

```

        return nStatus;
    }
    nStatus = this->oMutex.Unlock();
    ERR_RETURN( nStatus );
    return RT_S_SUCCESS;
}

int GThreadGroup::DeleteObject(
    const char *_sKey
)
{
    int nStatus;
    int nObjType;
    GThreadBase *pObject;
    String sKey;
    const char *sMsg = "DeleteThreadVtKey[ %s ]VtGroup[ %s ]";
    nStatus = 0;
    nObjType = 0;
    pObject = NULL;
    NULL_RETURN( _sKey );
    sKey = _sKey;
    nStatus = this->oMutex.Lock();
    ERR_RETURN( nStatus );
    nStatus = this->oThreadMap.Remove( sKey, pObject );
    if( ERR_CHECK( nStatus ) )
    {
        this->oMutex.Unlock();
        return nStatus;
    }
    if( pObject == NULL )
    {
        this->oMutex.Unlock();
        return RT_F_NULL_POINTER;
    }
    nStatus = this->oMutex.Unlock();
    ERR_RETURN( nStatus );
    delete pObject;
    pObject = NULL;
    return RT_S_SUCCESS;
}

int GThreadGroup::DeleteAllObject(
    void
)
{
    int nSize;
    int i;
    int nStatus;
    String sKey;
    GThreadBase *pObject;
    static const char *sMsg = "DeleteObjectVtKey[ %s ]VtGroup[ %s ]";
    nSize = 0;
    i = 0;
    nStatus = 0;
    pObject = NULL;
    nStatus = this->oMutex.Lock();
    ERR_RETURN( nStatus );
    nSize = this->oThreadMap.GetElementSize();
    for( i = 0 ; i < nSize ; i++ )
    {
        nStatus = this->oThreadMap.GetToIndex( i, sKey, pObject );
        if( ERR_CHECK( nStatus ) )
        {
            this->oMutex.Unlock();
            return nStatus;
        }
        if( pObject == NULL )
        {
            this->oMutex.Unlock();
            return RT_F_NULL_POINTER;
        }
        delete pObject;
        pObject = NULL;
    }
    this->oThreadMap.Clear();
    nStatus = this->oMutex.Unlock();
    ERR_RETURN( nStatus );
    return RT_S_SUCCESS;
}

int GThreadGroup::ExecAllObject(
    int _nExec
    , int &nArg
)
{

```