

doc\_GThreadManager.cpp

```

GThreadGroup( const char *_sGroupName );
virtual ~GThreadGroup();
virtual int ExecAllObject( int _nExec, int &_nArg );
virtual int DeleteObject( const char *_sKey );
virtual int DeleteAllObject();
};

#endif
---GThreadGroup.cpp
#include "GThreadGroup.h"
#include "rt_log_macro.h"
#include <new>
static const char *sProc[] = {
    "Start thread"
    , "Stop thread"
    , "Check activ count"
    , "Check thread count"
    , "Check thread status"
    , "Get max priority"
    , "Thread join"
};

GThreadGroup::GThreadGroup(
    const char *_sGroupName
)
: sGroupName( _sGroupName )
{
}

GThreadGroup::~GThreadGroup(
    void
)
{
    this->DeleteAllObject();
}

int GThreadGroup::AddThread(
    const char *_sKey
    , Runnable *_pObject
    , int _nSetPriority
    , int _nSetPolicy
    , int _nSetScope
)
{
    int nStatus;
    GThreadBase *pThread;
    String skey;
    static const char *sMsg = "AddThreadYtKey[ %s ]YtGroup[ %s ]";
    nStatus = 0;
    pThread = NULL;
    NULL_RETURN( _sKey );
    NULL_RETURN( _pObject );
    pThread = new( std::nothrow )GThread(
        _pObject
        , _nSetPriority
        , _nSetPolicy
        , _nSetScope
    );
    ALLOC_RETURN( pThread );
    skey = _sKey;
    nStatus = this->oMutex.Lock();
    ERR_RETURN( nStatus );
    nStatus = this->oThreadMap.Put( skey, pThread );
    if( ERR_CHECK( nStatus ))
    {
        this->oMutex.Unlock();
        return nStatus;
    }
    nStatus = this->oMutex.Unlock();
    ERR_RETURN( nStatus );
    return RT_S_SUCCESS;
}

int GThreadGroup::AddGroup(
    GThreadGroup *_pObject
)
{
    int nStatus;
    const char *sMsg = "AddGroupYtKey[ %s ]YtGroup[ %s ]";
    nStatus = 0;
    NULL_RETURN( _pObject );
    nStatus = this->oMutex.Lock();
    ERR_RETURN( nStatus );
    nStatus = this->oThreadMap.Put( _pObject->sGroupName , _pObject );
    if( ERR_CHECK( nStatus ))
    {
        this->oMutex.Unlock();
    }
}

```