph to IDataListener



3.10.1. Public Method List

virtual bool notifyExpired (dtct::Ddc &ddc)=0 throw (common::DistributedException)
virtual bool notifyReplicaExpired (dtct::Ddc &ddc)=0 throw (common::DistributedException)
*virtual bool notifyOneThreadInit (const std::vector< dtct::Ddc * > &ddc, std::map< std::string, void * > &resourceMap)=0 throw (common::DistributedException)*
virtual void getOriginalDataStateNotify (bool &isNotify)=0 throw (common::DistributedException)
virtual void notifyUpOriginalData (const std::string &ddcId)=0 throw (common::DistributedException)
virtual void notifyTransOriginalData (dtct::Ddc &ddc, bool &isUpdated)=0 throw (common::DistributedException)
virtual ~IDataListener ()=0

3.10.2. Static Public Method List

Nothing

3.10.3. Enumeration

Nothing

3.10.4. Method

virtual xnode::distributedprocessingbase::mspr::IDataListener::~IDataListener ()[pure virtual]

Destructor

Argument:

Nothing

virtual void xnode::distributedprocessingbase::mspr::IDataListener::getOriginalDataStateNotify (bool & isNotify) throw common::DistributedException) [pure virtual]

Acquire whether or not to notify original data statementWhen data gravitation occurs and data statement of own original data changes, acquire whether or not to need to notify it to APL.Argument:

isNotify	OUT:whether or not to notify original data statement (true:Notification is required, false:Notification is not required)

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

Supplement explanation:

- If "isNotify" is set true and this API is returned, distributed processing base calls original data promotion notification/original data transfer notification

virtual bool xnode::distributedprocessingbase::mspr::IDataListener::notifyExpired (dtct::Ddc & ddc) throw common::DistributedException) [pure virtual]

Floating data notification (original)

If data has not been accessed in the term of validity specified when registering in DDC, it is notified to APL that the data is floating data.

Argument:

ddc	IN:DDC that can float

Exception:

internalError	Processing abnormality occurred

Return value:

Wheter or not to delete DDC (true: required delete, false: non-required delete)

Supplement explanation:

- If this API returns true, DDC is deleted.
- If this API returns false, the term of validity starts to be monitored again from the time.
- Notified DDC data is read-only.

*virtual bool xnode::distributedprocessingbase::mspr::IDataListener::notifyOneThreadInit (const std::vector< dtct::Ddc * > & ddc, std::map< std::string, void * > & resourceMap) throw common::DistributedException) [pure virtual]*

Individual initial setting notification

Notify that individual initial setting occurs to APL

Argument:

ddc	IN:Processing DDC list
resourceMap	IN:Resource registrationMap

Exception:

internalError	Processing abnormality occurred

Return value:

Wheter or not to delete DDC (true: required delete, false: non-required delete)

Supplement explanation:

- Data that APL registerd is registerd in resource registration Map passed in the argument.

- If this API returns true, DDC is deleted.

virtual bool xnode::distributedprocessingbase::mspr::IDataListener::notifyReplicaExpired (dtct::Ddc & ddc) throw common::DistributedException) [pure virtual]

Floating data notification (replicated)

If replicated data has not been accessed in the term specified by "the terms of validity" plus "Config" those are specified in DDC registering, it is notified to APL that the data is floating data.Argument:

ddc	IN:DDC that can float

Exception:

internalError	Processing abnormality occurred

Return value:

Wheter or not to delete DDC (true: required delete, false: non-required delete)

Supplement explanation:

- If this API returns true, DDC is deleted.
- If this API returns false, the term of validity starts to be monitored again from the time. Notified DDC data is read-only.

virtual void xnode::distributedprocessingbase::mspr::IDataListener::notifyTransOriginalData (dtct::Ddc & ddc, bool & isUpdated) throw common::DistributedException) [pure virtual]

Original data transfer notification

Notify to APL that data gravitation occurs and own original data is transferred.

Argument:

ddc	IN:Copy of original data
isUpdated	OUT: Whether or not to update DDC (true: updated, false: not updated)

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

Supplement explanation:

- If "isUpdated" is set true, distributed processing base reflects updated contents in original data.
- If "isUpdated" is set false, updated contents are not reflected.
- Copy of notified original data is deleted in distributed processing base.

Caution:

- If copy of notified original data attempts to be updated, set "isUpdated" true.

virtual void xnode::distributedprocessingbase::mspr::IDataListener::notifyUpOriginalData (const std::string & ddcId) throw common::DistributedException) [pure virtual]

Original data promotion notification

Notify to APL that data gravitation occurs and itselve has owned original data.

Argument:

ddcId	IN:DDCID of original data

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

3.11. xnode::distributedprocessingbase::mspr::IFactoryListener

Listener base class for creating APL listener class defined in "processor.h"

3.11.1. Public Method List

*virtual void createApplication (const std::string &className, IApplicationListener *&applicationListener)=0 throw (common::DistributedException)*
*virtual void createAsyncTask (const std::string &className, IAsyncTaskListener *&asyncTask)=0 throw (common::DistributedException)*
virtual ~IFactoryListener ()=0

3.11.2. Static Public Method List

Nothing

3.11.3. Enumeration

Nothing

3.11.4. Method

virtual xnode::distributedprocessingbase::mspr::IFactoryListener::~IFactoryListener ()[pure virtual]

Destructor

Argument:

Nothing

virtual void xnode::distributedprocessingbase::mspr::IFactoryListener::createApplication (const std::string & className, IApplicationListener *& applicationListener) throw common::DistributedException) [pure virtual]

APL listener instance creation for signal processing

This API is executed in distributed processing base with class name specified in the config of distributed processing base.

Argument:

className	IN:Target class name to be created
applicationListener	OUT:APL listener instance for signal processing

Exception:

invalidArgument	Parameter abnormality

Return value:

Nothing

Supplement explanation:

- Created instance is deleted at the distributed processing base.

*virtual void xnode::distributedprocessingbase::mspr::IFactoryListener::createAsyncTask (const std::string & className, IAsyncTaskListener *& asyncTask) throw common::DistributedException) [pure virtual]*

Asynchronous task processing instance creation

When asynchronous task is registerd with setOneTimeTask()/setCycleTask(), class name specified when registering is passed to argument. This API creates asynchronous task listener instance corresponding to the class name and responds.Argument:

className	IN:Target class name to be created
asyncTask	OUT:Asynchronous task processing instance

Exception:

invalidArgument	Parameter abnormality

Return value:

Nothing

Supplement explanation:

- If asynchronous task is not used, dummy implementation can be done.
- If asynchronous task is persistent failover, this API is executed in persistent failover destination server.
- Created instance is deleted at the distributed processing base.

3.12. xnode::distributedprocessingbase::odtm::LocalDdcIdIterator

Iterator control class for DDCID acquisition defined in "originaldata.h"

3.12.1. Public Method List

virtual bool next (std::string &ddcId)=0

3.12.2. Static Public Method List

Nothing

3.12.3. Enumeration

Nothing

3.12.4. Method

xnode::distributedprocessingbase::odtm::LocalDdcIdIterator::LocalDdcIdIterator ()[protected]

Constructor

Argument:

Nothing

Exception:

internalError	Processing abnormality occurred

virtual xnode::distributedprocessingbase::odtm::LocalDdcIdIterator::~LocalDdcIdIterator ()[protected], [pure virtual]

Destructo

Argument:

Nothing

virtual bool xnode::distributedprocessingbase::odtm::LocalDdcIdIterator::next (std::string & ddcId)[pure virtual]

Sequential acquisition of local DDCID

Argument:

ddcId	OUT:DDCID acquired sequentially

Exception:

internalError	Processing abnormality occurred

Return value:

Wheter or not to acquire (true:succees to acquire, false:failure to acquire).

Caution:

- If return value is “false”, instance executing “next” cannot be acquired because the instance acquired by Sequential acquisition is final.

3.13. xnode::distributedprocessingbase::logm::LogMng

Log management class, defined in “LogMng.hpp”

3.13.1. Public Method List

*void signalTraceLog (enum ELogSigKind aLogSigKind, enum ELogDirection aLogSigDirect, const char *format,...)*
*void apiTraceLog (enum ELogApiKind aLogApiKind, enum ELogPointKind aLogPointKind, enum ELogAplTrcLvl aLogAplTrcLvl, const std::string &aFileName, const std::string &aFuncName, int aLineNum, const char *format,...)*
*void eventLog (enum ELogApiKind aLogApiKind, enum ELogEvtKind aLogEvtKind, enum ELogDbgLvl aLogDbgLvl, const std::string &aFileName, const std::string &aFuncName, int aLineNum, const char *format,...)*
bool isEventLogDbgLevel3Enable (enum ELogApiKind aLogApiKind)

3.13.2. Static Public Method List

static LogMng & getInstance (void)

3.13.3. Enumeration

Nothing

3.13.4. Method

void xnode::distributedprocessingbase::logm::LogMng::apiTraceLog (enum ELogApiKind aLogApiKind, enum ELogPointKind aLogPointKind, enum ELogAplTrcLvl aLogAplTrcLvl, const std::string & aFileName, const std::string & aFuncName, int aLineNum, const char * format, ...)

Request for collecting API trace log

APL requests for collecting log for APL API trace when API starts/ends.

Argument:

aLogApiKind	IN:APIclassification(APL/MDL)
aLogPointKind	IN:APIstarting/ending identifier(starting/ending)
aLogAplTrcLvl	IN:APLtrace level(Level1/Level2)
aFileName	IN:Log collection file name
aFuncName	IN:Log collection function name
aLineNum	IN:Log collection point (the number of lines in the files)
format,...	IN:Message itself (variable length argument)

Exception:

Nothing

Return value:

Nothing

Supplement explanation:

- Information LOG management outputs fixedly
  - Request time for log collection
- Log output destination/lotate file size are specified in Config.
- APL trace level
  - Collection point according to the level is set in APL.

Caution

- APL trace level
  - Level1 is enabled when APL API trace output level in Confing is Level1 or more than Level1.
  - Level2 is enabled when APL API trace output level in Confing is Level2 or more than Level2.

void xnode::distributedprocessingbase::logm::LogMng::eventLog (enum ELogApiKind aLogApiKind, enum ELogEvtKind aLogEvtKind, enum ELogDbgLvl aLogDbgLvl, const std::string & aFileName, const std::string & aFuncName, int aLineNum, const char * format, ...)

Request for event log collection

APL requests for collecting logs when notifying event necessary for Debug.

Argument:

aLogApiKind	IN:APIclassification(APL/MDL)
aLogEvtKind	IN:Event identifier(CRI/ERR/WARN/INFO/DBG)
aLogDbgLvl	IN:DBGtrace level(Level1/Level2/Level3)
aFileName	IN:Log collection file name
aFuncName	IN:Log collection function name
aLineNum	IN: Log collection point (the number of lines in the files)
format,...	IN:Message itself (variable length argument)

Exception:

Nothing

Return value:

Nothing

Supplement explanation:

- Information LOG management outputs fixedly
  - Request time for log collection
- Log output destination/lotate file size are specified in Config.
- DBG trace level
  - This is enabled only when event identifier is “Debug”.
  - Initial value (default) is Level1 (ST level)

Caution:

- DBG trace level
  - In level1, logs that have influences to performance are forbidden.
  - Level1 (ST level) is enabled when APL API Debug output level in Confing is Level1 or more than Level1.
  - Level2 (ST level) is enabled when APL API Debug output level in Confing is Level2 or more than Level2.
  - Level3 (ST level) is enabled when APL API Debug output level in Confing is Level3 or more than Level3.

static LogMng& xnode::distributedprocessingbase::logm::LogMng::getInstance (void)[static]

Log management class instance acquitision

Acquire unique instance in Log management class.

Argument:

Nothing

Exception:

Nothing

Return value:

aLogMngOid Log management class instance

bool xnode::distributedprocessingbase::logm::LogMng::isEventLogDbgLevel3Enable (enum ELogApiKind aLogApiKind)

Check whether event log DBG trace level3 is enabled or not

Check whetеher event log level3 of DBG trace is enabled or not.

Argument:

aLogApiKind	IN:APIclassification(APL/MDL)

Exception:

Nothing

Return value:

Event log DBG trace level3(true:enable, false:disable).

*void xnode::distributedprocessingbase::logm::LogMng::signalTraceLog (enum ELogSigKind aLogSigKind, enum ELogDirection aLogSigDirect, const char * format, ...)*

Request for signal trace log collection

Request for collecting trace log during signal transmission/reception (translocation)

Argument:

aLogSigKind	IN:Signal classification
aLogSigDirect	IN:Transmission/reception (translocation) identifier
format,...	IN:Messege itself (variable length argument)

Exception:

Nothing

Return value:

Nothing

Supplement explanation:

- Information LOG management outputs fixedly
  - Request time for log collection
- Log output destination/lotate file size are specified in Config.

3.14. xnode::distributedprocessingbase::gmbc::MemberControlInterface

Interface class for APL of member control *FB*, defined in “membercontrol.h”

3.14.1. Public Method List

void getMembers (bool processor, std::vector< std::string > &addressList) throw (common::DistributedException)
void getBalancerAddress (int port, std::string &balancerAddress, int &balancerPort) throw (common::DistributedException)
void leaveCluster (const std::string &causeMessage, const std::string &requester)
*void setHealthCheck (int(*hc_entry)(void *))*
*void deleteHealthCheck (int(*entry)(void *))*

3.14.2. Static Public Method List

static MemberControlInterface & getInstance ()

3.14.3. Enumeration

Nothing

3.14.4. Method

void xnode::distributedprocessingbase::gmbc::MemberControlInterface::deleteHealthCheck (int(*)(void *) entry)

Callback function delete for health check

Release registration for callback function called in executing health check.

Argument:

entry	IN:Function pointer of callback for health check

Returun value:

Nothing

void xnode::distributedprocessingbase::gmbc::MemberControlInterface::getBalancerAddress (int port, std::string & balancerAddress, int & balancerPort) throw common::DistributedException)

B address acquisition

Acquire IP address/port open to outside.

Argument:

port	IN:Waiting port number (for PS)
balancerAddress	OUT:B address corresponding to each port
balancerPort	OUT:B port corresponding to each port

Exception:

balancerNotFound	B address corresponding to specified port does not exist

Return value:

Nothing

Caution:

- Acquire B IP address/port with this API every time using them, because they can be changed under operation.

static MemberControlInterface& xnode::distributedprocessingbase::gmbc::MemberControlInterface::getInstance ()[static]

Acquisition of interface class instance for member control *FB*

Argument:

Nothing

Exception:

Nothing

Return value:

Unique instance in this class

void xnode::distributedprocessingbase::gmbc::MemberControlInterface::getMembers (bool processor, std::vector< std::string > & addressList) throw common::DistributedException)

Cluster member list acquisition

Acquire information about current all cluster member.

Argument:

processor	IN:Acquired member (true: PS member, false: D member)
addressList	OUT:IP address list of cluster member (In IPv6, full name is returned, not abbreviation)

Exception:

notStarted	Execution before service starts/after stops

Return value:

Nothing

Supplement explanation:

- If “processor” is true, returned member is only target member to process signals. (Member of D only and member which is set maintenance decreasing flags)
- If “processor” is false, returned member is that of D only or DPS.

Caution:

- This API can be usable after notifying service start, and not be usable after service stop for maintenance decreasing.

void xnode::distributedprocessingbase::gmbc::MemberControlInterface::leaveCluster (const std::string & causeMessage, const std::string & requester)

Cluster forced separation

Forced separation (failure leave) occurs according to APL's decision.

Argument:

causeMessage	IN:Reason for forced separation
requester	IN:Information about requester

Return value:

Nothing

Supplement explanation:

- Infomation passed with argument is used when outputting logs and alarms.

*void xnode::distributedprocessingbase::gmbc::MemberControlInterface::setHealthCheck (int(*)(void *) hc_entry)*

Registration for callback function for health check

Callback function called in executing health check is registered

Argument:

hc_entry	IN:Function pointer of callback for health check

Return value:

Nothing

3.15. xnode::distributedprocessingbase::odtm::OriginalDataInterface

Interface class for APL of original data management *FB*, defined in “originaldata.h”

3.15.1. Public Method List

virtual void registerDDC (const Ddc &ddc, int replNum, enum ReplicationMode replMode, int expireTime, bool repl, bool consistency)=0 throw (DistributedException)
virtual void updateDDC (const Ddc &ddc, bool repl)=0 throw (DistributedException)
*virtual void getDDC (const std::string &ddcId, Ddc *&ddc)=0 throw (DistributedException)*
virtual void removeDDC (const std::string &ddcId)=0 throw (DistributedException)
*virtual void getOtherDDC (const DispatchKey &dispatchKey, Ddc *&ddc)=0 throw (DistributedException)*
*virtual void getLocalDDC (const std::string &ddcId, Ddc *&ddc)=0 throw (DistributedException)*
*virtual void createDDC (const DispatchKey &dispatchKey, Ddc *&ddc)=0 throw (DistributedException)*
virtual int getRecreateDataCount (const std::string &ddcId)=0 throw (DistributedException)
virtual void clearRecreateDataCount (const std::string &ddcId)=0 throw (DistributedException)
*virtual void createLocalDdcIdIterator (LocalDdcIdIterator *&iterator)=0 throw (DistributedException)*

3.15.2. Static Public Method List

static OriginalDataInterface & getInstance ()

3.15.3. Enumeration

Nothing

3.15.4. Method

xnode::distributedprocessingbase::odtm::OriginalDataInterface::OriginalDataInterface ()[protected]

Constructor

Argument:

Nothing

virtual xnode::distributedprocessingbase::odtm::OriginalDataInterface::~OriginalDataInterface ()[protected], [pure virtual]

Destructor

Argument:

Nothing

virtual void xnode::distributedprocessingbase::odtm::OriginalDataInterface::clearRecreateDataCount (const std::string & ddcId) throw DistributedException) [pure virtual]

Reset of the number of continuous persitent failover

The number of that replicated data is promoted to original data is reset by local gravitation (relocation after failure leave)

Argument:

ddcId	IN:DDCID

Exception:

dataNotExist	Original data does not exist.
notLocked	Target DDC isn't locked.
internalError	Processing abnormality occurred

Return value:

Nothing

Caution:

- APL needs to call this API if it is confirmed DDC is normal.

virtual void xnode::distributedprocessingbase::odtm::OriginalDataInterface::createDDC (const DispatchKey & dispatchKey, Ddc *& ddc) throw DistributedException) [pure virtual]

.Data creation

Create new DDC data.

Argument:

dispatchKey	IN:Dispatch key (passed by "dispatch()")
ddc	OUT:Created variable for storing DDC

Exception:

invalidArgument	Parameter abnormality
alreadyExist	Original data already exists.
dataGetFailed	Original data fails to be acquired (because of not acquiring "lock" .etc)
internalError	Processing abnormality occurred

Return value:

Nothing

Caution:

- Instances this API acquired need to be deleted at its caller if "registerDDC" isn't executed.

virtual void xnode::distributedprocessingbase::odtm::OriginalDataInterface::createLocalDdcIdIterator (LocalDdcIdIterator *& iterator) throw DistributedException) [pure virtual]

Creation of sequentially acquired instance of local DDCID

Acquire iterator instances for enumerating local data lists.

Argument:

iterator	OUT:Class instance for control of iterator acquiring DDCID

Exception:

internalError	Processing abnormality occurred

Return value:

Nothing

Caution:

- Instances this API acquired need to be deleted at its caller.

virtual void xnode::distributedprocessingbase::odtm::OriginalDataInterface::getDDC (const std::string & ddcId, Ddc *& ddc) throw DistributedException) [pure virtual]

Data acquisition

Acquire DDC stored in original data management.

Argument:

ddcId	IN:Target DDC ID to be acquired
ddc	OUT:Variable for storing acquired DDC

Exception:

dataNotExist	Original data does not exist.
dataGetFailed	Original data fails to be acquired (because of not acquiring “lock” .etc)
timerRegisterFailed	Floating check timer fails to be operated
internalError	Processing abnormality occurred

Return value:

Nothing

Caution:

- Instances this API acquired need to be deleted at its caller if “updateDDC” isn’t executed.

static OriginalDataInterface& xnode::distributedprocessingbase::odtm::OriginalDataInterface::getInstance ()[static]

Acquisition of class instances of original data *FB* interface

Argument:

Nothing

Exception:

Nothing

Return value:

Unique instance in this class

*virtual void xnode::distributedprocessingbase::odtm::OriginalDataInterface::getLocalDDC (const std::string & ddcId, Ddc *& ddc) throw DistributedException) [pure virtual]*

Local data reference

Acquire copy for reference of DDC member itself owns.

Argument:

ddcId	IN:Target DDC ID to be acquired
ddc	OUT:Variable for storing acquired DDC

Exception:

dataNotExist	Original data does not exist.
dataGetFailed	Original data fails to be acquired (because of lack of memory .etc)
internalError	Processing abnormality occurred

Return value:

Nothing

Caution:

- Instances this API acquired need to be deleted at its caller.
- Data this API acquires is read-only and not usable for initial registration/update/delete.

*virtual void xnode::distributedprocessingbase::odtm::OriginalDataInterface::getOtherDDC (const DispatchKey & dispatchKey, Ddc *& ddc) throw DistributedException) [pure virtual]*

Acquirsition of other session data

Acquire DDC of other sessions (except DDC specified by dispatch key).

Argument:

dispatchKey	IN:Dispatch key (DDCID and hash value are required parameter)
ddc	OUT:Variable for storing acquired DDC

Exception:

dataNotExist	Original data does not exist.
lockTimeout	Time out of lock acquisition
invalidThread	Execution on those except distributed processing base
dataGetFailed	Original data fails to be acquired (because of not acquiring “lock” .etc)
internalError	Processing abnormality occurred

Return value:

Nothing

Supplement explanation:

- Other session data is acquired after executing “lock”. However,”Dead lock” can occur in this API. Therefore, if time-out value is over that defined in “Config”, “Exception” is thrown.

Caution:

- Instances this API acquired need to be deleted at its caller if “updateDDC” isn’t executed.

virtual int xnode::distributedprocessingbase::odtm::OriginalDataInterface::getRecreateDataCount (const std::string & ddcId) throw DistributedException) [pure virtual]

Acquition of the number of continuous persitent failover

The number of that replicated data is promoted to original data is acquired by local gravitation (relocation after failure leave)

Argument:

ddcId	IN:DDCID

Exception:

dataNotExist	Original data does not exist.
notLocked	Target DDC isn't locked.
internalError	Processing abnormality occurred

Return value:

The number of persitent failover

Supplement explanation:

- If return value is more than 1, it is considered that a failure occurs and data inconsistency can occur.

virtual void xnode::distributedprocessingbase::odtm::OriginalDataInterface::registerDDC (const Ddc & ddc, int replNum, enum ReplicationMode replMode, int expireTime, bool repl, bool consistency) throw DistributedException) [pure virtual]

New data registration

Management of DDC passed with argument starts at original data management.

Argument:

ddc	IN:DDC registered newly
replNum	IN:The degree of data replication
replMode	IN:Replication synchronous mode
expireTime	IN: The term of validity of DDC. If data hasn't been accessed during the term of validity, it is notified to APL and the term is deleted. "0" means infinity (in milliseconds).
repl	IN:persitent failover flag (true:persitent failover required, false:not required)
consistency	IN:Whether or not to execute periodical consistency (true:periodical consistency required, false:not required)

Exception:

alreadyExist	Original data already exists.
dataRegisterFailed	Original data fails to be created(because of not acquiring lock, registrating DDC acquired by "getLocalDDC" .etc)
replicationFailed	Replication failed
invalidArgument	Parameter abnormality (e.g. the degree of replication is less than 0, the term of validity is minus .etc)
taskRegisterFailed	Asynchronous task fails to be operated
timerRegisterFailed	Floating check timer fails to be operated
internalError	Processing abnormality occurred

Return value:

Nothing

Caution:

- DDC instances registerd in this API mustn't be deleted.

virtual void xnode::distributedprocessingbase::odtm::OriginalDataInterface::removeDDC (const std::string & ddcId) throw DistributedException) [pure virtual]

Data removal

Remove DDC storing in original data management.

Argument:

ddcId	IN:Target DDC ID to be removed

Exception:

dataNotExist	Original data doesn't exist.
dataRemoveFailed	Original data fails to be removed
taskRegisterFailed	Asynchronous task fails to be operated
timerRegisterFailed	Floating check timer fails to be operated
internalError	Processing abnormality occurred

Return value:

Nothing

Supplement explanation:

- Data is removed in replication synchronous mode specified when creating data.

virtual void xnode::distributedprocessingbase::odtm::OriginalDataInterface::updateDDC (const Ddc & ddc, bool repl) throw DistributedException) [pure virtual]

Data update

Update original data under management according to DDC passed with argument.

Argument:

ddc	IN:Target DDC to be updated
repl	IN:Persistent failover flag (true:persisten failover required, false:not required)

Exception:

dataNotExist	Original data doesn't exist.
dataUpdateFailed	Original data fails to be updated (because of not acquiring lock, registrating DDC acquired by "getLocalDDC" .etc)
replicationFailed	Replication failed
taskRegisterFailed	Asynchronous task fails to be operated
timerRegisterFailed	Floating check timer fails to be operated
internalError	Processing abnormality occurred

Return value:

Nothing

Supplement explanation:

- Replication synchronous mode specified during creation is used.

Caution:

- DDC instances updated with this API mustn't be deleted.

3.16. xnode::distributedprocessingbase::mspr::ProcessorInterface

Interface class for APL of execution control *FB*, defined in “processor.h”.

3.16.1. Public Method List

*void setDataListener (IDataListener *listener) throw (common::DistributedException)*
*void setFactoryListener (IFactoryListener *listener) throw (common::DistributedException)*
void getDispatchMember (DispatchKey &dispatchKey, std::string &address) throw (common::DistributedException)
void addOneTimeTask (std::string &taskId, const std::string &ddcId, const std::string &className, int delay, const std::string &extraData) throw (common::DistributedException)
void addCyclicTask (std::string &taskId, const std::string &ddcId, const std::string &className, int delay, bool fixedRate, const std::vector< int > &interval, const int repeat, const std::string &extraData) throw (common::DistributedException)
void removeTask (const std::string &taskId) throw (common::DistributedException)
*void addResource (const std::string &key, void *value) throw (common::DistributedException)*
*void * getResource (const std::string &key) throw (common::DistributedException)*
*void * removeResource (const std::string &key) throw (common::DistributedException)*

3.16.2. Static Public Method List

static ProcessorInterface & getInstance ()

3.16.3. Enumeration

Nothing

3.16.4. Method

void xnode::distributedprocessingbase::mspr::ProcessorInterface::addCyclicTask (std::string & taskId, const std::string & ddcId, const std::string & className, int delay, bool fixedRate, const std::vector< int > & interval, const int repeat, const std::string & extraData) throw common::DistributedException)

Asynchronous task (rotation) registration (not persistent failover)

Register asynchronous task executed cyclically and repeatedly.

Argument:

taskId	OUT:Asynchronous task ID
ddcId	IN:Target DDC ID to be operated
className	IN:Listener class name for asynchronous task execution
delay	IN: An execution standby time (in milliseconds) until first expiration.(0 means instantaneous execution)
fixedRate	IN: Operation mode (true:fixed frequency, false: fixed delay)
interval	IN: List of expiration interval. (in milliseconds)
repeat	IN: Repeat count of list of expiration interval. (“-1” means repeating infinitely)
extraData	IN:Additional information required for creating instance of task execution class for APL.

Exception:

notStarted	Execution before service starts/after stops
invalidArgument	Parameter abnormality
internalError	Processing abnormality occurred

Return value:

Nothing

Caution:

- This API can register only DDC member itself owns. If other members’ DDC is specified, “getDDC()” returns “error” in executing a task. This API can be usable after notifying service starts, and not be usable after service stop for maintenance decreasing.

void xnode::distributedprocessingbase::mspr::ProcessorInterface::addOneTimeTask (std::string & taskId, const std::string & ddcId, const std::string & className, int delay, const std::string & extraData) throw common::DistributedException)

Asynchronous task (single) registration (not persistent failover)

Register asynchronous task executed only one time.

Argument:

taskId	OUT:Asynchronous task ID
ddcId	IN:Target DDC ID to be operated
className	IN:Listener class name for asynchronous task execution
delay	IN: An execution standby time (in milliseconds) until first expiration.(0 means instantaneous execution)
extraData	IN:Additional information required for creating instance of task execution class for APL.

Exception:

notStarted	Execution before service starts/after stops
invalidArgument	Parameter abnormality
internalError	Processing abnormality occurred

Return value:

Nothing

Caution:

- This API can register only DDC member itself owns. If other members' DDC is specified, "getDDC()" returns "error" in executing a task.
- This API can be usable after notifying service starts, and not be usable after service stop for maintenance decreasing.

void xnode::distributedprocessingbase::mspr::ProcessorInterface::addResource (const std::string & key, void * value) throw common::DistributedException)

Registration for resource for individual initial configuration

Argument:

key	IN:Registration key name
value	IN:Registration information pointer

Exception:

invalidThread	Execution on those except distributed processing base
internalError	Processing abnormaility occurance

Return value:

Nothing

Caution:

- Registrator is responsible for deleting data specified by registered pointer.

void xnode::distributedprocessingbase::mspr::ProcessorInterface::getDispatchMember (DispatchKey & dispatchKey, std::string & address) throw common::DistributedException)

Dispather member acquisition

Acquire member (dispatcher) information based on dispatch key.

Argument:

dispatchKey	IN:Dispatch key (hash value is required parameter)
address	OUT:Dispatcher member address

Exception:

invalidArgument	Parameter abnormality
notStarted	Execution before service starts/after stops
internalError	Processing abnormaility occurance

Return value:

Nothing

Caution:

- This API can be usable after notifying service starts, and not be usable after service stop for maintenance decreasing..

static ProcessorInterface& xnode::distributedprocessingbase::mspr::ProcessorInterface::getInstance ()[static]

Acquisition of interface class for execution control *FB*

Argument:

Nothing

Exception:

Nothing

Return value:

Unique instance in this class

void* xnode::distributedprocessingbase::mspr::ProcessorInterface::getResource (const std::string & key) throw common::DistributedException)

Acquisition of resource for individual initial configuration

Acquire information registered in “Registration for resource for individual initial configuration”

Argument:

key	IN:Registration key name

Exception:

invalidThread	Execution on those except distributed processing base
internalError	Processing abnormaility occurance

Return value:

Registration information pointer

Supplement explanation:

- If corresponding key doesn’t exist, “null” is returned.

void* xnode::distributedprocessingbase::mspr::ProcessorInterface::removeResource (const std::string & key) throw common::DistributedException)

Removal of resource for individual initial configration

Remove information registered in “Registration for resource for individual initial configuration”

Argument:

key	IN:Registration key name

Exception:

invalidThread	Execution on those except distributed processing base
internalError	Processing abnormaility occurance

Return value:

Removed registration information pointer

Supplement explanation:

- If corresponding key doesn’t exist, “null” is returned..

void xnode::distributedprocessingbase::mspr::ProcessorInterface::removeTask (const std::string & taskId) throw common::DistributedException)

Asynchronous task registration release (not persistent failover)

Cancel asynchronous task registerd already.

Argument:

taskId	IN:Asynchronous task ID delivered when registering asynchronous task

Exception:

notStarted	Execution before service starts/after stops
internalError	Processing abnormaility occurance

Return value:

Nothing

Caution:

- Tasks already executing cannot be canceled.
- This API can be usable after notifying service starts, and not be usable after service stop for maintenance decreasing..

void xnode::distributedprocessingbase::mspr::ProcessorInterface::setDataListener (IDataListener * listener) throw common::DistributedException)

Registration for listener for DDC data processing

Argument:

listener	IN:Listener for callback

Exception:

invalidArgument	Parameter abnormality

Return value:

Nothing

*void xnode::distributedprocessingbase::mspr::ProcessorInterface::setFactoryListener (**IFactoryListener** * listener) throw **common::DistributedException**)*

Registration for lisnter for creating APL listener class

Argument:

listener	IN:Listener for callback

Exception:

invalidArgument	Parameter abnormality

Return value:

Nothing

Caution:

- APL must execute this listener registration.

3.17. xnode::distributedprocessingbase::Socket

Socket class for concealing buffering processing defined in “distributedprocessingbase.h”.

3.17.1. Public Method List

*virtual int read (char *buffer, int count)=0 throw (common::DistributedException)*
virtual int getConnectionId ()=0
virtual ~Socket ()=0

3.17.2. Static Public Method List

Nothing

3.17.3. Enumeration

Nothing

3.17.4. Method

virtual xnode::distributedprocessingbase::Socket::~Socket ()[pure virtual]

Destractor

Argument:

Nothing

virtual int xnode::distributedprocessingbase::Socket::getConnectionId ()[pure virtual]

Acquisition of TCP connection ID

Argument:

Nothing

Return value:

TCP connection ID

Supplement explanation:

- TCP connection ID is specified in using API for TCP signal transmission.

virtual int xnode::distributedprocessingbase::Socket::read (char * buffer, int count) throw common::DistributedException) [pure virtual]

Reading from the socket

If data buffered in this instance exists, read data from buffer is returned. After this, data read from the socket is returned.

Argument:

buffer	OUT:Buffer storing read data
count	IN:The number of read bytes

Exception:

internalError	Processing abnormaility occurance
ioError	IOerror

Return value:

The number of read bytes (If connection is cut, "-1" is returned.)

Supplement explanation:

- If connection is cut, "-1" is returned.

3.18. xnode::distributedprocessingbase::comc::TcpConnectionMng

TCPconnection managemene class defined in “TcpConnectionMng.hpp”.”

3.18.1. Public Method List

virtual void getTcpConnectionInfo (int aTcpConnId, int &aSrcPort, std::string &aDstAddr, int &aDstPort, sv_socket_id_t &aSocketId)=0 throw (common::DistributedException)
virtual void connect (const std::string &aDstAddr, int aDstPort, const std::string &aSrcAddr, int aSrcPort, int &aTcpConnId)=0 throw (common::DistributedException)
virtual void disconnect (int aTcpConnId)=0 throw (common::DistributedException)

3.18.2. Static Public Method List

static TcpConnectionMng & getInstanceProc (void)

3.18.3. Enumeration

Nothing

3.18.4. Method

virtual void xnode::distributedprocessingbase::comc::TcpConnectionMng::connect (const std::string & aDstAddr, int aDstPort, const std::string & aSrcAddr, int aSrcPort, int & aTcpConnId) throw common::DistributedException) [pure virtual]

Request for connection

Absorb a series of work form creating TCP socket to connecting TCP connectin using server base API, and manage the TCP connection.

Argument:

aDstAddr	IN:Connection destination IP address
aDstPort	IN:Connection destination port number
aSrcAddr	IN:Connection source IP address
aSrcPort	IN:Connection source port number
aTcpConnId	OUT:TCPconnection ID

Exception:

invalidArgument	Parameter abnormality
serverApiError	Server base API error
internalError	Processing abnormaility occurance

Return value:

Nothing

Supplement explanation:

- If connection source port number is that of short-life port, “0” is defined.

virtual void xnode::distributedprocessingbase::comc::TcpConnectionMng::disconnect (int aTcpConnId) throw common::DistributedException) [pure virtual]

Request for disconnection

Close TCP socket connecting to input TCP connection ID, and remove disconnected TCP connection in management information.

Argument:

aTcpConnId	IN:TCPconnection ID

Exception:

invalidArgument	Parameter abnormality
serverApiError	Server base API error
connectionNotExist	Specified connection registration does not exist
internalError	Processing abnormaility occurance

Return value:

Nothing

static TcpConnectionMng& xnode::distributedprocessingbase::comc::TcpConnectionMng::getInstanceProc (void)[static]

Acquisition of instance of TCP connection management class (for P)

Acquire instance of TCP connection management class (for P)

Argument:

Nothing

Exception:

Nothing

Return value:

Instance of TCP connection management class (for P)

Supplement explanation:

- Reference of uniquie static instance in this class is returned with return value.

Caution:

- If API of TCP connection management class (for P) need to be used, acquire instances with this API, and call each API via the insatances.

virtual void xnode::distributedprocessingbase::comc::TcpConnectionMng::getTcpConnectionInfo (int aTcpConnId, int & aSrcPort, std::string & aDstAddr, int & aDstPort, sv_socket_id_t & aSocketId) throw common::DistributedException) [pure virtual]

Acquisition of TCP connection information

Return each information corresponding to input TCP connection ID

Argument:

aTcpConnId	IN:TCPconnectionID
aSrcPort	OUT:Connection source port number
aDstAddr	OUT:Connection destination IP address
aDstPort	OUT:Connection destination port number
aSocketId	OUT:SocketID

Exception:

invalidArgument	Parameter abnormality
connectionNotExist	Specified connection registration does not exist
internalError	Processing abnormaility occurance

Return value:

Nothing

3.19. xnode::distributedprocessingbase::cmnp::ThreadPool

Thread pool class defined in “ThreadPool.hpp”.

3.19.1. Public Method List

*void startThread (void *aThreadArgPtr, void *aOcInitResourcePtr, void *aMzInitResourcePtr, bool aQueueFlag) throw (common::DistributedException)*
void getAvailableThreadNum (int &aTotalNum, int &aAvailableNum) throw (common::DistributedException)

3.19.2. Static Public Method List

*static void registOneCallInitParameter (void *aParameterPtr) throw (common::DistributedException)*
static void deleteOneCallInitParameter (void) throw (common::DistributedException)
*static void getOneCallInitParameter (void *&aParameterPtr) throw (common::DistributedException)*
*static void registMazeParameter (void *aParameterPtr) throw (common::DistributedException)*
static void deleteMazeParameter (void) throw (common::DistributedException)
*static void getMazeParameter (void *&aParameterPtr) throw (common::DistributedException)*
static void registMazeObservation (sv_time_msec_t aObserveTime, sv_thr_maze_observe_kind_t aOvserveKind) throw (common::DistributedException)
static void deleteMazeObservation (void) throw (common::DistributedException)
static void getAvailableThreadNum (sv_thr_pool_id_t aThr_poolId, int &aTotalNum, int &aAvailableNum) throw (common::DistributedException)
static void getThreadId (sv_thr_id_t &aThreadId) throw (common::DistributedException)
static void getThreadPoolId (sv_thr_pool_id_t &aThreadPoolId) throw (common::DistributedException)
static void exitThread (void) throw (common::DistributedException)

3.19.3. Enumeration

Nothing

3.19.4. Method

static void xnode::distributedprocessingbase::cmnp::ThreadPool::deleteMazeObservation (void) throw common::DistributedException) [static]

Release of observing maze of Thread itself

Release maze observation registration executed with “registMazeObservation”.

Argument:

Nothing

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers “svThrDeleteMazeObservation()” of server base API.
- Server base API can be called instead of this API.
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::deleteMazeParameter (void) throw common::DistributedException) [static]

Release of resource registration (for self-thread maze detection)

Delete the argument pawed when execution entry of self-thread maze detection starts.

Argument:

Nothing

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers “svThrDeleteMazeObservation()” of server base API.
- Server base API can be called instead of this API.
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::deleteOneCallInitParameter (void) throw common::DistributedException) [static]

Release of resource registration (for individual initial setting of self-thread)

Delete variable passed when entry of individual initial setting of self-thread starts.

Argument:

Nothing

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers “svThrDeleteOneCallInitParameter()” of server base API.
- Server base API can be called instead of this API..
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::exitThread (void) throw common::DistributedException) [static]

Self-thread exit

Exit self-thread

Argument:

Nothing

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers “svThrExitDeleteThred()” of server base API.Server base API can be called instead of this API..
- If error is returned at server base API, exception (serverApiError) is thrown.

Caution:

- If callback is that in individual initial configuration occurring, thread need to be exited calling this API in maze occurring.
- However, if callback is that in thread starting entry, this API mustn’t be called.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::getAvailableThreadNum (sv_thr_pool_id_t aThr_poolId, int & aTotalNum, int & aAvailableNum) throw common::DistributedException) [static]

Acquisition of the number of empty thread in thread pool

Acquire the number of all threads and usable empty thread in thread pool.

Argument:

aThr_poolId	IN:Thread pool ID
aTotalNum	OUT:The number of all threads in self-thread pool
aAvailableNum	OUT:The number of empty threads in self-thread pool

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Using this method, the number of empty threads can be acquired with "svThrGetThreadNumber()" of server base API.
- Server base API can be called instead of this API.
- If error is returned at server base API, exception (serverApiError) is thrown.

void xnode::distributedprocessingbase::cmnp::ThreadPool::getAvailableThreadNum (int & aTotalNum, int & aAvailableNum) throw common::DistributedException)

Acquisition of the number of empty threads in self-thread pool

Acquire the number of all threads and available empty threads in self-thread pool.

Argument:

aTotalNum	OUT:The number of threads in self-thread pool
aAvailableNum	OUT:The number of empty thread in self-thread pool

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Using this method, the number of empty threads can be acquired with "svThrGetThreadNumber()" of server base API.
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::getMazeParameter (void *& aParameterPtr) throw common::DistributedException) [static]

Resource reference (for self-thread maze detection)

Refer to variables passed when individual initial setting entry of self-thread and execution entry in maze detection start.

Argument:

aParameterPtr	OUT:Resouce ponter (maze)

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to "Serve_ base_API_specification", because this method covers "svThrReferMazeParameter()" of server base API.
- Server base API can be called instead of this API..
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::getOneCallInitParameter (void *& aParameterPtr) throw common::DistributedException) [static]

Resource reference (for individual initial setting of self-thread)

Refer to variables passed when individual initial setting entry of self-thread starts.

Argument:

aParameterPtr	OUT:Resource pointer (individulal initial setting)

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to "Serve_ base_API_specification", because this method covers "svThrReferOneCallInitParameter()" of server base API.
- Server base API can be called instead of this API..
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::getThreadId (sv_thr_id_t & aThreadId) throw common::DistributedException) [static]

Self-thread ID acquisition

Acquire thread ID of self-thread

Argument:

aThreadId	OUT:Self-thread ID

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers “svThrGetSelfThreadId()” of server base API.
- Server base API can be called instead of this API..
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::getThreadPoolId (sv_thr_pool_id_t & aThreadPoolId) throw common::DistributedException) [static]

Acquisition of Self-thread pool ID

Acquire pool ID of thread pool self-thread belongs to.

Argument:

aThreadPoolId	OUT:Self-thread pool ID

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers “svThrGetThreadPoolId()” of server base API.
- Server base API can be called instead of this API..
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::registMazeObservation (sv_time_msec_t aObserveTime, sv_thr_maze_observe_kind_t aOvserveKind) throw common::DistributedException) [static]

Registration for observation of self-thread maze)

Start observation of maze to self-thread at a specified time

Argument:

aObserveTime	IN:Observation time (in milliseconds)
aOvserveKind	IN:Observatio classification (CPU time/real time)

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to "Serve_ base_API_specification", because this method covers "svThrRegesterMazeObservation()" of server base API.
- Server base API can be called instead of this API..
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::registMazeParameter (void * aParameterPtr) throw common::DistributedException) [static]

Resource resistration (for self-thread maze detection)

Register arguments passed when execution entry in self-thread maze detection starts.

Argument:

aParameterPtr	IN:Resource pointer (maze)

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to "Serve_ base_API_specification", because this method covers "svThrRegisterMazeParameter()" of server base API.
- Server base API can be called instead of this API..
- If error is returned at server base API, exception (serverApiError) is thrown.

static void xnode::distributedprocessingbase::cmnp::ThreadPool::registOneCallInitParameter (void * aParameterPtr) throw common::DistributedException) [static]

Resource registration (for individual initial settings of self-thread)

Register arguments passed when individual initial setting entry of self-thread starts.

Argument:

aParameterPtr	IN:Resource pointer (individual initial setting)

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to "Serve_ base_API_specification", because this method covers "svThrOneCallInitParameter()" of server base API.
- Server base API can be called instead of this API..
- If error is returned at server base API, exception (serverApiError) is thrown.

void xnode::distributedprocessingbase::cmnp::ThreadPool::startThread (void * aThreadArgPtr, void * aOcInitResourcePtr, void * aMzInitResourcePtr, bool aQueueFlag) throw common::DistributedException)

Thread starting

Start one thread in thread pool.

Argument:

aThreadArgPtr	IN:Taken over thread parameter
aOcInitResourcePtr	IN:Resource pointer (individual initial setting)
aMzInitResourcePtr	IN:Resource pointer (maze)
aQueueFlag	IN:Queuing when thread pool is exhausted (true: enabled, false: disabled)

Exception:

serverApiError	Server base API error
internalError	Processing abnormaility occurance

Return value:

Nothing

Supplement explanation:

- Queuing when thread pool is exhausted (aQueueFlag)
  - true:If no space is left in thread pool, execute queuing and return from method. Execute this method as soon as the space of the thread pool is available
  - false: If no space is left in thread pool, exception (serverApiError/SV_E_NOTHER) is thrown.
- Detail of this method is refered to "Serve_ base_API_specification", because this method covers "svThrStartPoolThread()" of server base API.
- If error is returned at server base API, exception (serverApiError) is thrown.

Caution:

- Calling from hosts is needless, because calling to exit a thread of server base API is executed in thread pool class after thread starting entry ends. However, in maze occurring, it needs to be called at user because thread-end isn't called in the class in individual initial setting occurring.

3.20. xnode::distributedprocessingbase::cmnp::ThreadPoolMng

Thread pool manager class defined in “ThreadPoolMng.hpp”.

3.20.1. Public Method List

*void create (int aThreadNum, sv_thr_attr_t *aAttributePtr, void *(*aThreadEntryPtr)(void *), void *(*aOcInitEntryPtr)(sv_thr_id_t, void *), void *(*aMzInitEntryPtr)(sv_thr_id_t, void *), sv_thr_pool_id_t &aThreadPoolId, ThreadPool *&aThreadPoolPtr) throw (common::DistributedException)*
*void release (ThreadPool *aThreadPoolPtr) throw (common::DistributedException)*

3.20.2. Static Public Method List

static ThreadPoolMng & getInstance (void)

3.20.3. Enumeration

Nothing

3.20.4. Method

void xnode::distributedprocessingbase::cmnp::ThreadPoolMng::create (int aThreadNum, sv_thr_attr_t * aAttributePtr, void *(*)(void *) aThreadEntryPtr, void *(*)(sv_thr_id_t, void *) aOcInitEntryPtr, void *(*)(sv_thr_id_t, void *) aMzInitEntryPtr, sv_thr_pool_id_t & aThreadPoolId, ThreadPool *& aThreadPoolPtr) throw common::DistributedException)

Creation of thread pool instances

Create and return instances of thread pool class, and manage them in this class.

Argument:

aThreadNum	IN:The number of threads
aAttributePtr	IN:Thread attribute
aThreadEntryPtr	IN:Thread starting entry address (CallBack)
aOcInitEntryPtr	IN:Entry address of individual initialization occurred (CallBack)
aMzInitEntryPtr	IN:Maze occurance entry address(CallBack)
aThreadPoolId	OUT:Thread pool ID
aThreadPoolPtr	OUT:Thread pool class instance

Exception:

serverApiError	Server base API error
internalError	Processing abnormaility occurance

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers creating thread pool (svThrCreateThreadPool) of server base API.
- Entry activation origin doesn’t have to care return values (void*) of thread activation entry, individual initial setting occurance entry, maze occurance entry (only has to return “null”).
- If error is returned at server base API, exception (serverApiError) is thrown.
- The definition of entry of each callback function is refered to “

static ThreadPoolMng& xnode::distributedprocessingbase::cmnp::ThreadPoolMng::getInstance (void)[static]

Acquisiton of thread pool manager class instances

Acquire unique class instance in thread pool manager class.

Argument:

Nothing

Exception:

Nothing

Return value:

Thread pool manager class instance

*void xnode::distributedprocessingbase::cmnp::ThreadPoolMng::release (ThreadPool * aThreadPoolPtr) throw common::DistributedException)*

Release of thread pool instance

Delete thread pool forcibly, and release instances of thread pool class managed in this class.

Argument:

aThreadPoolPtr	IN:Thread pool calss instance

Exception:

serverApiError	Server base API error
internalError	Processing abnormaility occurance

Return value:

Nothing

Supplement explanation:

- If thread pool class instances out of management are specified, exception (internalError) is thrown.
- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers deleting thread pool (svThrDeleteThreadPool) of server base API.
- If error is returned at server base API, exception (serverApiError) is thrown.

3.21. xnode::distributedprocessingbase::cmnp::TimerMng

Time manager class defined in “TimerMng.hpp”.

3.21.1. Public Method List

*void regist (enum ERegisterKind aRegisterKind, int aRegisterUid, void *(*aTimerEntryPtr)(sv_tmr_timer_id_t, sv_tmr_timer_name_t, void *, int, int), void *aEntryParamPtr, void *(*aOcInitEntryPtr)(sv_thr_id_t, void *), void *aOcArgPtr, void *(*aMzInitEntryPtr)(sv_thr_id_t, void *), void *aMzArgPtr, const sv_tmr_timer_attr_t &aAttributePtr, sv_tmr_timer_id_t &aTimerId) throw (common::DistributedException)*
void cancel (sv_tmr_timer_id_t aTimerId, bool aFlag) throw (common::DistributedException)
void cancelAll (bool aFlag) throw (common::DistributedException)
void getList (std::vector< sv_tmr_timer_info_t > &aTimerInfoList) throw (common::DistributedException)
void getTimerInfo (sv_tmr_timer_id_t aTimerId, sv_tmr_timer_info_t &aTimerInfo) throw (common::DistributedException)
void getTimePoolThreadNumber (int &aTotalNum, int &aAvailableNum) throw (common::DistributedException)

3.21.2. Static Public Method List

static TimerMng & getInstance (void)

3.21.3. Enumeration

Nothing

3.21.4. Method

void xnode::distributedprocessingbase::cmnp::TimerMng::cancel (sv_tmr_timer_id_t aTimerId, bool aFlag) throw common::DistributedException)

Timer release

Execute timer release corresponding to specified timer ID.

Argument:

aTimerId	IN:Timer ID
aFlag	IN:Forcible delete flag (true: enable, false: disable)

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Even if specified timer is running, it is possible to specify whether or not to stop the timer with forcible delete flag. “Timer is running” means that timer activation entry is being executed after the timer expires.
- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers timer release (svTmrCancelTimer) of server base API.
- If error is returned at server base API, exception (serverApiError) is thrown.

void xnode::distributedprocessingbase::cmnp::TimerMng::cancelAll (bool aFlag) throw common::DistributedException)

Timer all release

Relase all timer registered already.

Argument:

aFlag	IN:Forcible delete flag (true: enable, false:disable)

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers all timer release (svTmrCancelAllTimer) of server base API.
- If error is returned at server base API, exception (serverApiError) is thrown.

static TimerMng& xnode::distributedprocessingbase::cmnp::TimerMng::getInstance (void)[static]

Acquisition of timer manager class instance

Acquire unique instance of time manager class.

Argument:

Nothing

Exception:

Nothing

Return value:

Timer manager class instance

void xnode::distributedprocessingbase::cmnp::TimerMng::getList (std::vector< sv_tmr_timer_info_t > & aTimerInfoList) throw common::DistributedException)

Timer list acquisition

Acquire lists of timer information as many as the number of registered timers.

Argument:

aTimerInfoList	OUT:Timer information list

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers showing timer list (svTmrGetTimerList) of server base API.
- If error is returned at server base API, exception (serverApiError) is thrown.
- Storage region stored in timer list of server base API is released after this method ends.

void xnode::distributedprocessingbase::cmnp::TimerMng::getTimePoolThreadNumber (int & aTotalNum, int & aAvailableNum) throw common::DistributedException)

Acquisition of the number of available threads in thread pool for timer

Acquire the number of all threads and unused threads in thread pool for timer.

Argument:

aTotalNum	OUT:The number of all threads in thread pool for timer
aAvailableNum	OUT:The number of unused threads in thread pool for timer

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Acquire the number of all threads in thread pool reserved by timer and unused threads not put out at the time of publication of this API."The number of all threads" – "the number of unused threads" is the number of running timers.
- Detail of this method is refered to "Serve_ base_API_specification", because this method covers acquiring the number of capable thread pool for timer (svTmrGetPoolThreadNumber) of server base API.
- If error is returned at server base API, exception (serverApiError) is thrown.

void xnode::distributedprocessingbase::cmnp::TimerMng::getTimerInfo (sv_tmr_timer_id_t aTimerId, sv_tmr_timer_info_t & aTimerInfo) throw common::DistributedException)

Timer reference

Acquire information about timer corresponding to specified timer ID

Argument:

aTimerId	IN:Timer ID
aTimerInfo	OUT:Timer information

Exception:

serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Detail of this method is refered to "Serve_ base_API_specification", because this method wraps referring timer (ID specified) (svTmrGetTimerInfo) of server base API.
- If error is returned at server base API, exception (serverApiError) is thrown.
- This is set when timer reference (ID specified) (svTmrGetTimerInfo) is called.

void xnode::distributedprocessingbase::cmnp::TimerMng::regist (enum ERegisterKind aRegisterKind, int aRegisterUid, void *(*)(sv_tmr_timer_id_t, sv_tmr_timer_name_t, void *, int, int) aTimerEntryPtr, void * aEntryParamPtr, void *(*)(sv_thr_id_t, void *) aOcInitEntryPtr, void * aOcArgPtr, void *(*)(sv_thr_id_t, void *) aMzInitEntryPtr, void * aMzArgPtr, const sv_tmr_timer_attr_t & aAttributePtr, sv_tmr_timer_id_t & aTimerId) throw common::DistributedException)

Timer registration

At the specified time, put the thread out from thread pool for timer, and activate function of specified entry.Argument:

aRegisterKind	IN:Registrant classification(APL/MDL)
aRegisterUid	IN:Unique ID of each registrant
aTimerEntryPtr	IN:Timer activation entry address(Call Back)
aEntryParamPtr	IN:Timer activation entry parameter
aOcInitEntryPtr	IN:Individual initial setting entry address
aOcArgPtr	IN:Resource pointer passed to individual initial setting entry
aMzInitEntryPtr	IN:Activation entry address in maze occurance
aMzArgPtr	IN:Resource pointer passed to execution entry in maze detection
aAttributePtr	IN:Timer attribute
aTimerId	OUT:Timer ID

Exception:

invalidArgument	Parameter abnormality
serverApiError	Server base API error

Return value:

Nothing

Supplement explanation:

- Definition of tunction of timer activation entry address (Call Back)
  - void* (*aTimerEntryPtr)(*
    *sv_tmr_timer_id_t aTimerId, // IN:*Timer ID
    *sv_tmr_timer_name_t aTimerName, // IN:*Timer name
    void aEntryParam, // IN:Timer activation entry parameter
    int aLastFlag, // IN:Flag of last time
    int aSkipNum // IN:The number of skips
    );
- If “disable” is set by registants, Unique ID of each registant is “Don’t care”. Timer activation entry parameter can be set information for identifying which timer expires easily by the registant freely (e.g. register instances of class to be notified).Entry executed when individual initial setting/maze occurs in timer activation enty can be registered as individual initial setting entry/entry in maze occurance. This can be optional specifying “null”. If abnormality occurs in omitted timer, exit the current process.
- Registration/reference/release of resource for individual initial setting of timer activation entry and for mazu detection can be executed using API described below thread pool class provides.
  - Resource registration (for individual initial setting of self-thread)
  - Resource reference (for individual initial setting of self-thread)
  - Release of resource registration (for individual initial setting of self-thread)
  - Resource registration (in maze detection of self-thread)
  - Resource reference (in maze detection of self-thread)

- Release of resource registration (in maze detection of self-thread)

- Issuing “self-thread maze observation registration” and “self-thred maze observation release” provided by thread pool class in timer activation entry makes it possible to start/end maze observation of timer activation entry.
- In timer name notified in timer activation entry, OR of each registrant set in timer registration (aRegisterKind)(APL:0x00000000, MDL: 0x10000000) and unique ID of the registant (aRegisterUid)(0x04000000～0x0FFFFFFF) is set
- Entry activation source doesn’t have to care a return value of timer activation entry (void*) (only has to return “null”).
- Detail of this method is refered to “Serve_ base_API_specification”, because this method covers timer registration (svTmrRegisterTimer) of server base API.
- If error is returned at server base API, exception (serverApiError) is thrown.

Caution:

- About unique ID of each registrant
  If it is set except “disable”, set in a following range.
  0x04000000～0x0FFFFFFF

3.22. xnode::distributedprocessingbase::comc::UdpControl

UDPcommunication control class defined in “UdpControl.hpp”.

3.22.1. Public Method List

*virtual void send (const std::string &aDstAddr, int aDstPort, const std::string &aSrcAddr, int aSrcPort, void *aBufferPtr, size_t aSendSize)=0 throw (common::DistributedException)*

3.22.2. Static Public Method List

static UdpControl & getInstanceProc (void)

3.22.3. Enumeration

Nothing

3.22.4. Method

static UdpControl& xnode::distributedprocessingbase::comc::UdpControl::getInstanceProc (void)[static]

Acquisition of instance of UDP communication control class (for P)

Acquire instances of UDP communication control class (for P).

Argument:

Nothing

Exception:

Nothing

Return value:

Instance of UDP communication control class (for P)

Supplement explanation:

- Reference of uniquie static instance in this class is returned with return value.

Caution:

- If API of UDP connection management class is used, acquire instances with this API and call each API via the instances.

virtual void xnode::distributedprocessingbase::comc::UdpControl::send (const std::string & aDstAddr, int aDstPort, const std::string & aSrcAddr, int aSrcPort, void * aBufferPtr, size_t aSendSize) throw common::DistributedException) [pure virtual]

Transmission request

Execute UDP transmission using server base API.

Argument:

aDstAddr	IN:Transmission destination IP address
aDstPort	IN:Transmission destination port number
aSrcAddr	IN:Transmission source IP address
aSrcPort	IN:Transmission source port number
aBufferPtr	IN:Bufferpointer
aSendSize	IN:Transmission size

Exception:

invalidArgument	Parameter abnormality
serverApiError	Server base API error
internalError	Processing abnormaility occurance

Return value:

Nothing

Supplement explanation:

- Buffer region is defined as a region reserved/managed by calling object. Release after transmission is executed by calling object as necessary. (Buffer region is whichever stack/static region, heap region and region API reserved by Buffer acquires)

Caution:

- In PS-APL, set B's IP address/port number to transmission source IP address/transmission source port number.

4. File

4.1. aplinit.h

APL initialization/finalization function header.

4.1.1. Include File

```
#include <stdbool.h>
```

4.1.2. Function List

bool init_application (void)
bool start_application (void)
bool destroy_application (void)

4.1.3. Function

bool destroy_application (void)

Service stopping/expiration notification to APL

Return value:

Processing result (true: successful termination, false: unsuccessful)

bool init_application (void)

.Initialization notification to APL

Return value:

Processing result (true: successful termination, false: unsuccessful)

bool start_application (void)

Service start notification to APL

Return value:

Processing result (true: successful termination, false: unsuccessful)

4.1.4. Macro Definition

Nothing

4.2. BufferPool.hpp

Buffer pool class header

4.2.1. Include File

```
#include "DistributedException.hpp"
#include <boost/thread/thread.hpp>
```

4.2.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::cmnp	Namespace of common parts *FB.*

4.2.3. Class List

class xnode::distributedprocessingbase::cmnp::BufferPool	Buffer pool class

4.2.4. Macro Definition

Nothing

4.3. BufferPoolMng.hpp

Buffer pool manager class header

4.3.1. Include File

```
#include <sv_api.h>
#include "DistributedException.hpp"
#include "BufferPool.hpp"
```

4.3.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::cmnp	Namespace of common parts *FB.*

4.3.3. Class List

class xnode::distributedprocessingbase::cmnp::BufferPoolMng	Buffer pool manager class

4.3.4. Macro Definition

Nothing

4.4. datacontainer.h

Header of namespace of data container *FB*

4.4.1. Include File

```
#include "distributedprocessingbase.h"
```

4.4.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::dtct	Namespace of data container *FB*

4.4.3. Class List

class xnode::distributedprocessingbase::dtct::Ddc	Class showing DistributedDataContainer, that is managaement data of distributed processing base

4.4.4. Macro Definition

Nothing

4.5. dispatcher.h

Header of namespace of dispatch control *FB*

4.5.1. Include File

```
#include "distributedprocessingbase.h"
```

4.5.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::msdp	Namespace of dispatch control *FB*

4.5.3. Class List

class xnode::distributedprocessingbase::msdp::DispatcherInterface	Interface class for APL of dispatch control *FB*

4.5.4. Macro Definition

Nothing

4.6. DistributedException.hpp

Common exception class header for distributed processing base APL

4.6.1. Include File

```
#include <string>
#include <vector>
```

4.6.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::common	Namespace of common

4.6.3. Class List

class *xnode::distributedprocessingbase::common::DistributedException*	Common exception class for APL thrown by distributed processing base

4.6.4. Macro Definition

Nothing

4.7. distributedprocessingbase.h

Header of namespace of distributed processing base

4.7.1. Include File

```
#include <string>
#include <stdint.h>
#include <boost/serialization/export.hpp>
#include <boost/serialization/serialization.hpp>
#include <boost/serialization/string.hpp>
#include "DistributedException.hpp"
#include "LogMng.hpp"
```

4.7.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base

4.7.3. Class List

class xnode::distributedprocessingbase::DispatchKey	Class storing dispatch key data
class xnode::distributedprocessingbase::Socket	Socket class for concealing buffering processing

4.7.4. Macro Definition

Nothing

4.8. LogMng.hpp

LOG management class header

4.8.1. Include File

```
#include <string>
#include <map>
```

4.8.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::logm	Namespace of log management *FB*

4.8.3. Class List

class xnode::distributedprocessingbase::logm::LogMng	LOG management class

4.8.4. Macro Definition

#define LOGM_APL_EVT_CRI(format, ...)

Request for collecting event logs (for Critical Error)

APL requests for collecting log in requiring event occuarance as Critical Error

Supplement explanation:

- Log output when obstruction that can be a factor of system stopping is assumed.

#define LOGM_APL_EVT_DBG_LV1(format, ...)

Request for collecting event logs (for Debug Level 1[ST])

APL requests for collecting logs when notifying necessary event for Debug

Supplement explanation:

- Level 1 is assumed as developer's optional log output for ST

Caution:

- Logs that have influences to performance are forbidden
- This is enabled when APL Debug output level of Config is Level 1 or more than Level 1.

#define LOGM_APL_EVT_DBG_LV2(format, ...)

Request for collecting event logs (for Debug Level 2[IT])

APL requests for collecting logs when notifying necessary event for Debug

Supplement explanation:

- Level 2 is assumed as developer's optional log output for IT.

Caution:

- This is enabled when APL Debug output level of Config is Level 2 or more than Level 2.

#define LOGM_APL_EVT_DBG_LV3(format, ...)

Request for collecting event logs (for Debug Level 3[CT])

APL requests for collecting logs when notifying necessary event for Debug

Supplement explanation:

- Level 3 is assumed as developer's optional log output for CT

Caution:

- This is enabled when APL Debug output level of Config is Level 3 or more than Level 3.

#define LOGM_APL_EVT_ERR(format, ...)

Request for collecting event logs (for Error)

APL requests for collecting logs when event to be notified as Error occurring.

Supplement explanation:

- Log output when obstruction that can be a factor of not system stopping but processing failure (parameter abnormality,state mismatch, API execution error .etc) is assumed.

#define LOGM_APL_EVT_INFO(format, ...)

Request for collecting event logs (for Info)

APL requests for collecting logs when event to be notified as "Information" occurring.

Supplement explanation:

- Log output to be needed for analyzing obstruction when important events (APL state/data change .etc) occur is assumed.

#define LOGM_APL_EVT_WARN(format, ...)

Request for collecting event logs (for Warning)

APL requests for collecting logs when event to be notified as "Warning" occurring.

Supplement explanation:

- Log output of information to be needed for analyzing obstruction when abnormality not affecting system operation is detected and processing can continue is assumed.

#define LOGM_APL_IS_EVT_DBG_LV3_ENABLE()

Confirmation whether event log DBG trace level 3 is enabled/disabled

Confirm whether Debug Level 3 of APL event log collection is enabled.

Return value:

Wheter event log DBG trace level 3 is enabled or disabled (true: enabled, false: disabled)

#define LOGM_APL_TRC_LV1_END(format, ...)

Request for collecting API finalization trace log (for Level 1)

APL requests for collecting logs in API finalization for APL API trace

Supplement explanation:

- Collecting point according to Level 1 is defined in APL.

Caution:

- This is enabled when APL API trace output level in Confing is Level1 or more than Level1.

#define LOGM_APL_TRC_LV1_STA(format, ...)

Request for collecting API start trace log (for Level 1)

APL requests for collecting logs in API start for APL API trace.

Supplement explanation:

- Collecting point according to Level 1 is defined in APL.

Caution:

- This is enabled when APL API trace output level in Confing is Level1 or more than Level1..

#define LOGM_APL_TRC_LV2_END(format, ...)

Request for collecting API finalization trace log (for Level 1)

APL requests for collecting logs in API finalization for APL API trace.

Supplement explanation:

- Collecting point according to Level 2 is defined in APL.

Caution:

- This is enabled when APL API trace output level in Confing is Level 2 or more than Level 2.

#define LOGM_APL_TRC_LV2_STA(format, ...)

Request for collecting API start trace log (for Level 2)

APL requests for collecting logs in API start for APL API trace.

Supplement explanation:

- Collecting point according to Level 2 is defined in APL.

Caution:

- This is enabled when APL API trace output level in Confing is Level 2 or more than Level 2.

#define LOGM_SIG_TRC_P_TCP_RCV(format, ...)

Request for collecting P signal (TCP) reception trace log

Request for collecting trace log when receiveing P signal (TCP)

#define LOGM_SIG_TRC_P_TCP_SND(format, ...)

Request for collecting P signal (TCP) transmission trace log

Request for collecting trace logs when transmitting P signal (TCP)

#define LOGM_SIG_TRC_P_UDP_RCV(format, ...)

Request for collecting P signal (UDP) reception trace log

Request for collecting trace log when receiving P signal (UDP)

#define LOGM_SIG_TRC_P_UDP_SND(format, ...)

Request for collecting P signal (UDP) transmission trace log

Request for trace logs when transmitting P signal (UDP)

4.9. membercontrol.h

Header of namespace of member control *FB*

4.9.1. Include File

```
#include <string>
#include <vector>
#include "distributedprocessingbase.h"
#include "DistributedException.hpp"
```

4.9.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::gmbc	Namespace of member control *FB*

4.9.3. Class List

class *xnode::distributedprocessingbase::gmbc::MemberControlInterface*	Interface class for APL of member control *FB*

4.9.4. Macro Definition

Nothing

4.10. originaldata.h

Header of namespace of original data management *FB*

4.10.1. Include File

```
#include "distributedprocessingbase.h"
#include "datacontainer.h"
```

4.10.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::odtm	Namespace of original data management *FB*

4.10.3. Class List

class xnode::distributedprocessingbase::odtm::LocalDdcIdIterator	Iterator control class for acquiring DDCID
class xnode::distributedprocessingbase::odtm::OriginalDataInterface	Interface class for APL of original data management *FB*

4.10.4. Macro Definition

Nothing

4.11. processor.h

Header of namespace of execution control *FB*

4.11.1. Include File

```
#include <vector>
#include <map>
#include "distributedprocessingbase.h"
#include "datacontainer.h"
```

4.11.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::mspr	Namespace of execution control *FB*

4.11.3. Class List

class xnode::distributedprocessingbase::mspr::IAsyncTaskListener	Listener base class for asynchronous task execution
class xnode::distributedprocessingbase::mspr::IApplicationListener	Listener base class for processing D and PS-APL
class xnode::distributedprocessingbase::mspr::IFactoryListener	Listener base class for creating APL listener class
class xnode::distributedprocessingbase::mspr::IDataListener	Listner base class for processing APL about DDC data
class xnode::distributedprocessingbase::mspr::DefaultDataListener	IDataListener default implementation class
class xnode::distributedprocessingbase::mspr::ProcessorInterface	Interface class for APL of execution control *FB*

4.11.4. Macro Definition

Nothing

4.12. TcpConnectionMng.hpp

TCP connection management class header

4.12.1. Include File

```
#include <string>
#include <vector>
#include <sv_api.h>
#include "DistributedException.hpp"
#include "InnerDistributedException.hpp"
```

4.12.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::comc	Namespace of communication control *FB*

4.12.3. Class List

class xnode::distributedprocessingbase::comc::TcpConnectionMng	TCP connection management class

4.12.4. Macro Definition

Nothing

4.13. ThreadPool.hpp

Thread pool class header

4.13.1. Include File

```
#include <sv_api.h>
#include <queue>
#include <boost/thread/thread.hpp>
#include "DistributedException.hpp"
```

4.13.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::cmnp	Namespace of common parts *FB.*

4.13.3. Class List

class xnode::distributedprocessingbase::cmnp::ThreadPool	Thread pool class

4.13.4. Macro Definition

Nothing

4.14. ThreadPoolMng.hpp

Thread pool manager class header

4.14.1. Include File

```
#include <sv_api.h>
#include "ThreadPool.hpp"
#include "DistributedException.hpp"
```

4.14.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::cmnp	Namespace of common parts *FB.*

4.14.3. Class List

class xnode::distributedprocessingbase::cmnp::ThreadPoolMng	Thread pool manager class

4.14.4. Macro Definition

Nothing

4.15. TimerMng.hpp

Timer manager class header

4.15.1. Include File

```
#include <vector>
#include <sv_api.h>
#include "DistributedException.hpp"
```

4.15.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::cmnp	Namespace of common parts *FB*

4.15.3. Class List

class xnode::distributedprocessingbase::cmnp::TimerMng	Timer manager class

4.15.4. Macro Definition

Nothing

4.16. UdpControl.hpp

UDP communication control class header

4.16.1. Include File

```
#include <string>
#include <vector>
#include <sv_api.h>
#include "InnerDistributedException.hpp"
```

4.16.2. Namespace List

xnode	Basic namespace of distributed processing base system
xnode::distributedprocessingbase	Namespace of distributed processing base
xnode::distributedprocessingbase::comc	Namespace of communication control *FB*

4.16.3. Class List

class xnode::distributedprocessingbase::comc::UdpControl	UDP communication control class

4.16.4. Macro Definition

Nothing