```
class RealTimeAsyncUpdaterTests : public juce::UnitTest
{
public:
    RealTimeAsyncUpdaterTests()
        : juce::UnitTest ("RealTimeAsyncUpdater", "Tracktion:Longer") {}

    //==============================================================================
    void runTest() override
    {
        if (MessageManager::getInstanceWithoutCreating() == nullptr)
            return;

        beginTest ("juce::AsyncUpdater");
        runAsyncUpdateTest<juce::AsyncUpdater>();
        beginTest ("RealTimeAsyncUpdater");
        runAsyncUpdateTest<RealTimeAsyncUpdater>();
    }

    template<typename UpdaterType>
    void runAsyncUpdateTest())
    {
       ...
    }

    template<typename UpdaterType>
    struct UpdaterTest : public UpdaterType
    {
       …
    };
};

static RealTimeAsyncUpdaterTests realTimeAsyncUpdaterTests;
```

```
class RealTimeAsyncUpdaterTests : public juce::UnitTest
{
public:
    RealTimeAsyncUpdaterTests()
        : juce::UnitTest ("RealTimeAsyncUpdater", "Tracktion:Longer") {}

    //==============================================================================
    void runTest() override
    {
        if (MessageManager::getInstanceWithoutCreating() == nullptr)
            return;

        beginTest ("juce::AsyncUpdater");
        runAsyncUpdateTest<juce::AsyncUpdater>();
        beginTest ("RealTimeAsyncUpdater");
        runAsyncUpdateTest<RealTimeAsyncUpdater>();
    }

    template<typename UpdaterType>
    void runAsyncUpdateTest())
    {
       ...
    }

    template<typename UpdaterType>
    struct UpdaterTest : public UpdaterType
    {
       …
    };
};

static RealTimeAsyncUpdaterTests realTimeAsyncUpdaterTests;
```

```cpp
template<typename UpdaterType>
struct UpdaterTest : public UpdaterType
{
    UpdaterTest() = default;

    void sendUpdate()
    {
        hasDelivered = false;
        UpdaterType::triggerAsyncUpdate();
    }

    void handleAsyncUpdate() override
    {
        hasDelivered = true;
        event.signal();
        JUCE_ASSERT_MESSAGE_THREAD;
    }

    WaitableEvent event;
    std::atomic<bool> hasDelivered { false };
};
```