

doc_exe.cpp

```
#ifndef _Mutex
#define _Mutex
#include <pthread.h>
#include "rt_sts.h"
class Mutex
{
private:
    pthread_mutex_t moMutex ;
public:
    Mutex( void ) { pthread_mutex_init( &( this->moMutex ), NULL ); }
    ~Mutex( void ) { pthread_mutex_destroy( &( this->moMutex ) ); }
    inline int Lock( void )
    {
        return ( ( pthread_mutex_lock( &this->moMutex ) == 0 ) ?
            RT_S_SUCCESS : RT_E_LOCK );
    }
    inline int Unlock( void )
    {
        return ( ( pthread_mutex_unlock( &this->moMutex ) == 0 ) ?
            RT_S_SUCCESS : RT_E_UNLOCK );
    }
};
#endif

GThreadManager *pManager = GThreadManager::GetInstance();
FXR_ALLOC_RETURN(pManager, FXR_LOG_CMN_NEW_ERR FXR_LOG_FUNC_THREAD_ALL)
GThreadGroup *pManagerProvider = pManager->CreateGroup(FXR_LOG_CLASS_NAME_PROVIDER);
FXR_ALLOC_RETURN(pManagerProvider, FXR_LOG_CMN_NEW_ERR FXR_LOG_FUNC_THREAD_PROVIDER)
nStatus = pManager->AddGroup(pManagerProvider);
FXR_ERR_RETRUN(nStatus, FXR_LOG_CMN_ADD_ERR FXR_LOG_FUNC_THREAD_PROVIDER)
// OPE電文受信待機オブジェクトの生成
OPE oOPE( ( char* )PROC_SYSTEM_NAME, FxrLogger::GetLogWriter(), &oSysProp, NULL );

// OPE電文受信待機スレッド始動
FXR_LOG_WRITE_I(FXR_LOG_CMN_EXECUTE FXR_LOG_NAME_OPE)
nStatus = oOPE.Init();
FXR_ERR_RETRUN(nStatus, FXR_LOG_CMN_START_ERR FXR_LOG_NAME_OPE)

JoinThread *pJoinThread = new( std::nothrow )JoinThread();
FXR_ALLOC_RETURN(pJoinThread, FXR_LOG_CMN_NEW_ERR FXR_FUNC_THREAD_WAIT)
nStatus = pJoinThread->Init();
FXR_ERR_RETRUN(nStatus, FXR_LOG_CMN_START_ERR FXR_FUNC_THREAD_WAIT)

memset( sBuf, 0x00, sizeof( sBuf ) );
sprintf( sBuf, FXR_LOG_NAME_THREAD_PROVIDER, 1 );
pFXRProvider = new( std::nothrow )FxrProvider(
    &oSysProp
    , sBuf
    , 1
    , &oCurPairCdMtQueMap
);
FXR_ALLOC_RETURN(pFXRProvider, FXR_LOG_CMN_NEW_ERR FXR_LOG_NAME_PROVIDER)
nStatus = pFXRProvider->Init();
FXR_ERR_RETRUN(nStatus, FXR_LOG_CMN_INIT_ERR FXR_LOG_NAME_PROVIDER)
nStatus = pManagerProvider->AddThread(sBuf, pFXRProvider);
FXR_ERR_RETRUN(nStatus, FXR_LOG_CMN_ADD_ERR FXR_LOG_NAME_PROVIDER)
for( std::map<std::pair<int,int>, MTQue<RealTimeFXRate>*>::iterator itThread = oThreadMtQueMap.begin();
    itThread != oThreadMtQueMap.end(); itThread++ ) {
    memset( sBuf, 0x00, sizeof( sBuf ) );
    sprintf( sBuf, FXR_LOG_NAME_THREAD_UPDATER, itThread->first.first, itThread->first.second );
    pFXRUpdater = new( std::nothrow )FxrUpdater(
        &oSysProp
        , sBuf
        , itThread->first.first
        , itThread->first.second
        , itThread->second
    );
    FXR_ALLOC_RETURN(pFXRUpdater, FXR_LOG_CMN_NEW_ERR FXR_LOG_NAME_UPDATER)
    nStatus = pFXRUpdater->Init();
    FXR_ERR_RETRUN(nStatus, FXR_LOG_CMN_INIT_ERR FXR_LOG_NAME_UPDATER)
    nStatus = pManagerUpdater->AddThread(sBuf, pFXRUpdater);
    FXR_ERR_RETRUN(nStatus, FXR_LOG_CMN_ADD_ERR FXR_LOG_NAME_UPDATER)
}
FXR_LOG_WRITE_I(FXR_LOG_CMN_EXECUTE FXR_LOG_NAME_PROVIDER)
nStatus = pManagerProvider->StartThread( 1 );
FXR_ERR_RETRUN(nStatus, FXR_LOG_CMN_START_ERR FXR_LOG_NAME_PROVIDER)
nStatus = pJoinThread->JoinThreads();
if( ERR_CHECK(nStatus) ) {
    FXR_LOG_WRITE(nStatus, FXR_LOG_CMN_JOIN_THREAD_ERR FXR_FUNC_THREAD_WAIT);
}
FXR_LOG_WRITE_I(FXR_LOG_CMN_EXECUTE FXR_FUNC_END_PROVIDER)
nStatus = pManagerProvider->StopThread( GThreadBase::THREAD_STOP_NOMAL );
if( ERR_CHECK(nStatus) ) {
    FXR_LOG_WRITE(nStatus, FXR_LOG_CMN_JOIN_THREAD_ERR FXR_FUNC_END_PROVIDER)
}
else {
    FXR_LOG_WRITE_I(FXR_LOG_CMN_EXECUTE FXR_FUNC_WAIT_PROVIDER)
    nStatus = pManagerProvider->Join();
    if( ERR_CHECK(nStatus) ) {
        FXR_LOG_WRITE(nStatus, FXR_LOG_CMN_JOIN_THREAD_ERR FXR_FUNC_WAIT_PROVIDER)
    }
}
oOPE.StopLoop();
GThreadManager::DeleteInstance();
FXR_LOG_WRITE(RT_I_EXIT , FXR_LOG_PROCESS_EXIT);
delete pJoinThread;
pJoinThread = NULL;
for(std::map< std::pair<int,int>, MTQue<RealTimeFXRate>*>::iterator it = oThreadMtQueMap.begin();
    delete it->second;
}
```