

Hello All,

I propose to improve a little callback mechanism by adding new functionality.

Example: In a class I have

```
Callback2<P1, P2, P2, P3, P4> when_callback;
```

From other class I want to attach a method

```
OnCallback1(P1 p1, P2 p2, P3 p3, P4 p4, T1 t1, T2 t2, T3 t3){  
    ...  
}
```

When I try to attach:

```
Attach(){when_callback = THISBACK3(OnCallback1, val1, val2, val3)};
```

Returned an error.

With the attached patch this functionality is integrated.

Maximum can be 4 parameters and 4 arguments(constants). Total parameters can be 8.

Maybe I give a non-simple example.

If this functionality will be accepted that I can improve a little with functions(not methods).

A little example is:

```
#include <Core/Core.h>
```

```
using namespace Upp;
```

```
struct Foo {  
    int x;
```

```
void Action()          { Cout() << "Action: " << x << '\n'; }  
void ActionWithParam(int y) { Cout() << "ActionWithParam: " << x + y << '\n'; }
```

```
Callback WhenDo;  
Callback1<int> WhenDo1;
```

```

Callback4<int, String, String, String> WhenDo4;
void Do()          { WhenDo(); WhenDo1(2);WhenDo4(4, "2 par", "3 par", "4 par");}

Foo(int x = 0) : x(x) {}
};

void Fn()
{
    Cout() << "Fn!" << '\n';
}

struct Bar {
    Foo foo;

    void Action() { Cout() << "foo's Do called\n"; }
    void ActionMaxPar(int p1, String p2, String p3, String p4) { Cout() << "foo's Do called with max
par par 4 = "<<p4<<"\n"; }
    void ActionMaxParAndArg(int p1, String p2, String p3, String p4, int a1, String a2, String a3,
String a4) { Cout() << "foo's Do called with max par and arg arg4="<<a4<<"\n"; }

typedef Bar CLASSNAME;

Bar() { foo.WhenDo = THISBACK(Action);
    String v_arg1("1 arg");
    String v_arg2("2 arg");
    String v_arg3("3 arg");
    String v_arg4("4 arg");
    int v_arg1_i = 1;

    foo.WhenDo1 = THISBACK3(ActionMaxPar, v_arg1, v_arg2, v_arg3);
    foo.WhenDo4 = THISBACK(ActionMaxPar);
    foo.WhenDo4 << THISBACK4(ActionMaxParAndArg, v_arg1_i, v_arg2, v_arg3, v_arg4);
}
};

struct Safe : Pte<Safe> {
    void Action() { Cout() << "safe action!\n"; }
};

CONSOLE_APP_MAIN
{
    Foo a(10);
    Callback cb1 = callback(&a, &Foo::Action);
    Callback cb2 = callback(Fn);
    Callback1<int> cb3 = callback(&a, &Foo::ActionWithParam);
    Callback cb4 = callback1(&a, &Foo::ActionWithParam, 30);

    cb1();

```

```

cb2();
cb3(10);
cb4();

Cout() << "-----\n";
cb4 << cb2;
cb4();

Cout() << "-----\n";
Bar b;
b.foo.Do();

Cout() << "-----\n";
{
    Safe f;
    cb4 = pteback(&f, &Safe::Action);
    Cout() << "callback valid: " << (bool)cb4 << '\n';
    cb4();
}
Cout() << "callback valid: " << (bool)cb4 << '\n';
cb4();
}

```

(example attached)

Ion Lupascu (tojocky)

## File Attachments

---

- 1) [Callback.h](#), downloaded 558 times
  - 2) [Callback\\_example.zip](#), downloaded 482 times
-