
Subject: missing callback5 and implementation is provided

Posted by [aftershock](#) on Mon, 06 Apr 2015 20:44:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

callback5 is not implemented in core.

Here is the implementation. Will you add it?

Here is the implemented callback5

```
#define THISBACK5(m, a, b, c, d,e) callback5(this, &CLASSNAME::m, a, b, c, d,e)
```

```
template <class O, class M, class T1, class T2, class T3, class T4, class T5>
```

```
struct CallbackMethodActionArg5 : public CallbackAction
```

```
{
    O *object;
    M method;
    T1 arg1;
    T2 arg2;
    T3 arg3;
    T4 arg4;
    T5 arg5;
```

```
void Execute()
{
    ( object->*method ) ( arg1, arg2, arg3, arg4, arg5 );
}
```

```
CallbackMethodActionArg5 ( O *object, M method, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5 )
    : object ( object ), method ( method ), arg1 ( arg1 ), arg2 ( arg2 ), arg3 ( arg3 ), arg4 ( arg4 ),
    arg5 ( arg5 ) {}
};
```

```
template <class O, class M, class Q1, class Q2, class Q3, class Q4, class Q5, class T1, class T2,
class T3, class T4, class T5>
```

```
Callback callback5 ( O *object, void ( M::*method ) ( Q1, Q2, Q3, Q4, Q5 ), T1 arg1, T2 arg2, T3
arg3, T4 arg4, T5 arg5 )
```

```
{
    return Callback (
        new CallbackMethodActionArg5 < O, void ( M::* ) ( Q1, Q2, Q3, Q4, Q5 ), T1, T2, T3, T4, T5 >
        ( object, method, arg1, arg2, arg3, arg4, arg5 ) );
}
```

```

template <class O, class M, class Q1, class Q2, class Q3, class Q4, class Q5, class T1, class T2,
class T3, class T4, class T5>
Callback callback5 ( const O *object, void ( M::*method ) ( Q1, Q2, Q3, Q4, Q5 ) const, T1 arg1,
T2 arg2, T3 arg3, T4 arg4, T5 arg5 )
{
    return Callback ( new CallbackMethodActionArg5 < const O, void ( M::* ) ( Q1, Q2, Q3, Q4, Q5 )
const, T1, T2, T3, T4, T5 >
        ( object, method, arg1, arg2, arg3, arg4, arg5 ) );
}

```

```

template <class P1, class P2, class P3, class P4, class P5>

```

```

struct Callback5Action

```

```

{
    Atomic count;

    virtual void Execute ( P1 p1, P2 p2, P3 p3, P4 p4, P5 p5 ) = 0;
    virtual bool IsValid() const
    {
        return true;
    }
}

```

```

Callback5Action()

```

```

{
    count = 1;
}

```

```

virtual ~Callback5Action() {}
};

```

```

#ifdef HAS_LAMBDA

```

```

template <class P1, class P2, class P3, class P4, class P5>

```

```

struct LambdaCallback5 : public Callback4Action<P1, P2, P3, P4, P5>

```

```

{
    std::function < void ( P1, P2, P3, P4 , P5 ) > fn;
    virtual void Execute ( P1 p1, P2 p2, P3 p3, P4 p4 , P5 p5 )
    {
        fn ( p1, p2, p3, p4, p5 );
    }
}

```

```

LambdaCallback5 ( std::function < void ( P1, P2, P3, P4, P5 ) > fn ) : fn ( fn ) {}
};

```

```

#endif

```

```

template <class P1, class P2, class P3, class P4, class P5>

class Callback5 : Moveable< Callback5<P1, P2, P3, P4, P5> >
{
    Callback5Action<P1, P2, P3, P4, P5> *action;

    void Retain() const
    {
        if ( action )
            AtomicInc ( action->count );
    }

    void Release()
    {
        if ( action && AtomicDec ( action->count ) == 0 )
            delete action;
    }

    bool operator== ( const Callback5& );
    bool operator!= ( const Callback5& );

public:
    typedef Callback5 CLASSNAME;

    Callback5& operator= ( const Callback5& c );
    Callback5 ( const Callback5& c );
    void Clear()
    {
        Release();
        action = NULL;
    }

    operator bool() const
    {
        return action && action->IsValid();
    }

    void Execute ( P1 p1, P2 p2, P3 p3, P4 p4, P5 p5 ) const
    {
        if ( action )
            action->Execute ( p1, p2, p3, p4, p5 );
    }

    void operator() ( P1 p1, P2 p2, P3 p3, P4 p4, P5 p5 ) const
    {
        Execute ( p1, p2, p3, p4, p5 );
    }

```

```

}

explicit Callback5 ( Callback5Action <P1, P2, P3, P4, P5> *newaction )
{
    action = newaction;
}

Callback5()
{
    action = NULL;
}

Callback5 ( _CNULL )
{
    action = NULL;
}

~Callback5();

#ifdef CPP_11
template <class T>
Callback5 ( AnyLambda<T> l )
{
    action = new LambdaCallback4<P1, P2, P3, P4, P5> ( l.l );
}

#endif

static Callback5 Empty()
{
    return CNULL;
}

};

template <class P1, class P2, class P3, class P4, class P5>
Callback5<P1, P2, P3, P4, P5>& Callback5<P1, P2, P3, P4, P5>::operator= ( const Callback5 & c
)
{
    c.Retain();
    Release();
    action = c.action;
    return *this;
}

template <class P1, class P2, class P3, class P4, class P5>
Callback5<P1, P2, P3, P4, P5>::Callback5 ( const Callback5& c )
{
    action = c.action;
}

```

```
Retain();
}
```

```
template <class P1, class P2, class P3, class P4, class P5>
Callback5<P1, P2, P3, P4, P5>::~~Callback5()
{
    Release();
}
```

```
template < class X, class T1, class T2, class T3, class T4, class T5, class HC = X >
```

```
struct CallbackActionCallArg5 : public CallbackAction
{
    X      x;
    T1     arg1;
    T2     arg2;
    T3     arg3;
    T4     arg4;
    T5     arg5;
    void    Execute()
    {
        x ( arg1, arg2, arg3, arg4, arg5 );
    }
}
```

```
CallbackActionCallArg5 ( X x, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5 )
: x ( x ), arg1 ( arg1 ), arg2 ( arg2 ), arg3 ( arg3 ), arg4 ( arg4 ), arg5 ( arg5 ) {}
};
```

```
template <class Q1, class Q2, class Q3, class Q4, class Q5, class T1, class T2, class T3, class
T4, class T5>
Callback callback5 ( void ( *fn ) ( Q1, Q2, Q3, Q4, Q5 ), T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5
arg5 )
{
    return Callback (
        new CallbackActionCallArg5 < void ( * ) ( Q1, Q2, Q3, Q4, Q5 ), T1, T2, T3, T4, T5, uintptr_t >
( fn, arg1, arg2, arg3, arg4, arg5 ) );
}
```

```
template <class Q1, class Q2, class Q3, class Q4, class Q5, class T1, class T2, class T3, class
T4, class T5>
Callback callback5 ( Callback5<Q1, Q2, Q3, Q4, Q5> cb, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5
arg5 )
{
    return Callback (
        new CallbackActionCallArg5<Callback5<Q1, Q2, Q3, Q4, Q5>, T1, T2, T3, T4, T5> ( cb, arg1,
```

```
arg2, arg3, arg4, arg5 ) );  
}
```
