

CommonAPITests

Generated by Doxygen 1.9.3

1 CommonAPITests	1
2 Test List	3
3 File Index	25
3.1 File List	25
4 File Documentation	27
4.1 mainpagetests/01_mainpage.dox File Reference	27
4.2 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↵ commonapi.core.verification/src/AFExtended.cpp File Reference	27
4.2.1 Function Documentation	27
4.2.1.1 AFExtended_MethodCall()	27
4.2.1.2 AFExtended_Attributes()	28
4.2.1.3 AFExtended_Broadcast()	28
4.2.1.4 main()	28
4.2.2 Variable Documentation	28
4.2.2.1 servid	28
4.2.2.2 clientId	28
4.2.2.3 domain	28
4.2.2.4 testAddressBase	29
4.2.2.5 testAddressOnce	29
4.2.2.6 testAddressTwice	29
4.2.2.7 tasync	29
4.3 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↵ commonapi.core.verification/src/AFManaged.cpp File Reference	29
4.3.1 Macro Definition Documentation	30
4.3.1.1 INTERFACE_DEVICE	30
4.3.1.2 INTERFACE_SPECIAL_DEVICE	30
4.3.1.3 MIDDLE_INTERFACE	31
4.3.1.4 BOTTOM_INTERFACE	31
4.3.2 Function Documentation	31
4.3.2.1 AFManaged_AddRemoveManagedInterfaceSingle()	31
4.3.2.2 AFManaged_AddRemoveManagedInterfaceMultiple()	31
4.3.2.3 AFManaged_AddRemoveMultipleManagedInterfacesSingle()	32
4.3.2.4 AFManaged_AddRemoveMultipleManagedInterfacesMultiple()	32
4.3.2.5 AFManaged_AddRemoveMultipleManagedInterfacesMultipleProxyNotActive()	32
4.3.2.6 AFManaged_ProxyAddRemoveManagedInterfaceSingle()	32
4.3.2.7 AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicit()	33
4.3.2.8 AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicitAll()	33
4.3.2.9 AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationImplicit()	33
4.3.2.10 AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicit()	34
4.3.2.11 AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicitAll()	34
4.3.2.12 AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationImplicit()	34

4.3.2.13	AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallSingleDeregistrationExplicit()	35
4.3.2.14	AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallInProxyStatusEventSingleDeregistrationExplicit()	35
4.3.2.15	AFManaged_DeleteManagerProxyInsideProxyStatusEventCallbackAndMethodCall()	35
4.3.2.16	AFManaged_ProxyManagerTestPrimitiveMethods()	36
4.3.2.17	AFManaged_ProxyManagerTestNonPrimitiveMethodsSync()	36
4.3.2.18	AFManaged_ProxyManagerTestNonPrimitiveMethodsAsync()	36
4.3.2.19	AFManaged_ProxyManagerTestGetInstanceAvailabilityStatusAsync()	37
4.3.2.20	AFManaged_AddRemoveHierarchicalManagedInterface()	37
4.3.2.21	AFManaged_GetAvailableInstancesWithoutSubscribe()	38
4.3.2.22	AFManaged_CreateProxyToManagerInSameProcess()	38
4.3.2.23	main()	38
4.3.3	Variable Documentation	38
4.3.3.1	domain	38
4.4	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/AFPolymorph.cpp File Reference	38
4.4.1	Function Documentation	39
4.4.1.1	AFPolymorph_SetAndGetAttributeTypedef()	39
4.4.1.2	AFPolymorph_SetAndGetAttributeEnum()	39
4.4.1.3	AFPolymorph_SetAndGetAttributeUInt()	39
4.4.1.4	AFPolymorph_SetAndGetAttributeString()	40
4.4.1.5	AFPolymorph_SetAndGetAttributeStruct()	40
4.4.1.6	AFPolymorph_MethodCall()	40
4.4.1.7	AFPolymorph_Broadcast()	40
4.4.1.8	AFPolymorph_SetAndGetAttributeDoublyUsedBaseStruct()	40
4.4.1.9	main()	41
4.4.2	Variable Documentation	41
4.4.2.1	domain	41
4.4.2.2	testAddress	41
4.4.2.3	connectionId_client	41
4.4.2.4	connectionId_service	41
4.4.2.5	tasync	41
4.5	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/AFSelective.cpp File Reference	42
4.5.1	Function Documentation	42
4.5.1.1	AFSelective_SelectiveBroadcastRejected()	42
4.5.1.2	AFSelective_SelectiveBroadcast()	43
4.5.1.3	AFSelective_SelectiveMultiBroadcast()	43
4.5.1.4	AFSelective_ProxyBuildAndDestroy()	43
4.5.1.5	AFSelective_SelectiveRejectedMultiBroadcast()	43
4.5.1.6	AFSelective_Multiple_Subscriptions_SameConnection_CallErrorHandler()	44
4.5.1.7	AFSelective_Fire_Selective_Within_Subscription_Changed_Hook()	44
4.5.1.8	AFSelective_Two_proxies_subscribe_delete_one_proxy()	44
4.5.1.9	AFSelective_Two_proxies_subscribe_delete_one_proxy_error_listener_test()	44

4.5.1.10	main()	44
4.5.2	Variable Documentation	44
4.5.2.1	serviceld	44
4.5.2.2	clientId	44
4.5.2.3	otherclientId	45
4.5.2.4	domain	45
4.5.2.5	testAddress	45
4.5.2.6	tasync	45
4.6	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↔ commonapi.core.verification/src/CMAAttributes.cpp File Reference	45
4.6.1	Function Documentation	45
4.6.1.1	CMAAttributes_AttributeGetSynchronous()	46
4.6.1.2	CMAAttributes_AttributeGetAsynchronous()	46
4.6.1.3	CMAAttributes_AttributeSetSynchronous()	46
4.6.1.4	CMAAttributes_AttributeSetAsynchronous()	47
4.6.1.5	CMAAttributes_AttributeSubscription()	47
4.6.1.6	main()	47
4.6.2	Variable Documentation	47
4.6.2.1	serviceld	47
4.6.2.2	clientId	48
4.6.2.3	domain	48
4.6.2.4	testAddress	48
4.6.2.5	tasync	48
4.7	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↔ commonapi.core.verification/src/CMAAttributeSubscription.cpp File Reference	48
4.7.1	Typedef Documentation	49
4.7.1.1	ProxyPtr	49
4.7.2	Function Documentation	49
4.7.2.1	testSubscription()	49
4.7.2.2	CMAAttributeSubscription_SubscriptionStandard()	49
4.7.2.3	CMAAttributeSubscription_SubscriptionOnAvailable()	50
4.7.2.4	CMAAttributeSubscription_SubscriptionMultithreading()	50
4.7.2.5	CMAAttributeSubscription_SubscriptionUnsubscribeFromCallback()	50
4.7.2.6	CMAAttributeSubscription_SubscribeAndUnsubscribeTwoCallbacksCoexistent()	51
4.7.2.7	CMAAttributeSubscription_SubscribeAndUnsubscribeSequentially()	51
4.7.2.8	CMAAttributeSubscription_DISABLED_SubscribeAndUnsubscribeImplicitWithCreatingNewProxyWithReassign	
4.7.2.9	CMAAttributeSubscription_SubscribeAndUnsubscribeUnsubscribe()	52
4.7.2.10	CMAAttributeSubscription_SubscribeServiceNotAvailable()	53
4.7.2.11	CMAAttributeSubscription_SubscribeUnregisterSetValueRegisterService()	53
4.7.2.12	CMAAttributeSubscription_SubscribeUnregisterNoValueSetRegisterService()	54
4.7.2.13	CMAAttributeSubscription_SubscribeSecondProxyLater()	54
4.7.2.14	CMAAttributeSubscription_SubscribeThreeCallbacksServiceNotAvailable()	54
4.7.2.15	CMAAttributeSubscription_SubscribeThreeCallbacksServiceAvailable()	55

4.7.2.16 CMAAttributeSubscription_SubscribeAndUnsubscribeAndReSubscribe()	55
4.7.2.17 CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribe()	55
4.7.2.18 CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribeSameEventgroup()	56
4.7.2.19 CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeOneResubscribeSameEventgroup()	57
4.7.2.20 main()	57
4.7.3 Variable Documentation	57
4.7.3.1 daemonId	57
4.7.3.2 clientId	57
4.7.3.3 serviceId	58
4.7.3.4 domain	58
4.7.3.5 testAddress	58
4.7.3.6 daemonAddress	58
4.7.3.7 wt	58
4.7.3.8 wf	58
4.7.3.9 mut	58
4.7.3.10 data_queue	58
4.7.3.11 data_cond	59
4.8 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↔ commonapi.core.verification/src/CMBlockingCalls.cpp File Reference	59
4.8.1 Function Documentation	59
4.8.1.1 CMBlockingCalls_BlockInStubMethod()	59
4.8.1.2 CMBlockingCalls_BlockInProxyCallback()	60
4.8.1.3 CMBlockingCalls_BlockInAvailabilityHandler()	60
4.8.1.4 CMBlockingCalls_BlockInAvailabilityHandlerAndReceiveCallbacks()	60
4.8.1.5 CMBlockingCalls_NestedBlockInStubMethods()	60
4.8.1.6 main()	60
4.8.2 Variable Documentation	61
4.8.2.1 serviceId	61
4.8.2.2 clientId	61
4.8.2.3 clientId2	61
4.8.2.4 domain	61
4.8.2.5 testAddress	61
4.8.2.6 testAddress2	61
4.8.2.7 tasync	61
4.8.2.8 timeout	62
4.8.2.9 maxTimeoutCalls	62
4.8.2.10 wf	62
4.9 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↔ commonapi.core.verification/src/CMBroadcasts.cpp File Reference	62
4.9.1 Function Documentation	62
4.9.1.1 CMBroadcasts_NormalBroadcast()	63
4.9.1.2 CMBroadcasts_SelectiveBroadcastRejected()	63
4.9.1.3 CMBroadcasts_SelectiveBroadcast()	63

4.9.1.4 CMBroadcasts_BroadcastStubGoesOfflineOnlineAgain()	63
4.9.1.5 CMBroadcasts_SelectiveBroadcastStubGoesOfflineOnlineAgain()	64
4.9.1.6 CMBroadcasts_NormalBroadcast_Two_proxies_subscribe_and_one_reset()	64
4.9.1.7 CMBroadcasts_Two_proxies_subscribe_delete_one_proxy_status_listener_test()	64
4.9.1.8 main()	64
4.9.2 Variable Documentation	64
4.9.2.1 serviceId	64
4.9.2.2 clientId	65
4.9.2.3 otherclientId	65
4.9.2.4 domain	65
4.9.2.5 testAddress	65
4.9.2.6 tasync	65
4.9.2.7 wf	65
4.10 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↔ commonapi.core.verification/src/CMMMethodCalls.cpp File Reference	65
4.10.1 Function Documentation	66
4.10.1.1 CMMMethodCalls_SynchronousMethodCall()	66
4.10.1.2 CMMMethodCalls_FireAndForget()	66
4.10.1.3 CMMMethodCalls_AsynchronousMethodCall()	67
4.10.1.4 CMMMethodCalls_NestedSynchronousMethodCall()	67
4.10.1.5 CMMMethodCalls_NestedAsynchronousMethodCall()	67
4.10.1.6 CMMMethodCalls_NestedAsynchronousMethodCallsTimedOut()	67
4.10.1.7 CMMMethodCalls_AsynchronousMethodCallProxyNotAvailable()	68
4.10.1.8 CMMMethodCalls_NestedAsynchronousMethodCallProxyNotAvailable()	68
4.10.1.9 CMMMethodCalls_AsynchronousMethodCallProxyBecomesAvailable()	68
4.10.1.10 CMMMethodCalls_NestedAsynchronousMethodCallProxyBecomesAvailable()	69
4.10.1.11 CMMMethodCalls_AsynchronousMethodCallsProxyBecomesAvailable()	69
4.10.1.12 CMMMethodCalls_AsynchronousMethodCallProxyNotAvailableDeleteProxy()	69
4.10.1.13 CMMMethodCalls_AsynchronousMethodCallsReceiveNotAvailable()	70
4.10.1.14 main()	70
4.10.2 Variable Documentation	70
4.10.2.1 serviceId	70
4.10.2.2 clientId	70
4.10.2.3 domain	70
4.10.2.4 testAddress	70
4.10.2.5 testAddress2	71
4.10.2.6 tasync	71
4.10.2.7 timeout	71
4.10.2.8 maxTimeoutCalls	71
4.10.2.9 wf	71
4.11 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↔ commonapi.core.verification/src/DTAdvanced.cpp File Reference	71
4.11.1 Function Documentation	72

4.11.1.1	DTAdvanced_SendAndReceive()	72
4.11.1.2	DTAdvanced_SendAndReceiveInvalid()	72
4.11.1.3	DTAdvanced_DISABLED_SendAndReceiveMapInvalid()	72
4.11.1.4	DTAdvanced_AttributeSetInvalid()	72
4.11.1.5	DTAdvanced_DISABLED_AttributeSetInvalidMapLength()	73
4.11.1.6	DTAdvanced_AttributeSetAsyncInvalid()	73
4.11.1.7	DTAdvanced_AttributeSet()	73
4.11.1.8	DTAdvanced_BroadcastReceive()	73
4.11.1.9	main()	74
4.11.2	Variable Documentation	74
4.11.2.1	domain	74
4.11.2.2	testAddress	74
4.11.2.3	connectionIdService	74
4.11.2.4	connectionIdClient	74
4.11.2.5	tasync	74
4.12	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/DTCCombined.cpp File Reference	74
4.12.1	Function Documentation	75
4.12.1.1	DTCCombined_SendAndReceive()	75
4.12.1.2	DTCCombined_CheckInitialValue()	75
4.12.1.3	DTCCombined2_VariantWithLiteralEnum()	75
4.12.1.4	main()	75
4.12.2	Variable Documentation	76
4.12.2.1	domain	76
4.12.2.2	testAddress	76
4.12.2.3	connectionIdService	76
4.12.2.4	connectionIdClient	76
4.13	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/DTCConstants.cpp File Reference	76
4.13.1	Function Documentation	76
4.13.1.1	DTCConstants_InterfaceConstants()	77
4.13.1.2	DTCConstants_TypeCollectionConstants()	77
4.13.1.3	main()	77
4.14	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/DTDeployment.cpp File Reference	77
4.14.1	Function Documentation	77
4.14.1.1	DTDeployment_TryGetNoSubscriptionAttributeWithGetterIDSetToZeroInDeployment()	78
4.14.1.2	DTDeployment_TryGetAttributeWithGetterIDSetToZeroInDeployment()	78
4.14.1.3	main()	78
4.14.2	Variable Documentation	78
4.14.2.1	domain	78
4.14.2.2	testAddress	79
4.14.2.3	connectionIdService	79

4.14.2.4	connectionIdClient	79
4.15	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↔ commonapi.core.verification/src/DTDerived.cpp File Reference	79
4.15.1	Function Documentation	79
4.15.1.1	DTDerived_SendAndReceive()	79
4.15.1.2	DTDerived_AttributeSet()	80
4.15.1.3	DTDerived_BroadcastReceive()	80
4.15.1.4	main()	80
4.15.2	Variable Documentation	80
4.15.2.1	domain	80
4.15.2.2	testAddress	80
4.15.2.3	connectionId_client	81
4.15.2.4	connectionId_service	81
4.15.2.5	tasync	81
4.16	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↔ commonapi.core.verification/src/DTPrimitive.cpp File Reference	81
4.16.1	Function Documentation	81
4.16.1.1	DTPrimitive_SendAndReceive()	82
4.16.1.2	DTPrimitive_AttributeSet()	82
4.16.1.3	DTPrimitive_BroadcastReceive()	82
4.16.1.4	DTPrimitive_EmptyBroadcastReceive()	82
4.16.1.5	DTPrimitive_RangedIntegers()	83
4.16.1.6	main()	83
4.16.2	Variable Documentation	83
4.16.2.1	domain	83
4.16.2.2	testAddress	83
4.16.2.3	connectionIdService	83
4.16.2.4	connectionIdClient	83
4.16.2.5	tasync	84
4.17	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↔ commonapi.core.verification/src/PFComplex.cpp File Reference	84
4.17.1	Function Documentation	84
4.17.1.1	PFComplex_Ping_Pong_Complex_Synchronous()	84
4.17.1.2	PFComplex_Ping_Pong_Complex_Asynchronous()	85
4.17.1.3	main()	85
4.17.2	Variable Documentation	85
4.17.2.1	usecPerSecond	85
4.17.2.2	serviceId	85
4.17.2.3	clientId	85
4.17.2.4	domain	86
4.17.2.5	testAddress	86
4.17.2.6	maxArraySize	86
4.17.2.7	loopCountPerPayload	86

4.18	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/PFPrimitive.cpp File Reference	86
4.18.1	Function Documentation	86
4.18.1.1	PFPrimitive_Ping_Pong_Primitive_Synchronous()	87
4.18.1.2	PFPrimitive_Ping_Pong_Primitive_Asynchronous()	87
4.18.1.3	main()	87
4.18.2	Variable Documentation	87
4.18.2.1	serviceld	87
4.18.2.2	clientId	88
4.18.2.3	domain	88
4.18.2.4	testAddress	88
4.18.2.5	usecPerSecond	88
4.18.2.6	maxPrimitiveArraySize	88
4.18.2.7	loopCountPerPayload	88
4.19	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/RTBuildProxiesAndStubs.cpp File Reference	88
4.19.1	Function Documentation	89
4.19.1.1	RTBuildProxiesAndStubs_LoadedRuntimeCanBuildProxiesAndStubs()	89
4.19.1.2	RTBuildProxiesAndStubs_BuildProxiesAndStubsTwoTimes()	89
4.19.1.3	RTBuildProxiesAndStubs_BuildProxyTwoTimesWithReassigningAndStub()	90
4.19.1.4	RTBuildProxiesAndStubs_WaitForProxyDestruction()	90
4.19.1.5	RTBuildProxiesAndStubs_WaitForProxyDestructionCreatedInThread()	90
4.19.1.6	RTBuildProxiesAndStubs_WaitForProxyDestructionInTwoThreads()	91
4.19.1.7	RTBuildProxiesAndStubs_BuildProxySubscribeToProxyStatusEventBlockingCallAndShutdown()	91
4.19.1.8	main()	91
4.19.2	Variable Documentation	91
4.19.2.1	domain	91
4.19.2.2	testAddress	92
4.19.2.3	applicationNameService	92
4.19.2.4	applicationNameClient	92
4.19.2.5	task	92
4.20	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/RTLoadingRuntime.cpp File Reference	92
4.20.1	Function Documentation	92
4.20.1.1	RTLoadingRuntime_LoadsDefaultRuntime()	92
4.20.1.2	main()	93
4.21	/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/StabilitySP.cpp File Reference	93
4.21.1	Function Documentation	93
4.21.1.1	StabilitySP_RepeatedRegistrations()	93
4.21.1.2	StabilitySP_MultipleMethodCalls()	94
4.21.1.3	StabilitySP_MultipleAttributeSets()	94
4.21.1.4	StabilitySP_MultipleAttributeGets()	94

4.21.1.5 StabilitySP_MultipleAttributeGetAsyncs()	95
4.21.1.6 StabilitySP_MultipleAttributeSetAsyncs()	95
4.21.1.7 StabilitySP_MultipleAttributeSubscriptions()	95
4.21.1.8 main()	96
4.21.2 Variable Documentation	96
4.21.2.1 serviceId	96
4.21.2.2 clientId	96
4.21.2.3 domain	96
4.21.2.4 testAddress	96
4.21.2.5 COMMONAPI_CONFIG_SUFFIX	96
4.21.2.6 MAXSERVERCOUNT	96
4.21.2.7 MAXTHREADCOUNT	97
4.21.2.8 MAXMETHODCALLS	97
4.21.2.9 MAXREGLOOPS	97
4.21.2.10 MAXREGCOUNT	97
4.21.2.11 MESSAGE_SIZE	97
4.21.2.12 MAXSUBSCRIPTIONSETS	97
4.22 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↵ commonapi.core.verification/src/THMainLoopIndependence.cpp File Reference	97
4.22.1 Function Documentation	98
4.22.1.1 THMainLoopIndependence_ProxyReceivesAnswerOnlyIfStubMainLoopRuns()	98
4.22.1.2 THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersSync()	98
4.22.1.3 THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersAsync()	98
4.22.1.4 main()	99
4.22.2 Variable Documentation	99
4.22.2.1 domain	99
4.22.2.2 instance6	99
4.22.2.3 instance7	99
4.22.2.4 instance8	99
4.22.2.5 mainloopName1	99
4.22.2.6 mainloopName2	99
4.22.2.7 thirdPartyServiceId	100
4.22.2.8 tasync	100
4.23 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↵ commonapi.core.verification/src/THMainLoopIntegration.cpp File Reference	100
4.23.1 Function Documentation	100
4.23.1.1 THMainLoopIntegration_VerifyCommunicationWithMainLoop()	100
4.23.1.2 THMainLoopIntegration_VerifyTransportReading()	101
4.23.1.3 THMainLoopIntegration_VerifySyncCallMessageHandlingOrder()	101
4.23.1.4 THMainLoopIntegration_SelectiveErrorHandlerWithMainLoop()	101
4.23.1.5 THMainLoopIntegration_AsynchronousMethodCallsReceiveNotAvailable()	102
4.23.1.6 THMainLoopIntegration_CreateProxyToManagerInSameProcess()	102
4.23.1.7 main()	102

4.23.2 Variable Documentation	102
4.23.2.1 domain	102
4.23.2.2 instance	102
4.23.2.3 connection_client	102
4.23.2.4 connection_service	103
4.23.2.5 tasync	103
4.24 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.↵ commonapi.core.verification/src/THMainLoopTwoThreads.cpp File Reference	103
4.24.1 Function Documentation	103
4.24.1.1 THMainLoopTwoThreads_ProxyGetsAvailableStatus()	103
4.24.1.2 THMainLoopTwoThreads_ProxyGetsFunctionResponse()	103
4.24.1.3 main()	104
4.24.2 Variable Documentation	104
4.24.2.1 domain	104
4.24.2.2 instance	104
Index	105

Chapter 1

CommonAPITests

Copyright (C) 2015 BMW AG

- This file is part of GENIVI project IPC CommonAPI C++.

Contributions are licensed to the GENIVI Alliance under one or more Contribution License Agreements.

This document

This document provides a list of tests which are implemented in the project `+org.genivi.commonapi.core.\leftrightarrow verification+` which is part of CommonAPI-Tools. These tests are middleware independent and can be used to verify the correct implementation of middleware specific bindings.

About IPC CommonAPI C++

IPC CommonAPI C++ is a C++ based abstraction API for communication stacks, which enables applications to use different communication middleware - so called language bindings - as backend without any changes to the application code.

More information

can be found at the [project homepage](#)
Please see the [project download section](#) for available language bindings.

Chapter 2

Test List

Global [AFExtended_Attributes](#) ()

Check that attributes work through extended interfaces

Global [AFExtended_Broadcast](#) ()

Test broadcasts. Subscribe to a broadcast, and see that the value is correctly received.

Global [AFExtended_MethodCall](#) ()

Check that method calls work through extended interfaces

Global [AFManaged_AddRemoveManagedInterfaceMultiple](#) ()

Subscribe on the events about availability status changes at the manager

- Add a managed interface to the manager
- Check that the client is notified about the newly added interface
- Add a second instance of the same managed interface to the manager
- Check that the client is notified about the newly added interface
- Remove all the managed interfaces from the manager
- Check that the client is notified about the removed interfaces

Global [AFManaged_AddRemoveManagedInterfaceSingle](#) ()

Subscribe on the events about availability status changes at the manager

- Add a managed interface to the manager
- Check that the client is notified about the newly added interface
- Remove the managed interface from the manager
- Check that the client is notified about the removed interface

Global [AFManaged_AddRemoveMultipleManagedInterfacesMultiple](#) ()

Add a managed interface to the manager

- Check that the client is notified about the newly added interface
- Add a different managed interface to the manager
- Check that the client is notified about the newly added interface
- Add a second instance of the same managed interface to the manager
- Check that the client is notified about the newly added interface
- Remove all the managed interfaces from the manager
- Check that the client is notified about the removed interfaces

Global [AFManaged_AddRemoveMultipleManagedInterfacesMultipleProxyNotActive](#) ()

- Global AFManaged_AddRemoveMultipleManagedInterfacesSingle ()** .
- Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Add a different managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Remove all the managed interfaces from the manager
 - Check that the client is notified about the removed interfaces
- Global AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicit ()** .
- Subscribe on the events about availability status changes at the manager
 - Add managed interfaces to the manager
 - Check that the client is notified about the newly added interfaces
 - Build proxies through the manager to the managed interfaces
 - Call a method on the managed interfaces and check call status
 - Explicitly deregister managed interfaces through their instance name
- Global AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicitAll ()** .
- Subscribe on the events about availability status changes at the manager
 - Add managed interfaces to the manager
 - Check that the client is notified about the newly added interfaces
 - Build proxies through the manager to the managed interfaces
 - Call a method on the managed interfaces and check call status
 - Deregister all managed interfaces through manager's stub adapter
- Global AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationImplicit ()** .
- Subscribe on the events about availability status changes at the manager
 - Add managed interfaces to the manager
 - Check that the client is notified about the newly added interfaces
 - Build proxies through the manager to the managed interfaces
 - Call a method on the managed interfaces and check call status
 - Don't deregister managed interfaces. This is done in dtor of manager's StubAdapterInternal when manager service is unregistered in TearDown() method.
- Global AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicit ()** .
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Build a proxy through the manager to the managed device
 - Call a method on the managed device and check call status
 - Explicitly deregister managed interface through its instance name
- Global AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicitAll ()** .
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Build a proxy through the manager to the managed device
 - Call a method on the managed device and check call status
 - Deregister all managed interfaces through manager's stub adapter
- Global AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationImplicit ()** .
- Subscribe on the events about availability status changes at the manager

- Add a managed interface to the manager
- Check that the client is notified about the newly added interface
- Build a proxy through the manager to the managed device
- Call a method on the managed device and check call status
- Don't deregister managed interfaces. This is done in dtor of manager's StubAdapterInternal when manager service is unregistered in TearDown() method.

Global AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallInProxyStatusEventSingleDeregistration() .

- Subscribe on the events about availability status changes at the manager
- Add a managed interface to the manager
- Check that the client is notified about the newly added interface
- Build a proxy through the manager to the managed device inside the availability event/callback
- Subscribe to the proxy status event
- Call a method on the managed device and check call status inside the proxy status event/callback (status == CommonAPI::AVAILABILITY_STATUS::AVAILABLE)
- Remove and add the managed interface to the manager a few times
- Explicitly deregister managed interface through its instance name

Global AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallSingleDeregistrationExplicit() .

- Subscribe on the events about availability status changes at the manager
- Add a managed interface to the manager
- Check that the client is notified about the newly added interface
- Build a proxy through the manager to the managed device inside the availability event/callback
- Call a method on the managed device and check call status inside the availability event/callback
- Explicitly deregister managed interface through its instance name

Global AFManaged_CreateProxyToManagerInSameProcess()

Offer a interface manager and build two proxies to it. One proxy uses the same connection as the manager while the other uses a different connection. Check that both proxies get available and receive a available event

Global AFManaged_DeleteManagerProxyInsideProxyStatusEventCallbackAndMethodCall() .

- Subscribe to the proxy status event of the manager
- Subscribe on the events about availability status changes at the manager
- Add the managed interfaces to the manager
- Check that the client is notified about the newly added interfaces
- Unregister manager service
- Explicitly delete the proxy of the manager inside the proxy status event callback
- Register manager service and build new proxy (Setup())
- Subscribe on the events about availability status changes at the manager
- Add a managed interface to the manager
- Check that the client is notified about the newly added interface
- Build a proxy through the manager to the managed device
- Call a method on the managed device and check call status
- TearDown()

Global AFManaged_ProxyAddRemoveManagedInterfaceSingle() .

- Subscribe on the events about availability status changes at the manager
- Add a managed interface to the manager
- Check that the client is notified about the newly added interface

- Remove the managed interface from the manager
- Check that the client is notified about the removed interface

Global **AFManaged_ProxyManagerTestGetInstanceAvailabilityStatusAsync ()** .

Add a managed interface to the manager

- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
- Use the ProxyManager's checkInstanceAvailabilityStatusAsync method to check that all returned instances by getAvailableInstances are available
- Add a different managed interface to the manager
- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
- Use the ProxyManager's checkInstanceAvailabilityStatusAsync method to check that all returned instances by getAvailableInstances are available
- Add a second instance of the same managed interface to the manager
- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
- Use the ProxyManager's checkInstanceAvailabilityStatusAsync method to check that all returned instances by getAvailableInstances are available
- Remove all the managed interfaces from the manager
- Check that the client is notified about the removed interfaces

Global **AFManaged_ProxyManagerTestNonPrimitiveMethodsAsync ()** .

Add a managed interface to the manager

- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstancesAsync method to check that all registered instances are returned
- Add a different managed interface to the manager
- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstancesAsync method to check that all registered instances are returned
- Add a second instance of the same managed interface to the manager
- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstancesAsync method to check that all registered instances are returned
- Remove all the managed interfaces from the manager
- Check that the client is notified about the removed interfaces

Global **AFManaged_ProxyManagerTestNonPrimitiveMethodsSync ()** .

Add a managed interface to the manager

- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
- Use the ProxyManager's checkInstanceAvailabilityStatus method to check that all returned instances by getAvailableInstances are available
- Add a different managed interface to the manager
- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
- Use the ProxyManager's checkInstanceAvailabilityStatus method to check that all returned instances by getAvailableInstances are available
- Add a second instance of the same managed interface to the manager

- Check that the client is notified about the newly added interface
- Use the ProxyManager's `getAvailableInstances` method to check that all registered instances are returned
- Use the ProxyManager's `checkInstanceAvailabilityStatus` method to check that all returned instances by `getAvailableInstances` are available
- Remove all the managed interfaces from the manager
- Check that the client is notified about the removed interfaces

Global **AFManaged_ProxyManagerTestPrimitiveMethods ()** .

Test the `getConnectionId`, `getDomain` and `getInterface` methods available via the ProxyManager of the respective managed interfaces of the manager

Global **AFPolymorph_Broadcast ()** .

Call a method with a special value that tells the stub to send a broadcast signal

- verify that the received data matches the transmitted data

Global **AFPolymorph_MethodCall ()** .

Call a method whose input and output parameters are polymorphic structures

- verify that the received data matches the transmitted data

Global **AFPolymorph_SetAndGetAttributeDoublyUsedBaseStruct ()** .

Set and get an attribute through a polymorphic structure whose Base is also used by another identical structure.

- verify that the received data matches the transmitted data

Global **AFPolymorph_SetAndGetAttributeEnum ()** .

Set and get a enum-type attribute through a polymorphic structure

- verify that the received data matches the transmitted data

Global **AFPolymorph_SetAndGetAttributeString ()** .

Set and get a string-type attribute through a polymorphic structure

- verify that the received data matches the transmitted data

Global **AFPolymorph_SetAndGetAttributeStruct ()** .

Set and get a struct-type attribute through a polymorphic structure

- verify that the received data matches the transmitted data

Global **AFPolymorph_SetAndGetAttributeTypedef ()** .

Set and get a typedef-type attribute through a polymorphic structure

- verify that the received data matches the transmitted data

Global **AFPolymorph_SetAndGetAttributeUInt ()** .

Set and get a uint-type attribute through a polymorphic structure

- verify that the received data matches the transmitted data

Global **AFSelective_ProxyBuildAndDestroy ()**

Test multiple selective broadcasts, with rejection.

- subscribe to stub three times: once from proxy2, once from proxy1 (accepted) once from proxy2 (rejected)
- This should result with two subscription callbacks being called from broadcast.

Test Destruction of Proxies but service stay online There were an issue when a proxy which has never subscribed gets destructed with SomeIP binding (GLIPCI-1081). Therefore i added this test case.

Global **AFSelective_SelectiveBroadcast ()**

Test selective broadcasts.

- inform stub to start accepting subscriptions
- subscribe to the selective broadcast
- check that no error was received (in a reasonable time)

- inform stub to send a broadcast
- check that a correct value is received

Global **AFSelective_SelectiveBroadcastRejected ()**

Test selective broadcasts.

- inform stub to stop accepting subscriptions
- try to subscribe to the selective broadcast
- check that an error was received
- inform stub to send a broadcast
- check that nothing was received in a reasonable time

Global **AFSelective_SelectiveMultiBroadcast ()**

Test multiple selective broadcasts.

- inform stub to start accepting subscriptions
- subscribe to the selective broadcast
- check that no error was received (in a reasonable time)
- inform stub to send a broadcast
- check that a correct value is received

Global **CMAAttributes_AttributeGetAsynchronous ()**

Test asynchronous getValue API function for attributes with combinations of additional properties readonly and noSubscriptions (testAttribute, testA readonly, testB noSubscriptions, testC readonly noSubscriptions).

- Set attribute to certain value on stub side.
- Call getValue.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if value of is equal to expected value.

Global **CMAAttributes_AttributeGetSynchronous ()**

Test synchronous getValue API function for attributes with combinations of additional properties readonly and noSubscriptions (testAttribute, testA readonly, testB noSubscriptions, testC readonly noSubscriptions).

- Set attribute to certain value on stub side.
- Call getValue.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if value of is equal to expected value.

Global **CMAAttributes_AttributeSetAsynchronous ()**

Test asynchronous setValue API function for attributes with combinations of additional properties readonly and noSubscriptions (testAttribute, testB noSubscriptions).

- Set attribute to certain value on proxy side.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if returned value of setValue is equal to expected value.

Global **CMAAttributes_AttributeSetSynchronous ()**

Test synchronous setValue API function for attributes with combinations of additional properties readonly and noSubscriptions (testAttribute, testB noSubscriptions)

- Set attribute to certain value on proxy side.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if returned value of setValue is equal to expected value.

Global **CMAAttributes_AttributeSubscription ()**

Test subscription API function for attributes

Global `CMAAttributeSubscription_DISABLED_SubscribeAndUnsubscribeImplicitWithCreatingNewProxyWithReassigning` ()

Test of subscribing and unsubscribing implicit with creating a new proxy with reassigning

- subscribe first callback
- subscribe second callback
- change value
- check that both callbacks were executed by changing the value
- create new proxy with reassigning. So the connection won't be destroyed and the callbacks are unsubscribed implicitly.
- subscribe second callback
- change value
- check that only second callback was executed
- unsubscribe second callback
- change value
- check that both callbacks were not executed by changing the value

Global `CMAAttributeSubscription_SubscribeAndUnsubscribeAndReSubscribe` ()

Test of behaviour in case subscribe, unsubscribe and resubscribe is done

- set default value
- register service
- subscribe for the attribute
- current value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- proxy unsubscribes for the attribute
- value of attribute is not changed
- value received by proxy is reset to 0
- proxy resubscribes for the attribute
- current value must be communicated to the proxy
- value received must be equal to value received before last unsubscribe call
- unregister service

Global `CMAAttributeSubscription_SubscribeAndUnsubscribeSequentially` ()

Test of subscribing and immediately unsubscribing a callback

- subscribe first callback
- subscribe second callback
- unsubscribe second callback
- change value
- check that only first callback was executed

Test of subscribing and unsubscribing sequentially

- subscribe first callback
- subscribe second callback
- change value
- check that both callbacks were executed by changing the value
- unsubscribe first callback
- change value

- check that only second callback was executed
- unsubscribe second callback
- change value
- check that both callbacks were not executed by changing the value

Global **CMAAttributeSubscription_SubscribeAndUnsubscribeTwoCallbacksCoexistent ()**

Test of subscribe and unsubscribe with two coexistent callbacks

- subscribe both callbacks
- change value
- check that both callbacks were executed by changing the value
- unsubscribe both callbacks
- change value
- check that both callbacks were not executed by changing the value

Global **CMAAttributeSubscription_SubscribeAndUnsubscribeUnsubscribe ()**

Test of behaviour in case unsubscribe is called two times

- set default value
- register service
- subscribe for the attribute
- current value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- proxy unsubscribes for the attribute
- value of attribute is changed
- changed value must not be communicated to the proxy
- proxy unsubscribes again for the attribute
- value of attribute is changed
- changed value must not be communicated to the proxy
- unregister service

Global **CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribe ()**

Test of behaviour in case subscribe and unsubscribe is done for multiple proxys on the same attribute and afterwards all proxys resubscribe

- set default value
- register service
- subscribe for the attribute with proxyA
- subscribe for the attribute with proxyB
- current value must be communicated to the proxyA
- current value must be communicated to the proxyB
- value of attribute is changed
- changed value must be communicated to the proxyA
- changed value must be communicated to the proxyB
- proxyA and proxy B unsubscribe for the attribute
- value of attribute is not changed
- value received is reset to 0
- proxyA and proxyB resubscribe for the attribute
- current value must be communicated to the proxy as initial value

- value received must be equal to value received before last unsubscribe call
- unregister service

Global **CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribeSameEventgroup ()**

Test of behaviour in case subscribe and unsubscribe is done for multiple proxys on attributes in the same event-group and afterwards all proxys resubscribe

- set default value
- register service
- subscribe for the attribute 1 with proxyA
- subscribe for the attribute 2 with proxyB
- current value must be communicated to the proxyA
- current value must be communicated to the proxyB
- value of attribute is changed
- changed value must be communicated to the proxyA
- changed value must be communicated to the proxyB
- proxyA and proxy B unsubscribe for the attributes
- value of attributes is not changed
- value received is reset to 0
- proxyA and proxyB resubscribe for the attribute 1 and 2
- current value must be communicated to the proxys as initial value
- value received must be equal to value received before last unsubscribe call
- unregister service

Global **CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeOneResubscribeSameEventgroup ()**

Test of behaviour in case two proxys A and B subscribe to events that are in one eventgroup, proxyB unsubscribes, and proxy A is still expected to receive changed values.

- set default value
- register service
- subscribe for the attribute with proxyA
- subscribe for the attribute with proxyB
- current value must be communicated to the proxyA
- current value must be communicated to the proxyB
- value of attribute is changed
- changed value must be communicated to the proxyA
- changed value must be communicated to the proxyB
- proxyB unsubscribes for the attribute
- value of attribute is changed
- value received must be equal to changed value for proxy A
- unregister service

Global **CMAAttributeSubscription_SubscribeSecondProxyLater ()**

Test of subscribing a second proxy a little bit later

- proxy subscribes for an attribute of the service
- register service
- initial value must be communicated to the proxy
- create a second proxy
- second proxy subscribes for the same attribute of the service

- current attribute value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to both proxies
- unregister service

Global `CMAAttributeSubscription_SubscribeServiceNotAvailable ()`

Test of subscribing in case that service is not available

- set default value
- subscribe for the attribute
- no value is communicated to the proxy
- register service
- current value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- unregister service

Global `CMAAttributeSubscription_SubscribeThreeCallbacksServiceAvailable ()`

Test of subscribing three callbacks after registering the service

- register service
- proxy subscribes three callbacks for an attribute of the service
- initial value must be communicated to every callback

Global `CMAAttributeSubscription_SubscribeThreeCallbacksServiceNotAvailable ()`

Test of subscribing three callbacks before registering the service

- proxy subscribes three callbacks for an attribute of the service
- register service
- initial value must be communicated to every callback

Global `CMAAttributeSubscription_SubscribeUnregisterNoValueSetRegisterService ()`

Test of unregister a service in case a proxy is subscribed for an attribute of this service. During the unregistered time of the service the value of the attribute is not changed.

- register service
- proxy subscribes for an attribute of the service
- value of attribute is set
- changed value must be communicated to the proxy
- unregister service
- register service
- current attribute value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- unregister service

Global `CMAAttributeSubscription_SubscribeUnregisterSetValueRegisterService ()`

Test of unregister a service in case a proxy is subscribed for an attribute of this service. During the unregistered time of the service the value of the attribute is changed.

- register service
- proxy subscribes for an attribute of the service
- value of attribute is set
- changed value must be communicated to the proxy

- unregister service
- value of attribute is changed
- changed value must not be communicated to the proxy
- register service
- current attribute value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- unregister service

Global **CMAAttributeSubscription_SubscriptionMultithreading ()**

Subscription test with several threads.

- Start several threads.
- The threads subscribe for the availability status.
- The available-callback subscribes for TestAttribute if service is available for proxy and
- unsubscribes if service is not available for proxy.
- Change attribute in service by set method; the new attribute value should be received by all the threads.
- The new value is written into a queue.
- Check if the values of each thread are written into the queue.

Global **CMAAttributeSubscription_SubscriptionOnAvailable ()**

Subscription test with subscription on available-event.

- Subscribe for available-event.
- Available-callback subscribes for TestPredefinedTypeAttribute if service is available for proxy and unsubscribes if service is not available for proxy.
- Change attribute in service by set method; the new attribute value should be received by the proxy because the service is not registered.
- Register service and change value again; the value should now be received.
- Unregister and change value again.

Global **CMAAttributeSubscription_SubscriptionStandard ()**

Subscription standard test.

- Register service and check if proxy is available.
- Proxy subscribes for TestAttribute (uint8_t).
- Change attribute in service several times by set method.
- Callback function in proxy writes the received values in a queue.
- Check if values in the queue are the same as the values that were set in the service.
- Unregister test service.

Global **CMAAttributeSubscription_SubscriptionUnsubscribeFromCallback ()**

Subscription test : unsubscribe from the subscription callback.

- Register service and check if proxy is available.
- Proxy subscribes for TestAttribute (uint8_t).
- Change attribute in service by set method.
- Check if callback function in proxy received the right value.
- Change value to the magic value 99: this triggers the callback to unsubscribe.
- Change value again; the callback should now be called anymore.
- Unregister the test service.

Global [CMBlockingCalls_BlockInAvailabilityHandler \(\)](#)

Register availability handler which blocks and (de)register the corresponding service multiple times. After the service stays available do a method call and check that the answer is received

Global [CMBlockingCalls_BlockInAvailabilityHandlerAndReceiveCallbacks \(\)](#)

Create proxy to service and wait until it is reported as available via a registered availability handler. As soon as it is available start sending requests to the service and wait for its replies. Check that the replies for this requests are dispatched even if the availability handler for this service is blocked. This is tested through blocking in the availability handler after the main thread was notified about the services' availability

Global [CMBlockingCalls_BlockInProxyCallback \(\)](#)

Call test method and block in registered callback when processing responses. Check that all responses are delivered.

Global [CMBlockingCalls_BlockInStubMethod \(\)](#)

Call test method which generates blocking calls on stub side and check if answers are received.

Global [CMBlockingCalls_NestedBlockInStubMethods \(\)](#)

Call test method which generates blocking calls on stub. Ensure working dispatching even if main dispatch thread still blocked after a dispatch thread was spawned and joined again because another dispatch thread returned from the usercode in the meanwhile.

Global [CMBroadcasts_BroadcastStubGoesOfflineOnlineAgain \(\)](#)

Test BroadcastStubGoesOfflineOnlineAgain.

- service offline
- subscribe to broadcast
- service online
- fire broadcast -> proxy should receive
- service offline
- service online
- fire again -> proxy should receive again

Global [CMBroadcasts_NormalBroadcast \(\)](#)

Test broadcasts. Subscribe to a broadcast, and see that the value is correctly received.

Global [CMBroadcasts_SelectiveBroadcast \(\)](#)

Test selective broadcasts.

- inform stub to start accepting subscriptions
- subscribe to the selective broadcast
- check that no error was received (in a reasonable time)
- inform stub to send a broadcast
- check that a correct value is received

Global [CMBroadcasts_SelectiveBroadcastRejected \(\)](#)

Test selective broadcasts.

- inform stub to stop accepting subscriptions
- try to subscribe to the selective broadcast
- check that an error was received
- inform stub to send a broadcast
- check that nothing was received in a reasonable time

Global [CMBroadcasts_SelectiveBroadcastStubGoesOfflineOnlineAgain \(\)](#)

Test SelectiveBroadcastStubGoesOfflineOnlineAgain.

- service offline

- subscribe to selective broadcast
- service online
- fire selective broadcast -> proxy should receive
- service offline
- service online
- fire again -> proxy should receive again

Global **CMMethodCalls_AsynchronousMethodCall ()**

Call test method asynchronous and check call status.

- Test stub sets in-value of test method.
- Make asynchronous call of test method.
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored value in callback function.

Global **CMMethodCalls_AsynchronousMethodCallProxyBecomesAvailable ()**

Call test method asynchronous when proxy is not available. Proxy becomes available during call.

- Unregister service
- Wait that proxy is not available.
- Test stub sets in-value of test method.
- Set timeout of asynchronous call.
- Make asynchronous call of test method.
- Proxy becomes available during call.
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored value in callback function.

Global **CMMethodCalls_AsynchronousMethodCallProxyNotAvailable ()**

Call test method asynchronous when proxy is not available.

- Unregister service.
- Wait that proxy is not available.
- Test stub sets in-value of test method.
- Set timeout of asynchronous call.
- Make asynchronous call of test method.
- Do checks of call status (CommonAPI::CallStatus::NOT_AVAILABLE) and that timeout occurred.

Global **CMMethodCalls_AsynchronousMethodCallProxyNotAvailableDeleteProxy ()**

Call test method asynchronous when proxy is not available and delete proxy.

- Unregister service.
- Wait that proxy is not available.
- Test stub sets in-value of test method.
- Set timeout of asynchronous call.
- Make asynchronous call of test method.
- Start thread which deletes the proxy.
- Check if proxy could be deleted.
- Join created thread.

Global **CMMethodCalls_AsynchronousMethodCallsProxyBecomesAvailable ()**

Call test method asynchronous multiple times when proxy is not available. Proxy becomes available during call

- Unregister service
- Wait that proxy is not available
- Test stub set in-value of test methods.

- Set timeouts of asynchronous calls (timeouts that are reached and timeouts that are not reached).
- Make asynchronous calls of test method (2 expected timeouts, 3 successful calls).
- Proxy becomes available during call
- Do checks of call status (CommonAPI::CallStatus::SUCCESS and CommonAPI::CallStatus::NOT_AVAILABLE for expected timeouts), stored values and timeouts that occurred in callback functions.

Global **CMMethodCalls_AsynchronousMethodCallsReceiveNotAvailable ()**

Call test method via two proxies multiple times asynchronously while the service is unavailable and check if the provided callback is called with an error for every method call done.

Global **CMMethodCalls_FireAndForget ()**

Call fire and forget method and check via broadcast that value was received.

- Subscribe to broadcast
- Check that broadcast subscription succeeded
- Make fire and forget method call
- Check via broadcast that value was correctly received (Stub fires broadcast when value was received).

Global **CMMethodCalls_NestedAsynchronousMethodCall ()**

Call test method asynchronous and call test method asynchronous in callback (nested).

- Test stub sets in-values of test methods.
- Make asynchronous call of test method.
- Make asynchronous call of test method in callback (nested).
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored values in callback functions.

Global **CMMethodCalls_NestedAsynchronousMethodCallProxyBecomesAvailable ()**

Call test method asynchronous and call test method asynchronous in callback (nested) when proxy is not available. Proxy becomes available during call.

- Unregister service
- Wait that proxy is not available.
- Test stub sets in-values of test methods.
- Set timeout of asynchronous calls.
- Make asynchronous call of test method.
- Make asynchronous call of test method in callback (nested).
- Proxy becomes available during first async call.
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored value in callback functions.

Global **CMMethodCalls_NestedAsynchronousMethodCallProxyNotAvailable ()**

Call test method asynchronous and call test method asynchronous in callback (nested) when proxy is not available.

- Unregister service.
- Wait that proxy is not available.
- Test stub sets in-value of test methods.
- Set timeout of asynchronous calls.
- Make asynchronous call of test method.
- Make asynchronous call of test method in callback (nested).
- Do checks of call status (CommonAPI::CallStatus::NOT_AVAILABLE) and that timeouts occurred.

Global **CMMethodCalls_NestedAsynchronousMethodCallsTimedOut ()**

Call test method timeout asynchronous and call test method timeout asynchronous in callback (nested).

- Register second service with other instance

- Create second proxy to second service
- Make asynchronous call of test method timeout (first proxy)
- Make asynchronous call of test method timeout (second proxy)
- Check in callbacks if timeout occurred (CommonAPI::CallStatus::REMOTE_ERROR)
- Make asynchronous calls of test method timeout in callbacks as long as timeoutCalls_ < maxTimeout_↵ Calls_ (nested).
- Check if the same amount of timeouts occurred as async calls were done

Global **CMMethodCalls_NestedSynchronousMethodCall** ()

Call test method asynchronous and call test method synchronous in callback (nested).

- Test stub sets in-values of test methods.
- Make asynchronous call of test method.
- Make asynchronous call of test method in callback (nested).
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored values in callback functions.

Global **CMMethodCalls_SynchronousMethodCall** ()

Call test method synchronous and check call status.

- Test stub sets in-value of test method equal out-value of test method.
- Make synchronous call of test method.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if out value of test method is equal to in value.

Global **DTAdvanced_AttributeSet** ()

Test attribute functions with advanced types

- Call set function of attributes with advanced types
- Call get function and check if the return value is the same

Global **DTAdvanced_AttributeSetAsyncInvalid** ()

Test attribute asynchronous functions with invalid values

- Call set async function of attributes with invalid types
- Callback should be called with error status
- Check that attribute value has not changed

Global **DTAdvanced_AttributeSetInvalid** ()

Test attribute functions with invalid values

- Call set function of attributes with invalid types
- Check that the attribute's value has not changed

Global **DTAdvanced_BroadcastReceive** ()

Test broadcast with advanced types

- Subscribe to broadcast which contains advanced types
- Call function to cause the stub to fire broadcast event with the same content
- Check if the values in the callback function are as expected

Global **DTAdvanced_DISABLED_AttributeSetInvalidMapLength** ()

Test attribute functions with invalid map length

- Call set function of attributes with map length
- Check that an error returns

Global **DTCombined_CheckInitialValue** ()

Test that combined types are properly initialized

Global [DTCombined_SendAndReceive \(\)](#)

Test function call with combined type

- The combined type is one structure with combinations of advanced and primitive types
- Function call of a function that has for each advanced type one argument (test values) and one return value
- The stub copies the test values to the return values
- On client side the test values are compared with the return values

Global [DTConstants_InterfaceConstants \(\)](#)

See that we can access constants in an interface and that they have correct values

Global [DTConstants_TypeCollectionConstants \(\)](#)

See that we can access constants in type collection and that they have correct values

Global [DTDeployment_TryGetAttributeWithGetterIDSetToZeroInDeployment \(\)](#)

Test Try to get attribute deployed with GetterID=0

- Subscribe to changed event of attribute
- Set value to attribute via stub
- Make sure subscription handler was called
- Set value to attribute via proxy
- Make sure subscription handler was called
- Check via stub that proxy set correct value
- Try to get Attribute via proxy and make sure CallStatus::NOT_AVAILABLE is returned

Global [DTDeployment_TryGetNoSubscriptionAttributeWithGetterIDSetToZeroInDeployment \(\)](#)

Test Try to get noSubscription attribute deployed with GetterID=0 and NotifierID=0

- Set value to attribute via stub
- Set value to attribute via proxy
- Check via stub that proxy set correct value
- Try to get Attribute via proxy and make sure CallStatus::NOT_AVAILABLE is returned

Global [DTDerived_AttributeSet \(\)](#)

Test attribute functions with derived types

- Call set function of attributes with derived types
- Call get function and check if the return value is the same

Global [DTDerived_BroadcastReceive \(\)](#)

Test broadcast with derived types

- Subscribe to broadcast which contains derived types
- Call function to cause the stub to fire broadcast event with the same content
- Check if the values in the callback function are as expected

Global [DTPrimitive_AttributeSet \(\)](#)

Test attribute functions with primitive types

- Call set function of attributes with primitive types
- Call get function and check if the return value is the same

Global [DTPrimitive_BroadcastReceive \(\)](#)

Test broadcast with primitive types

- Subscribe to broadcast which contains primitive types
- Call function to cause the stub to fire broadcast event with the same content

- Check if the values in the callback function are as expected

Global **DTPrimitive_EmptyBroadcastReceive ()**

Test broadcast with empty broadcast

- Subscribe to broadcast which does not contain any datatypes
- Call function twice to cause the stub to fire a broadcast event
- Check if the callback function was called twice

Global **DTPrimitive_RangedIntegers ()**

Test ranged integer functionality

Global **DTPrimitive_SendAndReceive ()**

Test function call with primitive types

- Primitive types are: uint8_t, int8_t, uint16_t, int16_t, uint32_t, int32_t, uint64_t, int64_t, bool, float, double, std::string, ByteBuffer
- Function call of a function that has for each primitive type one argument (test values) and one return value
- The stub copies the test values to the return values
- On client side the test values are compared with the return values

Global **PFComplex_Ping_Pong_Complex_Asynchronous ()**

Test asynchronous ping pong function call

- complex array is array of a struct containing an union and another struc with primitive datatypes
- The stub just set (copies) the in array to the out array
- Only the CallStatus will be used to verify the async call has succeeded
- Using double payload every cycle, starting with 1 end with maxPrimitiveArraySize
- Doing loopCountPerPayload loops to calc the mean time

Global **PFComplex_Ping_Pong_Complex_Synchronous ()**

Test synchronous ping pong function call

- complex array is array of a struct containing an union and another struc with primitive datatypes
- The stub just set the in array to the out array
- CallStatus and array content will be used to verify the sync call has succeeded
- Using double payload every cycle, starting with 1 end with maxPrimitiveArraySize
- Doing primitiveLoopSize loops to build the mean time

Global **PFPrimitive_Ping_Pong_Primitive_Asynchronous ()**

Test asynchronous ping pong function call

- primitive array is array of UInt_8
 - The stub just set (copies) the in array to the out array
 - Only the CallStatus will be used to verify the async call has succeeded
 - Using double payload every cycle, starting with 1 end with maxPrimitiveArraySize
 - Doing primitiveLoopSize loops to build the mean time

Global **PFPrimitive_Ping_Pong_Primitive_Synchronous ()**

Test synchronous ping pong function call

- primitive array is array of UInt_8
 - The stub just set the in array to the out array
 - CallStatus and array content will be used to verify the sync call has succeeded
 - Using double payload every cycle, starting with 1 end with maxPrimitiveArraySize
 - Doing primitiveLoopSize loops to build the mean time

Global [RTBuildProxiesAndStubs_BuildProxiesAndStubsTwoTimes \(\)](#)

Loads Runtime, creates proxy and stub/service two times.

- Calls CommonAPI::Runtime::get() and checks if return value is true
- Create stub and register service
- Create proxy
- Do some synchronous calls
- Unregister the service.
- Create stub and register service
- Create proxy
- Checks whether proxy is available
- Unregister the service

Global [RTBuildProxiesAndStubs_BuildProxySubscribeToProxyStatusEventBlockingCallAndShutdown \(\)](#)

Loads Runtime, creates proxy, subscribes to proxy status event, does a blocking call and shutdown

- Calls CommonAPI::Runtime::get() and checks if return value is true.
- Checks if test proxy with domain and test instance can be created.
- Subscribes to proxy status event and simulate a blocking call (simulated by sleep) when proxy is getting available
- Register the test service
- Initiate shutdown when blocking call was done
- Unregister test service
- Wait till proxy is getting unavailable
- Destroy proxy
- Wait till proxy was destroyed and proxy status event handler is finished

Global [RTBuildProxiesAndStubs_BuildProxyTwoTimesWithReassigningAndStub \(\)](#)

Loads Runtime, creates proxy two times with reassigning and create stub/service.

- Calls CommonAPI::Runtime::get() and checks if return value is true
- Create proxy
- Create proxy again and reassign
- Create stub and register service
- Checks whether proxy is available
- Do synchronous calls
- Unregister the service.

Global [RTBuildProxiesAndStubs_LoadedRuntimeCanBuildProxiesAndStubs \(\)](#)

Loads Runtime, creates proxy and stub/service.

- Calls CommonAPI::Runtime::get() and checks if return value is true.
- Checks if test proxy with domain and test instance can be created.
- Checks if test stub can be created.
- Register the test service.
- Unregister the test service.

Global [RTBuildProxiesAndStubs_WaitForProxyDestruction \(\)](#)

Loads Runtime, creates proxy and stub/service, await proxy destruction

- Calls CommonAPI::Runtime::get() and checks if return value is true.
- Checks if test proxy with domain and test instance can be created
- Checks if test stub can be created.

- Register the test service.
- Wait for service availability
- Unregister the test service.
- Wait for on future till proxy was destroyed after `std::shared_ptr<>` ref from thread was released

Global **RTBuildProxiesAndStubs_WaitForProxyDestructionCreatedInThread ()**

Loads Runtime, creates proxy and stub/service, await proxy destruction

- Calls `CommonAPI::Runtime::get()` and checks if return value is true.
- Checks if test proxy with domain and test instance can be created (in an own thread).
- Checks if test stub can be created.
- Register the test service.
- Wait for service availability on the test proxy in it's thread.
- Unregister the test service.
- Wait till proxy was destroyed when `std::shared_ptr<>` in thread has been released.

Global **RTBuildProxiesAndStubs_WaitForProxyDestructionInTwoThreads ()**

Loads Runtime, creates proxy and stub/service, await proxy destruction in two threads

- Calls `CommonAPI::Runtime::get()` and checks if return value is true.
- Checks if test proxy with domain and test instance can be created (in an own thread).
- Wait till proxy was destroyed when `std::shared_ptr<>` in threads
- Join the threads that have been waiting for proxy destruction

Global **RTLodingRuntime_LoadsDefaultRuntime ()**

Loads Default Runtime.

- Calls `CommonAPI::Runtime::get()`.
- Success if return value is true.

Global **StabilitySP_MultipleAttributeGetAsyncs ()**

Create a number of services and proxies and get attributes through them.

- Register `MAXSERVERCOUNT` addresses as services
 - Set the attribute for service, at the stub side.
- Create `MAXTHREADCOUNT` threads, each of which creates a proxy for each service address and then gets attributes `MAXMETHODCALLS` times for each asynchronously
- Each attribute is `MESSAGE_SIZE` bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the callbacks are not called correct number of times

Global **StabilitySP_MultipleAttributeGets ()**

Create a number of services and proxies and get attributes through them.

- Register `MAXSERVERCOUNT` addresses as services
 - Set the attribute for service, at the stub side.
- Create `MAXTHREADCOUNT` threads, each of which creates a proxy for each service address and then gets attributes `MAXMETHODCALLS` times for each.
- Each attribute is `MESSAGE_SIZE` bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the returned attribute from the server is not correct

Global **StabilitySP_MultipleAttributeSetAsyncs ()**

Create a number of services and proxies and set attributes through them.

- Register `MAXSERVERCOUNT` addresses as services

- Set the attribute for service, at the stub side.
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then sets attributes MAXMETHODCALLS times for each asynchronously
- Each attribute is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the callbacks are not called correct number of times

Global [StabilitySP_MultipleAttributeSets \(\)](#)

Create a number of services and proxies and set attributes through them.

- Register MAXSERVERCOUNT addresses as services
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then sets attributes MAXMETHODCALLS times to each.
- Each attribute is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the return attribute from the server is not correct

Global [StabilitySP_MultipleAttributeSubscriptions \(\)](#)

Create a number of services and proxies and set attributes through them.

- Register MAXSERVERCOUNT addresses as services
 - Set the attribute for service, at the stub side.
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then sets attributes MAXMETHODCALLS times for each asynchronously
- Each attribute is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the callbacks are not called correct number of times

Global [StabilitySP_MultipleMethodCalls \(\)](#)

Create a number of services and proxies and send messages through them.

- Register MAXSERVERCOUNT addresses as services
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then sends MAXMETHODCALLS messages to each.
- Each message is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the return message from the server is not correct

Global [StabilitySP_RepeatedRegistrations \(\)](#)

Register and unregister services in a loop.

- do MAXREGLOOPS times:
 - register MAXREGCOUNT addresses as services
 - unregister the addresses that were just registered
 - check the return code of each register/unregister call
 - test fails if any of the return codes are false

Global [THMainLoopIndependence_ProxyReceivesAnswerOnlyIfStubMainLoopRuns \(\)](#)

Proxy Receives Answer Only If Stub MainLoop Runs.

- start proxy in thread 1 and call testPredefinedTypeMethod
- proxy should not receive answer, if the stub mainloop does not run
- run mainloop of stub
- now the stub mainloop also runs, so the proxy should receive the answer

Global [THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersAsync \(\)](#)

Proxy Receives Just His Own Answers.

- start 2 proxies in own threads
- call test method in each proxy asynchronously
- now each proxy should have received the answer to his own request

Global [THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersSync \(\)](#)

Proxy Receives Just His Own Answers.

- start 2 proxies in own threads
- call test method in each proxy synchronously
- now each proxy should have received the answer to his own request

Global [THMainLoopIntegration_AsynchronousMethodCallsReceiveNotAvailable \(\)](#)

Call test method multiple times asynchronously while the service is unavailable and check if the provided callback is called with an error for every method call done.

Global [THMainLoopIntegration_CreateProxyToManagerInSameProcess \(\)](#)

Offer a interface manager and build two proxies to it. One proxy uses the same connection as the manager while the other uses a different connection. Check that both proxies get available and receive a available event

Global [THMainLoopIntegration_SelectiveErrorHandlerWithMainLoop \(\)](#)

Verifies SelectiveError Handler is called correctly when used with mainloop

- get proxy with available flag = true
- Subscribe for selective Event and register error handler
- Stub fires event upon subscription
- Check that subscription handler and error handler were both called once
- Unregister Service and register Service again
- Check that subscription error handler was called again after service went offline and came online again (resubscription took place) and that the event was received a second time

Global [THMainLoopIntegration_VerifyCommunicationWithMainLoop \(\)](#)

Verifies communication with Main Loop.

- get proxy with available flag = true
- generate big test data
- send synchronous test message

Global [THMainLoopIntegration_VerifySyncCallMessageHandlingOrder \(\)](#)

Verifies Synchronous Call Message Handling Order.

- get proxy with available flag = true
- subscribe for broadcast event
- generate 5 test broadcasts
- 5 broadcasts should arrive in the right order

Global [THMainLoopIntegration_VerifyTransportReading \(\)](#)

Verifies Transport Reading When Dispatching Watches.

- get proxy with available flag = true
- generate big test data
- send asynchronous test message
- dispatch dispatchSource: the message must not be arrived
- dispatch watches (reads transport).
- dispatch dispatchSources again: now the message must be arrived.

Global [THMainLoopTwoThreads_ProxyGetsAvailableStatus \(\)](#)

Proxy Receives Available when MainLoop Dispatched sourced out to other thread.

Global [THMainLoopTwoThreads_ProxyGetsFunctionResponse \(\)](#)

Proxy gets function response when MainLoop Dispatched sourced out to other thread.

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ AFExtended.cpp	27
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ AFManaged.cpp	29
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ AFPolymorph.cpp	38
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ AFSelective.cpp	42
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ CMAAttributes.cpp	45
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ CMAAttributeSubscription.cpp	48
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ CMBlockingCalls.cpp	59
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ CMBroadcasts.cpp	62
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ CMMethodCalls.cpp	65
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ DTAdvanced.cpp	71
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ DTCombined.cpp	74
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ DTConstants.cpp	76
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ DTDeployment.cpp	77
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ DTDerived.cpp	79
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ DTPrimitive.cpp	81
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ PFComplex.cpp	84
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ PFPrimitive.cpp	86
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi. core.verification/src/ RTBuildProxiesAndStubs.cpp	88

/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.↵ core.verification/src/ RTLodingRuntime.cpp	92
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.↵ core.verification/src/ StabilitySP.cpp	93
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.↵ core.verification/src/ THMainLoopIndependence.cpp	97
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.↵ core.verification/src/ THMainLoopIntegration.cpp	100
/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.↵ core.verification/src/ THMainLoopTwoThreads.cpp	103

Chapter 4

File Documentation

4.1 mainpagetests/01_mainpage.dox File Reference

4.2 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/AFExtended.cpp File Reference

Functions

- void [AFExtended_MethodCall](#) ()
- void [AFExtended_Attributes](#) ()
- void [AFExtended_Broadcast](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [serviceld](#) = "service-sample"
- const std::string [clientId](#) = "client-sample"
- const std::string [domain](#) = "local"
- const std::string [testAddressBase](#) = "commonapi.advanced.extended.AFExtendedBase"
- const std::string [testAddressOnce](#) = "commonapi.advanced.extended.AFExtendedOnce"
- const std::string [testAddressTwice](#) = "commonapi.advanced.extended.AFExtendedTwice"
- const int [tasync](#) = 10000

4.2.1 Function Documentation

4.2.1.1 AFExtended_MethodCall()

```
void AFExtended_MethodCall ( )
```

Test Check that method calls work through extended interfaces

4.2.1.2 AFExtended_Attributes()

```
void AFExtended_Attributes ( )
```

Test Check that attributes work through extended interfaces

4.2.1.3 AFExtended_Broadcast()

```
void AFExtended_Broadcast ( )
```

Test Test broadcasts. Subscribe to a broadcast, and see that the value is correctly received.

4.2.1.4 main()

```
int main (
    int argc,
    char ** argv )
```

4.2.2 Variable Documentation

4.2.2.1 serviceId

```
const std::string serviceId = "service-sample"
```

4.2.2.2 clientId

```
const std::string clientId = "client-sample"
```

4.2.2.3 domain

```
const std::string domain = "local"
```


4.2.2.4 testAddressBase

```
const std::string testAddressBase = "commonapi.advanced.extended.AFExtendedBase"
```

4.2.2.5 testAddressOnce

```
const std::string testAddressOnce = "commonapi.advanced.extended.AFExtendedOnce"
```

4.2.2.6 testAddressTwice

```
const std::string testAddressTwice = "commonapi.advanced.extended.AFExtendedTwice"
```

4.2.2.7 tasync

```
const int tasync = 10000
```

4.3 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/AFManaged.cpp File Reference

Macros

- #define `INTERFACE_DEVICE` "commonapi.advanced.managed.Device:v1_0"
- #define `INTERFACE_SPECIAL_DEVICE` "commonapi.advanced.managed.SpecialDevice:v1_0"
- #define `MIDDLE_INTERFACE` "commonapi.advanced.managed.HLevelMiddle:v1_0"
- #define `BOTTOM_INTERFACE` "commonapi.advanced.managed.HLevelBottom:v1_0"

Functions

- void [AFManaged_AddRemoveManagedInterfaceSingle](#) ()
- void [AFManaged_AddRemoveManagedInterfaceMultiple](#) ()
- void [AFManaged_AddRemoveMultipleManagedInterfacesSingle](#) ()
- void [AFManaged_AddRemoveMultipleManagedInterfacesMultiple](#) ()
- void [AFManaged_AddRemoveMultipleManagedInterfacesMultipleProxyNotActive](#) ()
- void [AFManaged_ProxyAddRemoveManagedInterfaceSingle](#) ()
- void [AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicit](#) ()
- void [AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicitAll](#) ()
- void [AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationImplicit](#) ()
- void [AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicit](#) ()
- void [AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicitAll](#) ()
- void [AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationImplicit](#) ()
- void [AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallSingleDeregistrationExplicit](#) ()
- void [AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallInProxyStatusEventSingleDeregistrationExplicit](#) ()
- void [AFManaged_DeleteManagerProxyInsideProxyStatusEventCallbackAndMethodCall](#) ()
- void [AFManaged_ProxyManagerTestPrimitiveMethods](#) ()
- void [AFManaged_ProxyManagerTestNonPrimitiveMethodsSync](#) ()
- void [AFManaged_ProxyManagerTestNonPrimitiveMethodsAsync](#) ()
- void [AFManaged_ProxyManagerTestGetInstanceAvailabilityStatusAsync](#) ()
- void [AFManaged_AddRemoveHierarchicalManagedInterface](#) ()
- void [AFManaged_GetAvailableInstancesWithoutSubscribe](#) ()
- void [AFManaged_CreateProxyToManagerInSameProcess](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string & [domain](#) = "local"

4.3.1 Macro Definition Documentation

4.3.1.1 INTERFACE_DEVICE

```
#define INTERFACE_DEVICE "commonapi.advanced.managed.Device:v1_0"
```

4.3.1.2 INTERFACE_SPECIAL_DEVICE

```
#define INTERFACE_SPECIAL_DEVICE "commonapi.advanced.managed.SpecialDevice:v1_0"
```

4.3.1.3 MIDDLE_INTERFACE

```
#define MIDDLE_INTERFACE "commonapi.advanced.managed.HLevelMiddle:v1_0"
```

4.3.1.4 BOTTOM_INTERFACE

```
#define BOTTOM_INTERFACE "commonapi.advanced.managed.HLevelBottom:v1_0"
```

4.3.2 Function Documentation

4.3.2.1 AFManaged_AddRemoveManagedInterfaceSingle()

```
void AFManaged_AddRemoveManagedInterfaceSingle ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Remove the managed interface from the manager
 - Check that the client is notified about the removed interface

4.3.2.2 AFManaged_AddRemoveManagedInterfaceMultiple()

```
void AFManaged_AddRemoveManagedInterfaceMultiple ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Add a second instance of the same managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Remove all the managed interfaces from the manager
 - Check that the client is notified about the removed interfaces

4.3.2.3 AFManaged_AddRemoveMultipleManagedInterfacesSingle()

```
void AFManaged_AddRemoveMultipleManagedInterfacesSingle ( )
```

- Test**
- Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Add a different managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Remove all the managed interfaces from the manager
 - Check that the client is notified about the removed interfaces

4.3.2.4 AFManaged_AddRemoveMultipleManagedInterfacesMultiple()

```
void AFManaged_AddRemoveMultipleManagedInterfacesMultiple ( )
```

- Test**
- Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Add a different managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Add a second instance of the same managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Remove all the managed interfaces from the manager
 - Check that the client is notified about the removed interfaces

4.3.2.5 AFManaged_AddRemoveMultipleManagedInterfacesMultipleProxyNotActive()

```
void AFManaged_AddRemoveMultipleManagedInterfacesMultipleProxyNotActive ( )
```

- Test**
-

4.3.2.6 AFManaged_ProxyAddRemoveManagedInterfaceSingle()

```
void AFManaged_ProxyAddRemoveManagedInterfaceSingle ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Remove the managed interface from the manager
 - Check that the client is notified about the removed interface

4.3.2.7 AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicit()

```
void AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicit ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Build a proxy through the manager to the managed device
 - Call a method on the managed device and check call status
 - Explicitly deregister managed interface through its instance name

4.3.2.8 AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicitAll()

```
void AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationExplicitAll ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Build a proxy through the manager to the managed device
 - Call a method on the managed device and check call status
 - Deregister all managed interfaces through manager's stub adapter

4.3.2.9 AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationImplicit()

```
void AFManaged_BuildProxyThroughManagerAndMethodCallSingleDeregistrationImplicit ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Build a proxy through the manager to the managed device
 - Call a method on the managed device and check call status
 - Don't deregister managed interfaces. This is done in dtor of manager's StubAdapterInternal when manager service is unregistered in TearDown() method.

4.3.2.10 AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicit()

```
void AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicit ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add managed interfaces to the manager
 - Check that the client is notified about the newly added interfaces
 - Build proxies through the manager to the managed interfaces
 - Call a method on the managed interfaces and check call status
 - Explicitly deregister managed interfaces through their instance name

4.3.2.11 AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicitAll()

```
void AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationExplicitAll ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add managed interfaces to the manager
 - Check that the client is notified about the newly added interfaces
 - Build proxies through the manager to the managed interfaces
 - Call a method on the managed interfaces and check call status
 - Deregister all managed interfaces through manager's stub adapter

4.3.2.12 AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationImplicit()

```
void AFManaged_BuildProxyThroughManagerAndMethodCallMultipleDeregistrationImplicit ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add managed interfaces to the manager
 - Check that the client is notified about the newly added interfaces
 - Build proxies through the manager to the managed interfaces
 - Call a method on the managed interfaces and check call status
 - Don't deregister managed interfaces. This is done in dtor of manager's StubAdapterInternal when manager service is unregistered in TearDown() method.

4.3.2.13 AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallSingleDeregistrationExplicit()

```
void AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallSingleDeregistrationExplicit ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Build a proxy through the manager to the managed device inside the availability event/callback
 - Call a method on the managed device and check call status inside the availability event/callback
 - Explicitly deregister managed interface through its instance name

4.3.2.14 AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallInProxyStatusEventSingleDeregistrationExplicit()

```
void AFManaged_BuildProxyThroughManagerInAvailabilityEventAndMethodCallInProxyStatusEventSingleDeregistrationExplicit ( )
```

- Test**
- Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Build a proxy through the manager to the managed device inside the availability event/callback
 - Subscribe to the proxy status event
 - Call a method on the managed device and check call status inside the proxy status event/callback (status == CommonAPI::AVAILABILITY_STATUS::AVAILABLE)
 - Remove and add the managed interface to the manager a few times
 - Explicitly deregister managed interface through its instance name

4.3.2.15 AFManaged_DeleteManagerProxyInsideProxyStatusEventCallbackAndMethodCall()

```
void AFManaged_DeleteManagerProxyInsideProxyStatusEventCallbackAndMethodCall ( )
```

- Test**
- Subscribe to the proxy status event of the manager
 - Subscribe on the events about availability status changes at the manager
 - Add the managed interfaces to the manager
 - Check that the client is notified about the newly added interfaces
 - Unregister manager service
 - Explicitly delete the proxy of the manager inside the proxy status event callback
 - Register manager service and build new proxy (Setup())
 - Subscribe on the events about availability status changes at the manager
 - Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Build a proxy through the manager to the managed device
 - Call a method on the managed device and check call status
 - TearDown()

4.3.2.16 AFManaged_ProxyManagerTestPrimitiveMethods()

```
void AFManaged_ProxyManagerTestPrimitiveMethods ( )
```

- Test**
- Test the getConnectionId, getDomain and getInterface methods available via the ProxyManager of the respective managed interfaces of the manager

4.3.2.17 AFManaged_ProxyManagerTestNonPrimitiveMethodsSync()

```
void AFManaged_ProxyManagerTestNonPrimitiveMethodsSync ( )
```

- Test**
- Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
 - Use the ProxyManager's checkInstanceAvailabilityStatus method to check that all returned instances by getAvailableInstances are available
 - Add a different managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
 - Use the ProxyManager's checkInstanceAvailabilityStatus method to check that all returned instances by getAvailableInstances are available
 - Add a second instance of the same managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
 - Use the ProxyManager's checkInstanceAvailabilityStatus method to check that all returned instances by getAvailableInstances are available
 - Remove all the managed interfaces from the manager
 - Check that the client is notified about the removed interfaces

4.3.2.18 AFManaged_ProxyManagerTestNonPrimitiveMethodsAsync()

```
void AFManaged_ProxyManagerTestNonPrimitiveMethodsAsync ( )
```

- Test**
- Add a managed interface to the manager
 - Check that the client is notified about the newly added interface
 - Use the ProxyManager's getAvailableInstancesAsync method to check that all registered instances are returned
 - Add a different managed interface to the manager

- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstancesAsync method to check that all registered instances are returned
- Add a second instance of the same managed interface to the manager
- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstancesAsync method to check that all registered instances are returned
- Remove all the managed interfaces from the manager
- Check that the client is notified about the removed interfaces

4.3.2.19 AFManaged_ProxyManagerTestGetInstanceAvailabilityStatusAsync()

```
void AFManaged_ProxyManagerTestGetInstanceAvailabilityStatusAsync ( )
```

Test

- Add a managed interface to the manager
- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
- Use the ProxyManager's checkInstanceAvailabilityStatusAsync method to check that all returned instances by getAvailableInstances are available
- Add a different managed interface to the manager
- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
- Use the ProxyManager's checkInstanceAvailabilityStatusAsync method to check that all returned instances by getAvailableInstances are available
- Add a second instance of the same managed interface to the manager
- Check that the client is notified about the newly added interface
- Use the ProxyManager's getAvailableInstances method to check that all registered instances are returned
- Use the ProxyManager's checkInstanceAvailabilityStatusAsync method to check that all returned instances by getAvailableInstances are available
- Remove all the managed interfaces from the manager
- Check that the client is notified about the removed interfaces

4.3.2.20 AFManaged_AddRemoveHierarchicalManagedInterface()

```
void AFManaged_AddRemoveHierarchicalManagedInterface ( )
```

4.3.2.21 AFManaged_GetAvailableInstancesWithoutSubscribe()

```
void AFManaged_GetAvailableInstancesWithoutSubscribe ( )
```

4.3.2.22 AFManaged_CreateProxyToManagerInSameProcess()

```
void AFManaged_CreateProxyToManagerInSameProcess ( )
```

Test Offer a interface manager and build two proxies to it. One proxy uses the same connection as the manager while the other uses a different connection. Check that both proxies get available and receive a available event

4.3.2.23 main()

```
int main (
    int argc,
    char ** argv )
```

4.3.3 Variable Documentation

4.3.3.1 domain

```
const std::string& domain = "local"
```

4.4 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/← AFPolymorph.cpp File Reference

Functions

- void [AFPolymorph_SetAndGetAttributeTypedef](#) ()
- void [AFPolymorph_SetAndGetAttributeEnum](#) ()
- void [AFPolymorph_SetAndGetAttributeUInt](#) ()
- void [AFPolymorph_SetAndGetAttributeString](#) ()
- void [AFPolymorph_SetAndGetAttributeStruct](#) ()
- void [AFPolymorph_MethodCall](#) ()
- void [AFPolymorph_Broadcast](#) ()
- void [AFPolymorph_SetAndGetAttributeDoublyUsedBaseStruct](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string `domain` = "local"
- const std::string `testAddress` = "commonapi.advanced.polymorph.TestInterface"
- const std::string `connectionId_client` = "client-sample"
- const std::string `connectionId_service` = "service-sample"
- const int `tasync` = 10000

4.4.1 Function Documentation

4.4.1.1 AFPolymorph_SetAndGetAttributeTypedef()

```
void AFPolymorph_SetAndGetAttributeTypedef ( )
```

- Test**
- Set and get a typedef-type attribute through a polymorphic structure
 - verify that the received data matches the transmitted data

4.4.1.2 AFPolymorph_SetAndGetAttributeEnum()

```
void AFPolymorph_SetAndGetAttributeEnum ( )
```

- Test**
- Set and get a enum-type attribute through a polymorphic structure
 - verify that the received data matches the transmitted data

4.4.1.3 AFPolymorph_SetAndGetAttributeUInt()

```
void AFPolymorph_SetAndGetAttributeUInt ( )
```

- Test**
- Set and get a uint-type attribute through a polymorphic structure
 - verify that the received data matches the transmitted data

4.4.1.4 AFPolymorph_SetAndGetAttributeString()

```
void AFPolymorph_SetAndGetAttributeString ( )
```

- Test**
- Set and get a string-type attribute through a polymorphic structure
 - verify that the received data matches the transmitted data

4.4.1.5 AFPolymorph_SetAndGetAttributeStruct()

```
void AFPolymorph_SetAndGetAttributeStruct ( )
```

- Test**
- Set and get a struct-type attribute through a polymorphic structure
 - verify that the received data matches the transmitted data

4.4.1.6 AFPolymorph_MethodCall()

```
void AFPolymorph_MethodCall ( )
```

- Test**
- Call a method whose input and output parameters are polymorphic structures
 - verify that the received data matches the transmitted data

4.4.1.7 AFPolymorph_Broadcast()

```
void AFPolymorph_Broadcast ( )
```

- Test**
- Call a method with a special value that tells the stub to send a broadcast signal
 - verify that the received data matches the transmitted data

4.4.1.8 AFPolymorph_SetAndGetAttributeDoublyUsedBaseStruct()

```
void AFPolymorph_SetAndGetAttributeDoublyUsedBaseStruct ( )
```

- Test**
- Set and get an attribute through a polymorphic structure whose Base is also used by another identical structure.
 - verify that the received data matches the transmitted data

4.4.1.9 main()

```
int main (
    int argc,
    char ** argv )
```

4.4.2 Variable Documentation

4.4.2.1 domain

```
const std::string domain = "local"
```

4.4.2.2 testAddress

```
const std::string testAddress = "commonapi.advanced.polymorph.TestInterface"
```

4.4.2.3 connectionId_client

```
const std::string connectionId_client = "client-sample"
```

4.4.2.4 connectionId_service

```
const std::string connectionId_service = "service-sample"
```

4.4.2.5 tasync

```
const int tasync = 10000
```

4.5 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/↵ AFSelective.cpp File Reference

Functions

- void [AFSelective_SelectiveBroadcastRejected](#) ()
- void [AFSelective_SelectiveBroadcast](#) ()
- void [AFSelective_SelectiveMultiBroadcast](#) ()
- void [AFSelective_ProxyBuildAndDestroy](#) ()
- void [AFSelective_SelectiveRejectedMultiBroadcast](#) ()
- void [AFSelective_Multiple_Subscriptions_SameConnection_CallErrorHandler](#) ()
- void [AFSelective_Fire_Selective_Within_Subscription_Changed_Hook](#) ()
- void [AFSelective_Two_proxies_subscribe_delete_one_proxy](#) ()
- void [AFSelective_Two_proxies_subscribe_delete_one_proxy_error_listener_test](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [serviceId](#) = "service-sample"
- const std::string [clientId](#) = "client-sample"
- const std::string [otherclientId](#) = "other-client-sample"
- const std::string [domain](#) = "local"
- const std::string [testAddress](#) = "commonapi.advanced.bselective.TestInterface"
- const int [tasync](#) = 10000

4.5.1 Function Documentation

4.5.1.1 AFSelective_SelectiveBroadcastRejected()

```
void AFSelective_SelectiveBroadcastRejected ( )
```

Test Test selective broadcasts.

- inform stub to stop accepting subscriptions
- try to subscribe to the selective broadcast
- check that an error was received
- inform stub to send a broadcast
- check that nothing was received in a reasonable time

4.5.1.2 AFSelective_SelectiveBroadcast()

```
void AFSelective_SelectiveBroadcast ( )
```

Test Test selective broadcasts.

- inform stub to start accepting subscriptions
- subscribe to the selective broadcast
- check that no error was received (in a reasonable time)
- inform stub to send a broadcast
- check that a correct value is received

4.5.1.3 AFSelective_SelectiveMultiBroadcast()

```
void AFSelective_SelectiveMultiBroadcast ( )
```

Test Test multiple selective broadcasts.

- inform stub to start accepting subscriptions
- subscribe to the selective broadcast
- check that no error was received (in a reasonable time)
- inform stub to send a broadcast
- check that a correct value is received

4.5.1.4 AFSelective_ProxyBuildAndDestroy()

```
void AFSelective_ProxyBuildAndDestroy ( )
```

Test Test multiple selective broadcasts, with rejection.

- subscribe to stub three times: once from proxy2, once from proxy1 (accepted) once from proxy2 (rejected)
- This should result with two subscription callbacks being called from broadcast.

Test Test Destruction of Proxies but service stay online There were an issue when a proxy which has never subscribed gets destructed with SomeIP binding (GLIPCI-1081). Therefore i added this test case.

4.5.1.5 AFSelective_SelectiveRejectedMultiBroadcast()

```
void AFSelective_SelectiveRejectedMultiBroadcast ( )
```

4.5.1.6 AFSelective_Multiple_Subscriptions_SameConnection_CallErrorHandler()

```
void AFSelective_Multiple_Subscriptions_SameConnection_CallErrorHandler ( )
```

4.5.1.7 AFSelective_Fire_Selective_Within_Subscription_Changed_Hook()

```
void AFSelective_Fire_Selective_Within_Subscription_Changed_Hook ( )
```

4.5.1.8 AFSelective_Two_proxies_subscribe_delete_one_proxy()

```
void AFSelective_Two_proxies_subscribe_delete_one_proxy ( )
```

4.5.1.9 AFSelective_Two_proxies_subscribe_delete_one_proxy_error_listener_test()

```
void AFSelective_Two_proxies_subscribe_delete_one_proxy_error_listener_test ( )
```

4.5.1.10 main()

```
int main (
    int argc,
    char ** argv )
```

4.5.2 Variable Documentation

4.5.2.1 serviceId

```
const std::string serviceId = "service-sample"
```

4.5.2.2 clientId

```
const std::string clientId = "client-sample"
```


4.5.2.3 otherclientId

```
const std::string otherclientId = "other-client-sample"
```

4.5.2.4 domain

```
const std::string domain = "local"
```

4.5.2.5 testAddress

```
const std::string testAddress = "commonapi.advanced.bselective.TestInterface"
```

4.5.2.6 tasync

```
const int tasync = 10000
```

4.6 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/CMAAttributes.cpp File Reference

Functions

- void [CMAAttributes_AttributeGetSynchronous](#) ()
- void [CMAAttributes_AttributeGetAsynchronous](#) ()
- void [CMAAttributes_AttributeSetSynchronous](#) ()
- void [CMAAttributes_AttributeSetAsynchronous](#) ()
- void [CMAAttributes_AttributeSubscription](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [serviceld](#) = "service-sample"
- const std::string [clientId](#) = "client-sample"
- const std::string [domain](#) = "local"
- const std::string [testAddress](#) = "commonapi.communication.TestInterface"
- const int [tasync](#) = 10000

4.6.1 Function Documentation

4.6.1.1 CMAAttributes_AttributeGetSynchronous()

```
void CMAAttributes_AttributeGetSynchronous ( )
```

Test Test synchronous getValue API function for attributes with combinations of additional properties readonly and noSubscriptions (testAttribute, testA readonly, testB noSubscriptions, testC readonly noSubscriptions).

- Set attribute to certain value on stub side.
- Call getValue.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if value of is equal to expected value.

4.6.1.2 CMAAttributes_AttributeGetAsynchronous()

```
void CMAAttributes_AttributeGetAsynchronous ( )
```

Test Test asynchronous getValue API function for attributes with combinations of additional properties readonly and noSubscriptions (testAttribute, testA readonly, testB noSubscriptions, testC readonly noSubscriptions).

- Set attribute to certain value on stub side.
- Call getValue.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if value of is equal to expected value.

4.6.1.3 CMAAttributes_AttributeSetSynchronous()

```
void CMAAttributes_AttributeSetSynchronous ( )
```

Test Test synchronous setValue API function for attributes with combinations of additional properties readonly and noSubscriptions (testAttribute, testB noSubscriptions)

- Set attribute to certain value on proxy side.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if returned value of setValue is equal to expected value.

4.6.1.4 CMAAttributes_AttributeSetAsynchronous()

```
void CMAAttributes_AttributeSetAsynchronous ( )
```

Test Test asynchronous setValue API function for attributes with combinations of additional properties readonly and noSubscriptions (testAttribute, testB noSubscriptions).

- Set attribute to certain value on proxy side.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if returned value of setValue is equal to expected value.

4.6.1.5 CMAAttributes_AttributeSubscription()

```
void CMAAttributes_AttributeSubscription ( )
```

Test Test subscription API function for attributes

- Subscribe on testAttribute.
- Set attribute to certain value on stub side.
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and returned value in callback function.
- Checks if returned value of setValue is equal to expected value.
- Set attribute to certain value with synchronous call from proxy.
- Check again.

4.6.1.6 main()

```
int main (
    int argc,
    char ** argv )
```

4.6.2 Variable Documentation

4.6.2.1 serviceId

```
const std::string serviceId = "service-sample"
```

4.6.2.2 clientId

```
const std::string clientId = "client-sample"
```

4.6.2.3 domain

```
const std::string domain = "local"
```

4.6.2.4 testAddress

```
const std::string testAddress = "commonapi.communication.TestInterface"
```

4.6.2.5 tasync

```
const int tasync = 10000
```

4.7 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/CMAttributeSubscription.cpp File Reference

Typedefs

- typedef std::shared_ptr< v1_0::commonapi::communication::TestInterfaceProxy<> > [ProxyPtr](#)

Functions

- void [testSubscription](#) ([ProxyPtr](#) pp, std::shared_ptr< std::promise< bool > > subscribedToProxyStatus, std::shared_ptr< std::promise< bool > > subscribedToAttributePromise)
- void [CMAttributeSubscription_SubscriptionStandard](#) ()
- void [CMAttributeSubscription_SubscriptionOnAvailable](#) ()
- void [CMAttributeSubscription_SubscriptionMultithreading](#) ()
- void [CMAttributeSubscription_SubscriptionUnsubscribeFromCallback](#) ()
- void [CMAttributeSubscription_SubscribeAndUnsubscribeTwoCallbacksCoexistent](#) ()
- void [CMAttributeSubscription_SubscribeAndUnsubscribeSequentially](#) ()
- void [CMAttributeSubscription_DISABLED_SubscribeAndUnsubscribeImplicitWithCreatingNewProxyWithReassigning](#) ()
- void [CMAttributeSubscription_SubscribeAndUnsubscribeUnsubscribe](#) ()
- void [CMAttributeSubscription_SubscribeServiceNotAvailable](#) ()
- void [CMAttributeSubscription_SubscribeUnregisterSetValueRegisterService](#) ()
- void [CMAttributeSubscription_SubscribeUnregisterNoValueSetRegisterService](#) ()
- void [CMAttributeSubscription_SubscribeSecondProxyLater](#) ()
- void [CMAttributeSubscription_SubscribeThreeCallbacksServiceNotAvailable](#) ()
- void [CMAttributeSubscription_SubscribeThreeCallbacksServiceAvailable](#) ()
- void [CMAttributeSubscription_SubscribeAndUnsubscribeAndReSubscribe](#) ()
- void [CMAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribe](#) ()
- void [CMAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribeSameEventgroup](#) ()
- void [CMAttributeSubscription_SubscribeMultipleProxysUnsubscribeOneResubscribeSameEventgroup](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string `daemonId` = "service-sample"
- const std::string `clientId` = "client-sample"
- const std::string `serviceId` = "test-service"
- const std::string `domain` = "local"
- const std::string `testAddress` = "commonapi.communication.TestInterface"
- const std::string `daemonAddress` = "commonapi.communication.Daemon"
- const unsigned int `wt` = 10000
- const unsigned int `wf` = 1
- std::mutex `mut`
- std::deque< uint32_t > `data_queue`
- std::condition_variable `data_cond`

4.7.1 Typedef Documentation

4.7.1.1 ProxyPtr

```
typedef std::shared_ptr<v1_0::commonapi::communication::TestInterfaceProxy<> > ProxyPtr
```

4.7.2 Function Documentation

4.7.2.1 testSubscription()

```
void testSubscription (
    ProxyPtr pp,
    std::shared_ptr< std::promise< bool > > subscribedToProxyStatusPromise,
    std::shared_ptr< std::promise< bool > > subscribedToAttributePromise )
```

4.7.2.2 CMAAttributeSubscription_SubscriptionStandard()

```
void CMAAttributeSubscription_SubscriptionStandard ( )
```

Test Subscription standard test.

- Register service and check if proxy is available.
- Proxy subscribes for TestAttribute (uint8_t).
- Change attribute in service several times by set method.
- Callback function in proxy writes the received values in a queue.
- Check if values in the queue are the same as the values that were set in the service.
- Unregister test service.

4.7.2.3 CMAAttributeSubscription_SubscriptionOnAvailable()

```
void CMAAttributeSubscription_SubscriptionOnAvailable ( )
```

Test Subscription test with subscription on available-event.

- Subscribe for available-event.
- Available-callback subscribes for TestPredefinedTypeAttribute if service is available for proxy and unsubscribes if service is not available for proxy.
- Change attribute in service by set method; the new attribute value should be received by the proxy because the service is not registered.
- Register service and change value again; the value should now be received.
- Unregister and change value again.

4.7.2.4 CMAAttributeSubscription_SubscriptionMultithreading()

```
void CMAAttributeSubscription_SubscriptionMultithreading ( )
```

Test Subscription test with several threads.

- Start several threads.
- The threads subscribe for the availability status.
- The available-callback subscribes for TestAttribute if service is available for proxy and unsubscribes if service is not available for proxy.
- Change attribute in service by set method; the new attribute value should be received by all the threads.
- The new value is written into a queue.
- Check if the values of each thread are written into the queue.

4.7.2.5 CMAAttributeSubscription_SubscriptionUnsubscribeFromCallback()

```
void CMAAttributeSubscription_SubscriptionUnsubscribeFromCallback ( )
```

Test Subscription test : unsubscribe from the subscription callback.

- Register service and check if proxy is available.
- Proxy subscribes for TestAttribute (uint8_t).
- Change attribute in service by set method.
- Check if callback function in proxy received the right value.
- Change value to the magic value 99: this triggers the callback to unsubscribe.
- Change value again; the callback should now be called anymore.
- Unregister the test service.

4.7.2.6 CMAAttributeSubscription_SubscribeAndUnsubscribeTwoCallbacksCoexistent()

```
void CMAAttributeSubscription_SubscribeAndUnsubscribeTwoCallbacksCoexistent ( )
```

Test Test of subscribe and unsubscribe with two coexistent callbacks

- subscribe both callbacks
- change value
- check that both callbacks were executed by changing the value
- unsubscribe both callbacks
- change value
- check that both callbacks were not executed by changing the value

4.7.2.7 CMAAttributeSubscription_SubscribeAndUnsubscribeSequentially()

```
void CMAAttributeSubscription_SubscribeAndUnsubscribeSequentially ( )
```

Test Test of subscribing and immediately unsubscribing a callback

- subscribe first callback
- subscribe second callback
- unsubscribe second callback
- change value
- check that only first callback was executed

Test Test of subscribing and unsubscribing sequentially

- subscribe first callback
- subscribe second callback
- change value
- check that both callbacks were executed by changing the value
- unsubscribe first callback
- change value
- check that only second callback was executed
- unsubscribe second callback
- change value
- check that both callbacks were not executed by changing the value

4.7.2.8 CMAAttributeSubscription_DISABLED_SubscribeAndUnsubscribeImplicitWithCreatingNewProxyWithReassigning()

```
void CMAAttributeSubscription_DISABLED_SubscribeAndUnsubscribeImplicitWithCreatingNewProxy↔  
WithReassigning ( )
```

Test Test of subscribing and unsubscribing implicit with creating a new proxy with reassigning

- subscribe first callback
- subscribe second callback
- change value
- check that both callbacks were executed by changing the value
- create new proxy with reassigning. So the connection won't be destroyed and the callbacks are unsubscribed implicitly.
- subscribe second callback
- change value
- check that only second callback was executed
- unsubscribe second callback
- change value
- check that both callbacks were not executed by changing the value

4.7.2.9 CMAAttributeSubscription_SubscribeAndUnsubscribeUnsubscribe()

```
void CMAAttributeSubscription_SubscribeAndUnsubscribeUnsubscribe ( )
```

Test Test of behaviour in case unsubscribe is called two times

- set default value
- register service
- subscribe for the attribute
- current value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- proxy unsubscribes for the attribute
- value of attribute is changed
- changed value must not be communicated to the proxy
- proxy unsubscribes again for the attribute
- value of attribute is changed
- changed value must not be communicated to the proxy
- unregister service

4.7.2.10 CMAAttributeSubscription_SubscribeServiceNotAvailable()

```
void CMAAttributeSubscription_SubscribeServiceNotAvailable ( )
```

Test Test of subscribing in case that service is not available

- set default value
- subscribe for the attribute
- no value is communicated to the proxy
- register service
- current value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- unregister service

4.7.2.11 CMAAttributeSubscription_SubscribeUnregisterSetValueRegisterService()

```
void CMAAttributeSubscription_SubscribeUnregisterSetValueRegisterService ( )
```

Test Test of unregister a service in case a proxy is subscribed for an attribute of this service. During the unregistered time of the service the value of the attribute is changed.

- register service
- proxy subscribes for an attribute of the service
- value of attribute is set
- changed value must be communicated to the proxy
- unregister service
- value of attribute is changed
- changed value must not be communicated to the proxy
- register service
- current attribute value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- unregister service

4.7.2.12 CMAAttributeSubscription_SubscribeUnregisterNoValueSetRegisterService()

```
void CMAAttributeSubscription_SubscribeUnregisterNoValueSetRegisterService ( )
```

Test Test of unregister a service in case a proxy is subscribed for an attribute of this service. During the unregistered time of the service the value of the attribute is not changed.

- register service
- proxy subscribes for an attribute of the service
- value of attribute is set
- changed value must be communicated to the proxy
- unregister service
- register service
- current attribute value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- unregister service

4.7.2.13 CMAAttributeSubscription_SubscribeSecondProxyLater()

```
void CMAAttributeSubscription_SubscribeSecondProxyLater ( )
```

Test Test of subscribing a second proxy a little bit later

- proxy subscribes for an attribute of the service
- register service
- initial value must be communicated to the proxy
- create a second proxy
- second proxy subscribes for the same attribute of the service
- current attribute value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to both proxies
- unregister service

4.7.2.14 CMAAttributeSubscription_SubscribeThreeCallbacksServiceNotAvailable()

```
void CMAAttributeSubscription_SubscribeThreeCallbacksServiceNotAvailable ( )
```

Test Test of subscribing three callbacks before registering the service

- proxy subscribes three callbacks for an attribute of the service
- register service
- initial value must be communicated to every callback

4.7.2.15 CMAAttributeSubscription_SubscribeThreeCallbacksServiceAvailable()

```
void CMAAttributeSubscription_SubscribeThreeCallbacksServiceAvailable ( )
```

Test Test of subscribing three callbacks after registering the service

- register service
- proxy subscribes three callbacks for an attribute of the service
- initial value must be communicated to every callback

4.7.2.16 CMAAttributeSubscription_SubscribeAndUnsubscribeAndReSubscribe()

```
void CMAAttributeSubscription_SubscribeAndUnsubscribeAndReSubscribe ( )
```

Test Test of behaviour in case subscribe, unsubscribe and resubscribe is done

- set default value
- register service
- subscribe for the attribute
- current value must be communicated to the proxy
- value of attribute is changed
- changed value must be communicated to the proxy
- proxy unsubscribes for the attribute
- value of attribute is not changed
- value received by proxy is reset to 0
- proxy resubscribes for the attribute
- current value must be communicated to the proxy
- value received must be equal to value received before last unsubscribe call
- unregister service

4.7.2.17 CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribe()

```
void CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribe ( )
```

Test Test of behaviour in case subscribe and unsubscribe is done for multiple proxys on the same attribute and afterwards all proxys resubscribe

- set default value
- register service
- subscribe for the attribute with proxyA
- subscribe for the attribute with proxyB
- current value must be communicated to the proxyA

- current value must be communicated to the proxyB
- value of attribute is changed
- changed value must be communicated to the proxyA
- changed value must be communicated to the proxyB
- proxyA and proxy B unsubscribe for the attribute
- value of attribute is not changed
- value received is reset to 0
- proxyA and proxyB resubscribe for the attribute
- current value must be communicated to the proxy as initial value
- value received must be equal to value received before last unsubscribe call
- unregister service

4.7.2.18 CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribeSameEventgroup()

```
void CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllResubscribeSameEventgroup (
)
```

Test Test of behaviour in case subscribe and unsubscribe is done for multiple proxys on attributes in the same eventgroup and afterwards all proxys resubscribe

- set default value
- register service
- subscribe for the attribute 1 with proxyA
- subscribe for the attribute 2 with proxyB
- current value must be communicated to the proxyA
- current value must be communicated to the proxyB
- value of attribute is changed
- changed value must be communicated to the proxyA
- changed value must be communicated to the proxyB
- proxyA and proxy B unsubscribe for the attributes
- value of attributes is not changed
- value received is reset to 0
- proxyA and proxyB resubscribe for the attribute 1 and 2
- current value must be communicated to the proxys as initial value
- value received must be equal to value received before last unsubscribe call
- unregister service

4.7.2.19 CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeOneResubscribeSameEventgroup()

```
void CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeOneResubscribeSameEventgroup (
)
```

Test Test of behaviour in case two proxys A and B subscribe to events that are in one eventgroup, proxyB unsubscribes, and proxy A is still expected to receive changed values.

- set default value
- register service
- subscribe for the attribute with proxyA
- subscribe for the attribute with proxyB
- current value must be communicated to the proxyA
- current value must be communicated to the proxyB
- value of attribute is changed
- changed value must be communicated to the proxyA
- changed value must be communicated to the proxyB
- proxyB unsubscribes for the attribute
- value of attribute is changed
- value received must be equal to changed value for proxy A
- unregister service

4.7.2.20 main()

```
int main (
    int argc,
    char ** argv )
```

4.7.3 Variable Documentation

4.7.3.1 daemonId

```
const std::string daemonId = "service-sample"
```

4.7.3.2 clientId

```
const std::string clientId = "client-sample"
```

4.7.3.3 serviceId

```
const std::string serviceId = "test-service"
```

4.7.3.4 domain

```
const std::string domain = "local"
```

4.7.3.5 testAddress

```
const std::string testAddress = "commonapi.communication.TestInterface"
```

4.7.3.6 daemonAddress

```
const std::string daemonAddress = "commonapi.communication.Daemon"
```

4.7.3.7 wt

```
const unsigned int wt = 10000
```

4.7.3.8 wf

```
const unsigned int wf = 1
```

4.7.3.9 mut

```
std::mutex mut
```

4.7.3.10 data_queue

```
std::deque<uint32_t> data_queue
```

4.7.3.11 data_cond

`std::condition_variable data_cond`

4.8 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/CMBlockingCalls.cpp File Reference

Functions

- void `CMBlockingCalls_BlockInStubMethod` ()
- void `CMBlockingCalls_BlockInProxyCallback` ()
- void `CMBlockingCalls_BlockInAvailabilityHandler` ()
- void `CMBlockingCalls_BlockInAvailabilityHandlerAndReceiveCallbacks` ()
- void `CMBlockingCalls_NestedBlockInStubMethods` ()
- int `main` (int argc, char **argv)

Variables

- const std::string `serviceId` = "service-sample"
- const std::string `clientId` = "client-sample"
- const std::string `clientId2` = "other-client-sample"
- const std::string `domain` = "local"
- const std::string `testAddress` = "commonapi.communication.TestInterface"
- const std::string `testAddress2` = "commonapi.communication.TestInterface2"
- const int `tasync` = 20000
- const int `timeout` = 300
- const int `maxTimeoutCalls` = 10
- const unsigned int `wf` = 1

4.8.1 Function Documentation

4.8.1.1 CMBlockingCalls_BlockInStubMethod()

`void CMBlockingCalls_BlockInStubMethod ()`

Test Call test method which generates blocking calls on stub side and check if answers are received.

4.8.1.2 CMBlockingCalls_BlockInProxyCallback()

```
void CMBlockingCalls_BlockInProxyCallback ( )
```

Test Call test method and block in registered callback when processing responses. Check that all responses are delivered.

4.8.1.3 CMBlockingCalls_BlockInAvailabilityHandler()

```
void CMBlockingCalls_BlockInAvailabilityHandler ( )
```

Test Register availability handler which blocks and (de)register the corresponding service multiple times. After the service stays available do a method call and check that the answer is received

4.8.1.4 CMBlockingCalls_BlockInAvailabilityHandlerAndReceiveCallbacks()

```
void CMBlockingCalls_BlockInAvailabilityHandlerAndReceiveCallbacks ( )
```

Test Create proxy to service and wait until it is reported as available via a registered availability handler. As soon as it is available start sending requests to the service and wait for its replies. Check that the replies for this requests are dispatched even if the availability handler for this service is blocked. This is tested through blocking in the availability handler after the main thread was notified about the services' availability

4.8.1.5 CMBlockingCalls_NestedBlockInStubMethods()

```
void CMBlockingCalls_NestedBlockInStubMethods ( )
```

Test Call test method which generates blocking calls on stub. Ensure working dispatching even if main dispatch thread still blocked after a dispatch thread was spawned and joined again because another dispatch thread returned from the usercode in the meanwhile.

4.8.1.6 main()

```
int main (
    int argc,
    char ** argv )
```


4.8.2 Variable Documentation

4.8.2.1 serviceId

```
const std::string serviceId = "service-sample"
```

4.8.2.2 clientId

```
const std::string clientId = "client-sample"
```

4.8.2.3 clientId2

```
const std::string clientId2 = "other-client-sample"
```

4.8.2.4 domain

```
const std::string domain = "local"
```

4.8.2.5 testAddress

```
const std::string testAddress = "commonapi.communication.TestInterface"
```

4.8.2.6 testAddress2

```
const std::string testAddress2 = "commonapi.communication.TestInterface2"
```

4.8.2.7 tasync

```
const int tasync = 20000
```

4.8.2.8 timeout

```
const int timeout = 300
```

4.8.2.9 maxTimeoutCalls

```
const int maxTimeoutCalls = 10
```

4.8.2.10 wf

```
const unsigned int wf = 1
```

4.9 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/↵ CMBroadcasts.cpp File Reference

Functions

- void [CMBroadcasts_NormalBroadcast](#) ()
- void [CMBroadcasts_SelectiveBroadcastRejected](#) ()
- void [CMBroadcasts_SelectiveBroadcast](#) ()
- void [CMBroadcasts_BroadcastStubGoesOfflineOnlineAgain](#) ()
- void [CMBroadcasts_SelectiveBroadcastStubGoesOfflineOnlineAgain](#) ()
- void [CMBroadcasts_NormalBroadcast_Two_proxies_subscribe_and_one_reset](#) ()
- void [CMBroadcasts_Two_proxies_subscribe_delete_one_proxy_status_listener_test](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [serviceld](#) = "service-sample"
- const std::string [clientId](#) = "client-sample"
- const std::string [otherclientId](#) = "other-client-sample"
- const std::string [domain](#) = "local"
- const std::string [testAddress](#) = "commonapi.communication.TestInterface"
- const int [tasync](#) = 10000
- const unsigned int [wf](#) = 1

4.9.1 Function Documentation

4.9.1.1 CMBroadcasts_NormalBroadcast()

```
void CMBroadcasts_NormalBroadcast ( )
```

Test Test broadcasts. Subscribe to a broadcast, and see that the value is correctly received.

4.9.1.2 CMBroadcasts_SelectiveBroadcastRejected()

```
void CMBroadcasts_SelectiveBroadcastRejected ( )
```

Test Test selective broadcasts.

- inform stub to stop accepting subscriptions
- try to subscribe to the selective broadcast
- check that an error was received
- inform stub to send a broadcast
- check that nothing was received in a reasonable time

4.9.1.3 CMBroadcasts_SelectiveBroadcast()

```
void CMBroadcasts_SelectiveBroadcast ( )
```

Test Test selective broadcasts.

- inform stub to start accepting subscriptions
- subscribe to the selective broadcast
- check that no error was received (in a reasonable time)
- inform stub to send a broadcast
- check that a correct value is received

4.9.1.4 CMBroadcasts_BroadcastStubGoesOfflineOnlineAgain()

```
void CMBroadcasts_BroadcastStubGoesOfflineOnlineAgain ( )
```

Test Test BroadcastStubGoesOfflineOnlineAgain.

- service offline
- subscribe to broadcast
- service online
- fire broadcast -> proxy should receive
- service offline
- service online
- fire again -> proxy should receive again

4.9.1.5 CMBroadcasts_SelectiveBroadcastStubGoesOfflineOnlineAgain()

```
void CMBroadcasts_SelectiveBroadcastStubGoesOfflineOnlineAgain ( )
```

Test Test SelectiveBroadcastStubGoesOfflineOnlineAgain.

- service offline
- subscribe to selective broadcast
- service online
- fire selective broadcast -> proxy should receive
- service offline
- service online
- fire again -> proxy should receive again

4.9.1.6 CMBroadcasts_NormalBroadcast_Two_proxies_subscribe_and_one_reset()

```
void CMBroadcasts_NormalBroadcast_Two_proxies_subscribe_and_one_reset ( )
```

4.9.1.7 CMBroadcasts_Two_proxies_subscribe_delete_one_proxy_status_listener_test()

```
void CMBroadcasts_Two_proxies_subscribe_delete_one_proxy_status_listener_test ( )
```

4.9.1.8 main()

```
int main (
    int argc,
    char ** argv )
```

4.9.2 Variable Documentation

4.9.2.1 serviceId

```
const std::string serviceId = "service-sample"
```

4.9.2.2 clientId

```
const std::string clientId = "client-sample"
```

4.9.2.3 otherclientId

```
const std::string otherclientId = "other-client-sample"
```

4.9.2.4 domain

```
const std::string domain = "local"
```

4.9.2.5 testAddress

```
const std::string testAddress = "commonapi.communication.TestInterface"
```

4.9.2.6 tasync

```
const int tasync = 10000
```

4.9.2.7 wf

```
const unsigned int wf = 1
```

4.10 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/CMMMethodCalls.cpp File Reference

Functions

- void [CMMMethodCalls_SynchronousMethodCall](#) ()
- void [CMMMethodCalls_FireAndForget](#) ()
- void [CMMMethodCalls_AsynchronousMethodCall](#) ()
- void [CMMMethodCalls_NestedSynchronousMethodCall](#) ()
- void [CMMMethodCalls_NestedAsynchronousMethodCall](#) ()
- void [CMMMethodCalls_NestedAsynchronousMethodCallsTimedOut](#) ()
- void [CMMMethodCalls_AsynchronousMethodCallProxyNotAvailable](#) ()
- void [CMMMethodCalls_NestedAsynchronousMethodCallProxyNotAvailable](#) ()
- void [CMMMethodCalls_AsynchronousMethodCallProxyBecomesAvailable](#) ()
- void [CMMMethodCalls_NestedAsynchronousMethodCallProxyBecomesAvailable](#) ()
- void [CMMMethodCalls_AsynchronousMethodCallsProxyBecomesAvailable](#) ()
- void [CMMMethodCalls_AsynchronousMethodCallProxyNotAvailableDeleteProxy](#) ()
- void [CMMMethodCalls_AsynchronousMethodCallsReceiveNotAvailable](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string `serviceld` = "service-sample"
- const std::string `clientId` = "client-sample"
- const std::string `domain` = "local"
- const std::string `testAddress` = "commonapi.communication.TestInterface"
- const std::string `testAddress2` = "commonapi.communication.TestInterface2"
- const int `tasync` = 20000
- const int `timeout` = 300
- const int `maxTimeoutCalls` = 10
- const unsigned int `wf` = 1

4.10.1 Function Documentation

4.10.1.1 CMMethodCalls_SynchronousMethodCall()

```
void CMMethodCalls_SynchronousMethodCall ( )
```

Test Call test method synchronous and check call status.

- Test stub sets in-value of test method equal out-value of test method.
- Make synchronous call of test method.
- Check if returned call status is CommonAPI::CallStatus::SUCCESS.
- Check if out value of test method is equal to in value.

4.10.1.2 CMMethodCalls_FireAndForget()

```
void CMMethodCalls_FireAndForget ( )
```

Test Call fire and forget method and check via broadcast that value was received.

- Subscribe to broadcast
- Check that broadcast subscription succeeded
- Make fire and forget method call
- Check via broadcast that value was correctly received (Stub fires broadcast when value was received).

4.10.1.3 CMMethodCalls_AsynchronousMethodCall()

```
void CMMethodCalls_AsynchronousMethodCall ( )
```

Test Call test method asynchronous and check call status.

- Test stub sets in-value of test method.
- Make asynchronous call of test method.
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored value in callback function.

4.10.1.4 CMMethodCalls_NestedSynchronousMethodCall()

```
void CMMethodCalls_NestedSynchronousMethodCall ( )
```

Test Call test method asynchronous and call test method synchronous in callback (nested).

- Test stub sets in-values of test methods.
- Make asynchronous call of test method.
- Make asynchronous call of test method in callback (nested).
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored values in callback functions.

4.10.1.5 CMMethodCalls_NestedAsynchronousMethodCall()

```
void CMMethodCalls_NestedAsynchronousMethodCall ( )
```

Test Call test method asynchronous and call test method asynchronous in callback (nested).

- Test stub sets in-values of test methods.
- Make asynchronous call of test method.
- Make asynchronous call of test method in callback (nested).
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored values in callback functions.

4.10.1.6 CMMethodCalls_NestedAsynchronousMethodCallsTimedOut()

```
void CMMethodCalls_NestedAsynchronousMethodCallsTimedOut ( )
```

Test Call test method timeout asynchronous and call test method timeout asynchronous in callback (nested).

- Register second service with other instance
- Create second proxy to second service
- Make asynchronous call of test method timeout (first proxy)
- Make asynchronous call of test method timeout (second proxy)
- Check in callbacks if timeout occurred (CommonAPI::CallStatus::REMOTE_ERROR)
- Make asynchronous calls of test method timeout in callbacks as long as timeoutCalls_ < maxTimeoutCalls_ (nested).
- Check if the same amount of timeouts occurred as async calls were done

4.10.1.7 CMMethodCalls_AsynchronousMethodCallProxyNotAvailable()

```
void CMMethodCalls_AsynchronousMethodCallProxyNotAvailable ( )
```

Test Call test method asynchronous when proxy is not available.

- Unregister service.
- Wait that proxy is not available.
- Test stub sets in-value of test method.
- Set timeout of asynchronous call.
- Make asynchronous call of test method.
- Do checks of call status (CommonAPI::CallStatus::NOT_AVAILABLE) and that timeout occurred.

4.10.1.8 CMMethodCalls_NestedAsynchronousMethodCallProxyNotAvailable()

```
void CMMethodCalls_NestedAsynchronousMethodCallProxyNotAvailable ( )
```

Test Call test method asynchronous and call test method asynchronous in callback (nested) when proxy is not available.

- Unregister service.
- Wait that proxy is not available.
- Test stub sets in-value of test methods.
- Set timeout of asynchronous calls.
- Make asynchronous call of test method.
- Make asynchronous call of test method in callback (nested).
- Do checks of call status (CommonAPI::CallStatus::NOT_AVAILABLE) and that timeouts occurred.

4.10.1.9 CMMethodCalls_AsynchronousMethodCallProxyBecomesAvailable()

```
void CMMethodCalls_AsynchronousMethodCallProxyBecomesAvailable ( )
```

Test Call test method asynchronous when proxy is not available. Proxy becomes available during call.

- Unregister service
- Wait that proxy is not available.
- Test stub sets in-value of test method.
- Set timeout of asynchronous call.
- Make asynchronous call of test method.
- Proxy becomes available during call.
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored value in callback function.

4.10.1.10 CMMMethodCalls_NestedAsynchronousMethodCallProxyBecomesAvailable()

```
void CMMMethodCalls_NestedAsynchronousMethodCallProxyBecomesAvailable ( )
```

Test Call test method asynchronous and call test method asynchronous in callback (nested) when proxy is not available. Proxy becomes available during call.

- Unregiser service
- Wait that proxy is not available.
- Test stub sets in-values of test methods.
- Set timeout of asynchronous calls.
- Make asynchronous call of test method.
- Make asynchronous call of test method in callback (nested).
- Proxy becomes available during first async call.
- Do checks of call status (CommonAPI::CallStatus::SUCCESS) and stored value in callback functions.

4.10.1.11 CMMMethodCalls_AsynchronousMethodCallsProxyBecomesAvailable()

```
void CMMMethodCalls_AsynchronousMethodCallsProxyBecomesAvailable ( )
```

Test Call test method asynchronous multiple times when proxy is not available. Proxy becomes available during call

- Unregiser service
- Wait that proxy is not available
- Test stub set in-value of test methods.
- Set timeouts of asynchronous calls (timeouts that are reached and timeouts that are not reached).
- Make asynchronous calls of test method (2 expected timeouts, 3 successful calls).
- Proxy becomes available during call
- Do checks of call status (CommonAPI::CallStatus::SUCCESS and CommonAPI::CallStatus::NOT_AVAILABLE for expected timeouts), stored values and timeouts that occurred in callback functions.

4.10.1.12 CMMMethodCalls_AsynchronousMethodCallProxyNotAvailableDeleteProxy()

```
void CMMMethodCalls_AsynchronousMethodCallProxyNotAvailableDeleteProxy ( )
```

Test Call test method asynchronous when proxy is not available and delete proxy.

- Unregister service.
- Wait that proxy is not available.
- Test stub sets in-value of test method.
- Set timeout of asynchronous call.
- Make asynchronous call of test method.
- Start thread which deletes the proxy.
- Check if proxy could be deleted.
- Join created thread.

4.10.1.13 CMMethodCalls_AynchronousMethodCallsReceiveNotAvailable()

```
void CMMethodCalls_AynchronousMethodCallsReceiveNotAvailable ( )
```

Test Call test method via two proxies multiple times asynchronously while the service is unavailable and check if the provided callback is called with an error for every method call done.

4.10.1.14 main()

```
int main (
    int argc,
    char ** argv )
```

4.10.2 Variable Documentation

4.10.2.1 serviceId

```
const std::string serviceId = "service-sample"
```

4.10.2.2 clientId

```
const std::string clientId = "client-sample"
```

4.10.2.3 domain

```
const std::string domain = "local"
```

4.10.2.4 testAddress

```
const std::string testAddress = "commonapi.communication.TestInterface"
```

4.10.2.5 testAddress2

```
const std::string testAddress2 = "commonapi.communication.TestInterface2"
```

4.10.2.6 tasync

```
const int tasync = 20000
```

4.10.2.7 timeout

```
const int timeout = 300
```

4.10.2.8 maxTimeoutCalls

```
const int maxTimeoutCalls = 10
```

4.10.2.9 wf

```
const unsigned int wf = 1
```

4.11 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/DTAdvanced.cpp File Reference

Functions

- void [DTAdvanced_SendAndReceive](#) ()
- void [DTAdvanced_SendAndReceiveInvalid](#) ()
- void [DTAdvanced_DISABLED_SendAndReceiveMapInvalid](#) ()
- void [DTAdvanced_AttributeSetInvalid](#) ()
- void [DTAdvanced_DISABLED_AttributeSetInvalidMapLength](#) ()
- void [DTAdvanced_AttributeSetAsyncInvalid](#) ()
- void [DTAdvanced_AttributeSet](#) ()
- void [DTAdvanced_BroadcastReceive](#) ()
- int [main](#) (int argc, char **argv)

Variables

- `const std::string domain = "local"`
- `const std::string testAddress = "commonapi.datatypes.advanced.TestInterface"`
- `const std::string connectionIdService = "service-sample"`
- `const std::string connectionIdClient = "client-sample"`
- `const int tasync = 10000`

4.11.1 Function Documentation

4.11.1.1 DTAdvanced_SendAndReceive()

```
void DTAdvanced_SendAndReceive ( )
```

4.11.1.2 DTAdvanced_SendAndReceiveInvalid()

```
void DTAdvanced_SendAndReceiveInvalid ( )
```

4.11.1.3 DTAdvanced_DISABLED_SendAndReceiveMapInvalid()

```
void DTAdvanced_DISABLED_SendAndReceiveMapInvalid ( )
```

4.11.1.4 DTAdvanced_AttributeSetInvalid()

```
void DTAdvanced_AttributeSetInvalid ( )
```

Test Test attribute functions with invalid values

- Call set function of attributes with invalid types
- Check that the attribute's value has not changed

4.11.1.5 DTAdvanced_DISABLED_AttributeSetInvalidMapLength()

```
void DTAdvanced_DISABLED_AttributeSetInvalidMapLength ( )
```

Test Test attribute functions with invalid map length

- Call set function of attributes with map length
- Check that an error returns

4.11.1.6 DTAdvanced_AttributeSetAsyncInvalid()

```
void DTAdvanced_AttributeSetAsyncInvalid ( )
```

Test Test attribute asynchronous functions with invalid values

- Call set asynch function of attributes with invalid types
- Callback should be called with error status
- Check that attribute value has not changed

4.11.1.7 DTAdvanced_AttributeSet()

```
void DTAdvanced_AttributeSet ( )
```

Test Test attribute functions with advanced types

- Call set function of attributes with advanced types
- Call get function and check if the return value is the same

4.11.1.8 DTAdvanced_BroadcastReceive()

```
void DTAdvanced_BroadcastReceive ( )
```

Test Test broadcast with advanced types

- Subscribe to broadcast which contains advanced types
- Call function to cause the stub to fire broadcast event with the same content
- Check if the values in the callback function are as expected

4.11.1.9 main()

```
int main (
    int argc,
    char ** argv )
```

4.11.2 Variable Documentation

4.11.2.1 domain

```
const std::string domain = "local"
```

4.11.2.2 testAddress

```
const std::string testAddress = "commonapi.datatypes.advanced.TestInterface"
```

4.11.2.3 connectionIdService

```
const std::string connectionIdService = "service-sample"
```

4.11.2.4 connectionIdClient

```
const std::string connectionIdClient = "client-sample"
```

4.11.2.5 tasync

```
const int tasync = 10000
```

4.12 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.↵ verification/src/DTCombined.cpp File Reference

Functions

- void [DTCombined_SendAndReceive](#) ()
- void [DTCombined_CheckInitialValue](#) ()
- void [DTCombined2_VariantWithLiteralEnum](#) ()
- int [main](#) (int argc, char **argv)

Variables

- `const std::string domain = "local"`
- `const std::string testAddress = "commonapi.datatypes.combined.TestInterface"`
- `const std::string connectionIdService = "service-sample"`
- `const std::string connectionIdClient = "client-sample"`

4.12.1 Function Documentation

4.12.1.1 DTCombined_SendAndReceive()

```
void DTCombined_SendAndReceive ( )
```

Test Test function call with combined type

- The combined type is one structure with combinations of advanced and primitive types
- Function call of a function that has for each advanced type one argument (test values) and one return value
- The stub copies the test values to the return values
- On client side the test values are compared with the return values

4.12.1.2 DTCombined_CheckInitialValue()

```
void DTCombined_CheckInitialValue ( )
```

Test Test that combined types are properly initialized

4.12.1.3 DTCombined2_VariantWithLiteralEnum()

```
void DTCombined2_VariantWithLiteralEnum ( )
```

4.12.1.4 main()

```
int main (
    int argc,
    char ** argv )
```

4.12.2 Variable Documentation

4.12.2.1 domain

```
const std::string domain = "local"
```

4.12.2.2 testAddress

```
const std::string testAddress = "commonapi.datatypes.combined.TestInterface"
```

4.12.2.3 connectionIdService

```
const std::string connectionIdService = "service-sample"
```

4.12.2.4 connectionIdClient

```
const std::string connectionIdClient = "client-sample"
```

4.13 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.↵ verification/src/DTCConstants.cpp File Reference

Functions

- void [DTCConstants_InterfaceConstants](#) ()
- void [DTCConstants_TypeCollectionConstants](#) ()
- int [main](#) (int argc, char **argv)

4.13.1 Function Documentation

4.13.1.1 DTConstants_InterfaceConstants()

```
void DTConstants_InterfaceConstants ( )
```

Test See that we can access constants in an interface and that they have correct values

4.13.1.2 DTConstants_TypeCollectionConstants()

```
void DTConstants_TypeCollectionConstants ( )
```

Test See that we can access constants in type collection and that they have correct values

4.13.1.3 main()

```
int main (
    int argc,
    char ** argv )
```

4.14 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/DTDeployment.cpp File Reference

Functions

- void [DTDeployment_TryGetNoSubscriptionAttributeWithGetterIDSetToZeroInDeployment](#) ()
- void [DTDeployment_TryGetAttributeWithGetterIDSetToZeroInDeployment](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [domain](#) = "local"
- const std::string [testAddress](#) = "commonapi.datatypes.deployment.TestInterface"
- const std::string [connectionIdService](#) = "service-sample"
- const std::string [connectionIdClient](#) = "client-sample"

4.14.1 Function Documentation

4.14.1.1 DTDeployment_TryGetNoSubscriptionAttributeWithGetterIDSetToZeroInDeployment()

```
void DTDeployment_TryGetNoSubscriptionAttributeWithGetterIDSetToZeroInDeployment ( )
```

Test Test Try to get noSubscription attribute deployed with GetterID=0 and NotifierID=0

- Set value to attribute via stub
- Set value to attribute via proxy
- Check via stub that proxy set correct value
- Try to get Attribute via proxy and make sure CallStatus::NOT_AVAILABLE is returned

4.14.1.2 DTDeployment_TryGetAttributeWithGetterIDSetToZeroInDeployment()

```
void DTDeployment_TryGetAttributeWithGetterIDSetToZeroInDeployment ( )
```

Test Test Try to get attribute deployed with GetterID=0

- Subscribe to changed event of attribute
- Set value to attribute via stub
- Make sure subscription handler was called
- Set value to attribute via proxy
- Make sure subscription handler was called
- Check via stub that proxy set correct value
- Try to get Attribute via proxy and make sure CallStatus::NOT_AVAILABLE is returned

4.14.1.3 main()

```
int main (
    int argc,
    char ** argv )
```

4.14.2 Variable Documentation

4.14.2.1 domain

```
const std::string domain = "local"
```

4.14.2.2 testAddress

```
const std::string testAddress = "commonapi.datatypes.deployment.TestInterface"
```

4.14.2.3 connectionIdService

```
const std::string connectionIdService = "service-sample"
```

4.14.2.4 connectionIdClient

```
const std::string connectionIdClient = "client-sample"
```

4.15 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/DTDerived.cpp File Reference

Functions

- void [DTDerived_SendAndReceive](#) ()
- void [DTDerived_AttributeSet](#) ()
- void [DTDerived_BroadcastReceive](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [domain](#) = "local"
- const std::string [testAddress](#) = "commonapi.datatypes.derived.TestInterface"
- const std::string [connectionId_client](#) = "client-sample"
- const std::string [connectionId_service](#) = "service-sample"
- const int [tasync](#) = 10000

4.15.1 Function Documentation

4.15.1.1 DTDerived_SendAndReceive()

```
void DTDerived_SendAndReceive ( )
```

4.15.1.2 DTDerived_AttributeSet()

```
void DTDerived_AttributeSet ( )
```

Test Test attribute functions with derived types

- Call set function of attributes with derived types
- Call get function and check if the return value is the same

4.15.1.3 DTDerived_BroadcastReceive()

```
void DTDerived_BroadcastReceive ( )
```

Test Test broadcast with derived types

- Subscribe to broadcast which contains derived types
- Call function to cause the stub to fire broadcast event with the same content
- Check if the values in the callback function are as expected

4.15.1.4 main()

```
int main (
    int argc,
    char ** argv )
```

4.15.2 Variable Documentation

4.15.2.1 domain

```
const std::string domain = "local"
```

4.15.2.2 testAddress

```
const std::string testAddress = "commonapi.datatypes.derived.TestInterface"
```

4.15.2.3 connectionId_client

```
const std::string connectionId_client = "client-sample"
```

4.15.2.4 connectionId_service

```
const std::string connectionId_service = "service-sample"
```

4.15.2.5 tasync

```
const int tasync = 10000
```

4.16 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/DTPPrimitive.cpp File Reference ↩

Functions

- void [DTPPrimitive_SendAndReceive](#) ()
- void [DTPPrimitive_AttributeSet](#) ()
- void [DTPPrimitive_BroadcastReceive](#) ()
- void [DTPPrimitive_EmptyBroadcastReceive](#) ()
- void [DTPPrimitive_RangedIntegers](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [domain](#) = "local"
- const std::string [testAddress](#) = "commonapi.datatypes.primitive.TestInterface"
- const std::string [connectionIdService](#) = "service-sample"
- const std::string [connectionIdClient](#) = "client-sample"
- const int [tasync](#) = 10000

4.16.1 Function Documentation

4.16.1.1 DTPrimitive_SendAndReceive()

```
void DTPrimitive_SendAndReceive ( )
```

Test Test function call with primitive types

- Primitive types are: uint8_t, int8_t, uint16_t, int16_t, uint32_t, int32_t, uint64_t, int64_t, bool, float, double, std::string, ByteBuffer
- Function call of a function that has for each primitive type one argument (test values) and one return value
- The stub copies the test values to the return values
- On client side the test values are compared with the return values

4.16.1.2 DTPrimitive_AttributeSet()

```
void DTPrimitive_AttributeSet ( )
```

Test Test attribute functions with primitive types

- Call set function of attributes with primitive types
- Call get function and check if the return value is the same

4.16.1.3 DTPrimitive_BroadcastReceive()

```
void DTPrimitive_BroadcastReceive ( )
```

Test Test broadcast with primitive types

- Subscribe to broadcast which contains primitive types
- Call function to cause the stub to fire broadcast event with the same content
- Check if the values in the callback function are as expected

4.16.1.4 DTPrimitive_EmptyBroadcastReceive()

```
void DTPrimitive_EmptyBroadcastReceive ( )
```

Test Test broadcast with empty broadcast

- Subscribe to broadcast which does not contain any datatypes
- Call function twice to cause the stub to fire a broadcast event
- Check if the callback function was called twice

4.16.1.5 DTPrimitive_RangedIntegers()

```
void DTPrimitive_RangedIntegers ( )
```

Test Test ranged integer functionality

4.16.1.6 main()

```
int main (
    int argc,
    char ** argv )
```

4.16.2 Variable Documentation

4.16.2.1 domain

```
const std::string domain = "local"
```

4.16.2.2 testAddress

```
const std::string testAddress = "commonapi.datatypes.primitive.TestInterface"
```

4.16.2.3 connectionIdService

```
const std::string connectionIdService = "service-sample"
```

4.16.2.4 connectionIdClient

```
const std::string connectionIdClient = "client-sample"
```

4.16.2.5 tasync

```
const int tasync = 10000
```

4.17 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.↔ verification/src/PFComplex.cpp File Reference

Functions

- void [PFComplex_Ping_Pong_Complex_Synchronous](#) ()
- void [PFComplex_Ping_Pong_Complex_Asynchronous](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const int [usecPerSecond](#) = 1000000
- const std::string [serviceId](#) = "service-sample"
- const std::string [clientId](#) = "client-sample"
- const std::string [domain](#) = "local"
- const std::string [testAddress](#) = "commonapi.performance.complex.TestInterface"
- const int [maxArraySize](#) = 4096 / 16
- const int [loopCountPerPayload](#) = 1000

4.17.1 Function Documentation

4.17.1.1 PFComplex_Ping_Pong_Complex_Synchronous()

```
void PFComplex_Ping_Pong_Complex_Synchronous ( )
```

Test Test synchronous ping pong function call

- complex array is array of a struct containing an union and another struc with primitive datatypes
- The stub just set the in array to the out array
- CallStatus and array content will be used to verify the sync call has succeeded
- Using double payload every cycle, starting with 1 end with maxPrimitiveArraySize
- Doing primitiveLoopSize loops to build the mean time

4.17.1.2 PFComplex_Ping_Pong_Complex_Asynchronous()

```
void PFComplex_Ping_Pong_Complex_Asynchronous ( )
```

Test Test asynchronous ping pong function call

- complex array is array of a struct containing an union and another struct with primitive datatypes
- The stub just set (copies) the in array to the out array
- Only the CallStatus will be used to verify the async call has succeeded
- Using double payload every cycle, starting with 1 end with maxPrimitiveArraySize
- Doing loopCountPerPayload loops to calc the mean time

4.17.1.3 main()

```
int main (
    int argc,
    char ** argv )
```

4.17.2 Variable Documentation

4.17.2.1 usecPerSecond

```
const int usecPerSecond = 1000000
```

4.17.2.2 serviceId

```
const std::string serviceId = "service-sample"
```

4.17.2.3 clientId

```
const std::string clientId = "client-sample"
```

4.17.2.4 domain

```
const std::string domain = "local"
```

4.17.2.5 testAddress

```
const std::string testAddress = "commonapi.performance.complex.TestInterface"
```

4.17.2.6 maxArraySize

```
const int maxArraySize = 4096 / 16
```

4.17.2.7 loopCountPerPayload

```
const int loopCountPerPayload = 1000
```

4.18 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.↵ verification/src/PFPrimitive.cpp File Reference

Functions

- void [PFPrimitive_Ping_Pong_Primitive_Synchronous](#) ()
- void [PFPrimitive_Ping_Pong_Primitive_Asynchronous](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [serviceld](#) = "service-sample"
- const std::string [clientId](#) = "client-sample"
- const std::string [domain](#) = "local"
- const std::string [testAddress](#) = "commonapi.performance.primitive.TestInterface"
- const int [usecPerSecond](#) = 1000000
- const int [maxPrimitiveArraySize](#) = 1024*16
- const int [loopCountPerPayload](#) = 1000

4.18.1 Function Documentation

4.18.1.1 PFPrimitive_Ping_Pong_Primitive_Synchronous()

```
void PFPrimitive_Ping_Pong_Primitive_Synchronous ( )
```

Test Test synchronous ping pong function call

- primitive array is array of UInt_8
 - The stub just set the in array to the out array
 - CallStatus and array content will be used to verify the sync call has succeeded
 - Using double payload every cycle, starting with 1 end with maxPrimitiveArraySize
 - Doing primitiveLoopSize loops to build the mean time

4.18.1.2 PFPrimitive_Ping_Pong_Primitive_Asynchronous()

```
void PFPrimitive_Ping_Pong_Primitive_Asynchronous ( )
```

Test Test asynchronous ping pong function call

- primitive array is array of UInt_8
 - The stub just set (copies) the in array to the out array
 - Only the CallStatus will be used to verify the async call has succeeded
 - Using double payload every cycle, starting with 1 end with maxPrimitiveArraySize
 - Doing primitiveLoopSize loops to build the mean time

4.18.1.3 main()

```
int main (
    int argc,
    char ** argv )
```

4.18.2 Variable Documentation

4.18.2.1 serviceId

```
const std::string serviceId = "service-sample"
```

4.18.2.2 clientId

```
const std::string clientId = "client-sample"
```

4.18.2.3 domain

```
const std::string domain = "local"
```

4.18.2.4 testAddress

```
const std::string testAddress = "commonapi.performance.primitive.TestInterface"
```

4.18.2.5 usecPerSecond

```
const int usecPerSecond = 1000000
```

4.18.2.6 maxPrimitiveArraySize

```
const int maxPrimitiveArraySize = 1024*16
```

4.18.2.7 loopCountPerPayload

```
const int loopCountPerPayload = 1000
```

4.19 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.↵ verification/src/RTBuildProxiesAndStubs.cpp File Reference

Functions

- void [RTBuildProxiesAndStubs_LoadedRuntimeCanBuildProxiesAndStubs](#) ()
- void [RTBuildProxiesAndStubs_BuildProxiesAndStubsTwoTimes](#) ()
- void [RTBuildProxiesAndStubs_BuildProxyTwoTimesWithReassigningAndStub](#) ()
- void [RTBuildProxiesAndStubs_WaitForProxyDestruction](#) ()
- void [RTBuildProxiesAndStubs_WaitForProxyDestructionCreatedInThread](#) ()
- void [RTBuildProxiesAndStubs_WaitForProxyDestructionInTwoThreads](#) ()
- void [RTBuildProxiesAndStubs_BuildProxySubscribeToProxyStatusEventBlockingCallAndShutdown](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string `domain` = "local"
- const std::string `testAddress` = "commonapi.runtime.TestInterface"
- const std::string `applicationNameService` = "service-sample"
- const std::string `applicationNameClient` = "client-sample"
- const int `tasync` = 20000

4.19.1 Function Documentation

4.19.1.1 RTBuildProxiesAndStubs_LoadedRuntimeCanBuildProxiesAndStubs()

```
void RTBuildProxiesAndStubs_LoadedRuntimeCanBuildProxiesAndStubs ( )
```

Test Loads Runtime, creates proxy and stub/service.

- Calls `CommonAPI::Runtime::get()` and checks if return value is true.
- Checks if test proxy with domain and test instance can be created.
- Checks if test stub can be created.
- Register the test service.
- Unregister the test service.

4.19.1.2 RTBuildProxiesAndStubs_BuildProxiesAndStubsTwoTimes()

```
void RTBuildProxiesAndStubs_BuildProxiesAndStubsTwoTimes ( )
```

Test Loads Runtime, creates proxy and stub/service two times.

- Calls `CommonAPI::Runtime::get()` and checks if return value is true
- Create stub and register service
- Create proxy
- Do some synchronous calls
- Unregister the service.
- Create stub and register service
- Create proxy
- Checks whether proxy is available
- Unregister the service

4.19.1.3 RTBuildProxiesAndStubs_BuildProxyTwoTimesWithReassigningAndStub()

```
void RTBuildProxiesAndStubs_BuildProxyTwoTimesWithReassigningAndStub ( )
```

Test Loads Runtime, creates proxy two times with reassigning and create stub/service.

- Calls CommonAPI::Runtime::get() and checks if return value is true
- Create proxy
- Create proxy again and reassign
- Create stub and register service
- Checks whether proxy is available
- Do synchronous calls
- Unregister the service.

4.19.1.4 RTBuildProxiesAndStubs_WaitForProxyDestruction()

```
void RTBuildProxiesAndStubs_WaitForProxyDestruction ( )
```

Test Loads Runtime, creates proxy and stub/service, await proxy destruction

- Calls CommonAPI::Runtime::get() and checks if return value is true.
- Checks if test proxy with domain and test instance can be created
- Checks if test stub can be created.
- Register the test service.
- Wait for service availability
- Unregister the test service.
- Wait for on future till proxy was destroyed after std::shared_ptr<> ref from thread was released

4.19.1.5 RTBuildProxiesAndStubs_WaitForProxyDestructionCreatedInThread()

```
void RTBuildProxiesAndStubs_WaitForProxyDestructionCreatedInThread ( )
```

Test Loads Runtime, creates proxy and stub/service, await proxy destruction

- Calls CommonAPI::Runtime::get() and checks if return value is true.
- Checks if test proxy with domain and test instance can be created (in an own thread).
- Checks if test stub can be created.
- Register the test service.
- Wait for service availability on the test proxy in it's thread.
- Unregister the test service.
- Wait till proxy was destroyed when std::shared_ptr<> in thread has been released.

4.19.1.6 RTBuildProxiesAndStubs_WaitForProxyDestructionInTwoThreads()

```
void RTBuildProxiesAndStubs_WaitForProxyDestructionInTwoThreads ( )
```

Test Loads Runtime, creates proxy and stub/service, await proxy destruction in two threads

- Calls CommonAPI::Runtime::get() and checks if return value is true.
- Checks if test proxy with domain and test instance can be created (in an own thread).
- Wait till proxy was destroyed when std::shared_ptr<> in threads
- Join the threads that have been waiting for proxy destruction

4.19.1.7 RTBuildProxiesAndStubs_BuildProxySubscribeToProxyStatusEventBlockingCallAndShutdown()

```
void RTBuildProxiesAndStubs_BuildProxySubscribeToProxyStatusEventBlockingCallAndShutdown ( )
```

Test Loads Runtime, creates proxy, subscribes to proxy status event, does a blocking call and shutdown

- Calls CommonAPI::Runtime::get() and checks if return value is true.
- Checks if test proxy with domain and test instance can be created.
- Subscribes to proxy status event and simulate a blocking call (simulated by sleep) when proxy is getting available
- Register the test service
- Initiate shutdown when blocking call was done
- Unregister test service
- Wait till proxy is getting unavailable
- Destroy proxy
- Wait till proxy was destroyed and proxy status event handler is finished

4.19.1.8 main()

```
int main (
    int argc,
    char ** argv )
```

4.19.2 Variable Documentation

4.19.2.1 domain

```
const std::string domain = "local"
```

4.19.2.2 testAddress

```
const std::string testAddress = "commonapi.runtime.TestInterface"
```

4.19.2.3 applicationNameService

```
const std::string applicationNameService = "service-sample"
```

4.19.2.4 applicationNameClient

```
const std::string applicationNameClient = "client-sample"
```

4.19.2.5 tasync

```
const int tasync = 20000
```

4.20 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.↔ verification/src/RTLoadingRuntime.cpp File Reference

Functions

- void [RTLoadingRuntime_LoadsDefaultRuntime](#) ()
- int [main](#) (int argc, char **argv)

4.20.1 Function Documentation

4.20.1.1 RTLoadingRuntime_LoadsDefaultRuntime()

```
void RTLoadingRuntime_LoadsDefaultRuntime ( )
```

Test Loads Default Runtime.

- Calls CommonAPI::Runtime::get().
- Success if return value is true.

4.20.1.2 main()

```
int main (
    int argc,
    char ** argv )
```

4.21 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/StabilitySP.cpp File Reference

Functions

- void [StabilitySP_RepeatedRegistrations](#) ()
- void [StabilitySP_MultipleMethodCalls](#) ()
- void [StabilitySP_MultipleAttributeSets](#) ()
- void [StabilitySP_MultipleAttributeGets](#) ()
- void [StabilitySP_MultipleAttributeGetAsyncs](#) ()
- void [StabilitySP_MultipleAttributeSetAsyncs](#) ()
- void [StabilitySP_MultipleAttributeSubscriptions](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [serviceld](#) = "service-sample"
- const std::string [clientId](#) = "client-sample"
- const std::string [domain](#) = "local"
- const std::string [testAddress](#) = "commonapi.stability.sp.TestInterface"
- const std::string [COMMONAPI_CONFIG_SUFFIX](#) = ".conf"
- const int [MAXSERVERCOUNT](#) = 40
- const int [MAXTHREADCOUNT](#) = 8
- const int [MAXMETHODCALLS](#) = 80
- const int [MAXREGLOOPS](#) = 16
- const int [MAXREGCOUNT](#) = 16
- const int [MESSAGESIZE](#) = 80
- const int [MAXSUBSCRIPTIONSETS](#) = 10

4.21.1 Function Documentation

4.21.1.1 StabilitySP_RepeatedRegistrations()

```
void StabilitySP_RepeatedRegistrations ( )
```

Test Register and unregister services in a loop.

- do MAXREGLOOPS times:
- register MAXREGCOUNT addresses as services
- unregister the addresses that were just registered
- check the return code of each register/unregister call
- test fails if any of the return codes are false

4.21.1.2 StabilitySP_MultipleMethodCalls()

```
void StabilitySP_MultipleMethodCalls ( )
```

Test Create a number of services and proxies and send messages through them.

- Register MAXSERVERCOUNT addresses as services
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then sends MAXMETHODCALLS messages to each.
- Each message is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the return message from the server is not correct

4.21.1.3 StabilitySP_MultipleAttributeSets()

```
void StabilitySP_MultipleAttributeSets ( )
```

Test Create a number of services and proxies and set attributes through them.

- Register MAXSERVERCOUNT addresses as services
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then sets attributes MAXMETHODCALLS times to each.
- Each attribute is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the return attribute from the server is not correct

4.21.1.4 StabilitySP_MultipleAttributeGets()

```
void StabilitySP_MultipleAttributeGets ( )
```

Test Create a number of services and proxies and get attributes through them.

- Register MAXSERVERCOUNT addresses as services
 - Set the attribute for service, at the stub side.
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then gets attributes MAXMETHODCALLS times for each.
- Each attribute is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the returned attribute from the server is not correct

4.21.1.5 StabilitySP_MultipleAttributeGetAsyncns()

```
void StabilitySP_MultipleAttributeGetAsyncns ( )
```

Test Create a number of services and proxies and get attributes through them.

- Register MAXSERVERCOUNT addresses as services
 - Set the attribute for service, at the stub side.
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then gets attributes MAXMETHODCALLS times for each asynchronously
- Each attribute is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the callbacks are not called correct number of times

4.21.1.6 StabilitySP_MultipleAttributeSetAsyncns()

```
void StabilitySP_MultipleAttributeSetAsyncns ( )
```

Test Create a number of services and proxies and set attributes through them.

- Register MAXSERVERCOUNT addresses as services
 - Set the attribute for service, at the stub side.
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then sets attributes MAXMETHODCALLS times for each asynchronously
- Each attribute is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the callbacks are not called correct number of times

4.21.1.7 StabilitySP_MultipleAttributeSubscriptions()

```
void StabilitySP_MultipleAttributeSubscriptions ( )
```

Test Create a number of services and proxies and set attributes through them.

- Register MAXSERVERCOUNT addresses as services
 - Set the attribute for service, at the stub side.
- Create MAXTHREADCOUNT threads, each of which creates a proxy for each service address and then sets attributes MAXMETHODCALLS times for each asynchronously
- Each attribute is MESSAGE_SIZE bytes long.
- Test fails if any of the services fail to get registered or if any of the proxies won't get available or if the callbacks are not called correct number of times

4.21.1.8 main()

```
int main (
    int argc,
    char ** argv )
```

4.21.2 Variable Documentation

4.21.2.1 serviceId

```
const std::string serviceId = "service-sample"
```

4.21.2.2 clientId

```
const std::string clientId = "client-sample"
```

4.21.2.3 domain

```
const std::string domain = "local"
```

4.21.2.4 testAddress

```
const std::string testAddress = "commonapi.stability.sp.TestInterface"
```

4.21.2.5 COMMONAPI_CONFIG_SUFFIX

```
const std::string COMMONAPI_CONFIG_SUFFIX = ".conf"
```

4.21.2.6 MAXSERVERCOUNT

```
const int MAXSERVERCOUNT = 40
```

4.21.2.7 MAXTHREADCOUNT

```
const int MAXTHREADCOUNT = 8
```

4.21.2.8 MAXMETHODCALLS

```
const int MAXMETHODCALLS = 80
```

4.21.2.9 MAXREGLOOPS

```
const int MAXREGLOOPS = 16
```

4.21.2.10 MAXREGCOUNT

```
const int MAXREGCOUNT = 16
```

4.21.2.11 MESSAGESIZE

```
const int MESSAGESIZE = 80
```

4.21.2.12 MAXSUBSCRIPTIONSETS

```
const int MAXSUBSCRIPTIONSETS = 10
```

4.22 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/THMainLoopIndependence.cpp File Reference

Functions

- void [THMainLoopIndependence_ProxyReceivesAnswerOnlyIfStubMainLoopRuns](#) ()
- void [THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersSync](#) ()
- void [THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersAsync](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string `domain` = "local"
- const std::string `instance6` = "my.test.commonapi.address.six"
- const std::string `instance7` = "my.test.commonapi.address.seven"
- const std::string `instance8` = "my.test.commonapi.address.eight"
- const std::string `mainloopName1` = "client-sample"
- const std::string `mainloopName2` = "service-sample"
- const std::string `thirdPartyServiceId` = "mainloop-thirdParty"
- const int `tasync` = 10000

4.22.1 Function Documentation

4.22.1.1 THMainLoopIndependence_ProxyReceivesAnswerOnlyIfStubMainLoopRuns()

```
void THMainLoopIndependence_ProxyReceivesAnswerOnlyIfStubMainLoopRuns ( )
```

Test Proxy Receives Answer Only If Stub MainLoop Runs.

- start proxy in thread 1 and call testPredefinedTypeMethod
- proxy should not receive answer, if the stub mainloop does not run
- run mainloop of stub
- now the stub mainloop also runs, so the proxy should receive the answer

4.22.1.2 THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersSync()

```
void THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersSync ( )
```

Test Proxy Receives Just His Own Answers.

- start 2 proxies in own threads
- call test method in each proxy synchronously
- now each proxy should have received the answer to his own request

4.22.1.3 THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersAsync()

```
void THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersAsync ( )
```

Test Proxy Receives Just His Own Answers.

- start 2 proxies in own threads
- call test method in each proxy asynchronously
- now each proxy should have received the answer to his own request

4.22.1.4 main()

```
int main (
    int argc,
    char ** argv )
```

4.22.2 Variable Documentation

4.22.2.1 domain

```
const std::string domain = "local"
```

4.22.2.2 instance6

```
const std::string instance6 = "my.test.commonapi.address.six"
```

4.22.2.3 instance7

```
const std::string instance7 = "my.test.commonapi.address.seven"
```

4.22.2.4 instance8

```
const std::string instance8 = "my.test.commonapi.address.eight"
```

4.22.2.5 mainloopName1

```
const std::string mainloopName1 = "client-sample"
```

4.22.2.6 mainloopName2

```
const std::string mainloopName2 = "service-sample"
```

4.22.2.7 thirdPartyServiceId

```
const std::string thirdPartyServiceId = "mainloop-thirdParty"
```

4.22.2.8 tasync

```
const int tasync = 10000
```

4.23 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.↔ verification/src/THMainLoopIntegration.cpp File Reference

Functions

- void [THMainLoopIntegration_VerifyCommunicationWithMainLoop](#) ()
- void [THMainLoopIntegration_VerifyTransportReading](#) ()
- void [THMainLoopIntegration_VerifySyncCallMessageHandlingOrder](#) ()
- void [THMainLoopIntegration_SelectiveErrorHandlerWithMainLoop](#) ()
- void [THMainLoopIntegration_AsynchronousMethodCallsReceiveNotAvailable](#) ()
- void [THMainLoopIntegration_CreateProxyToManagerInSameProcess](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [domain](#) = "local"
- const std::string [instance](#) = "my.test.commonapi.address"
- const std::string [connection_client](#) = "client-sample"
- const std::string [connection_service](#) = "service-sample"
- const int [tasync](#) = 10000

4.23.1 Function Documentation

4.23.1.1 THMainLoopIntegration_VerifyCommunicationWithMainLoop()

```
void THMainLoopIntegration_VerifyCommunicationWithMainLoop ( )
```

Test Verifies communication with Main Loop.

- get proxy with available flag = true
- generate big test data
- send synchronous test message

4.23.1.2 THMainLoopIntegration_VerifyTransportReading()

```
void THMainLoopIntegration_VerifyTransportReading ( )
```

Test Verifies Transport Reading When Dispatching Watches.

- get proxy with available flag = true
- generate big test data
- send asynchronous test message
- dispatch dispatchSource: the message must not be arrived
- dispatch watches (reads transport).
- dispatch dispatchSources again: now the message must be arrived.

4.23.1.3 THMainLoopIntegration_VerifySyncCallMessageHandlingOrder()

```
void THMainLoopIntegration_VerifySyncCallMessageHandlingOrder ( )
```

Test Verifies Synchronous Call Message Handling Order.

- get proxy with available flag = true
- subscribe for broadcast event
- generate 5 test broadcasts
- 5 broadcasts should arrive in the right order

4.23.1.4 THMainLoopIntegration_SelectiveErrorHandlerWithMainLoop()

```
void THMainLoopIntegration_SelectiveErrorHandlerWithMainLoop ( )
```

Test Verifies SelectiveError Handler is called correctly when used with mainloop

- get proxy with available flag = true
- Subscribe for selective Event and register error handler
- Stub fires event upon subscription
- Check that subscription handler and error handler were both called once
- Unregister Service and register Service again
- Check that subscription error handler was called again after service went offline and came online again (resubscription took place) and that the event was received a second time

4.23.1.5 THMainLoopIntegration_AsynchronousMethodCallsReceiveNotAvailable()

```
void THMainLoopIntegration_AsynchronousMethodCallsReceiveNotAvailable ( )
```

Test Call test method multiple times asynchronously while the service is unavailable and check if the provided callback is called with an error for every method call done.

4.23.1.6 THMainLoopIntegration_CreateProxyToManagerInSameProcess()

```
void THMainLoopIntegration_CreateProxyToManagerInSameProcess ( )
```

Test Offer a interface manager and build two proxies to it. One proxy uses the same connection as the manager while the other uses a different connection. Check that both proxies get available and receive a available event

4.23.1.7 main()

```
int main (
    int argc,
    char ** argv )
```

4.23.2 Variable Documentation

4.23.2.1 domain

```
const std::string domain = "local"
```

4.23.2.2 instance

```
const std::string instance = "my.test.commonapi.address"
```

4.23.2.3 connection_client

```
const std::string connection_client = "client-sample"
```

4.23.2.4 connection_service

```
const std::string connection_service = "service-sample"
```

4.23.2.5 tasync

```
const int tasync = 10000
```

4.24 /home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/THMainLoopTwoThreads.cpp File Reference

Functions

- void [THMainLoopTwoThreads_ProxyGetsAvailableStatus](#) ()
- void [THMainLoopTwoThreads_ProxyGetsFunctionResponse](#) ()
- int [main](#) (int argc, char **argv)

Variables

- const std::string [domain](#) = "local"
- const std::string [instance](#) = "my.test.commonapi.address"

4.24.1 Function Documentation

4.24.1.1 THMainLoopTwoThreads_ProxyGetsAvailableStatus()

```
void THMainLoopTwoThreads_ProxyGetsAvailableStatus ( )
```

Test Proxy Receives Available when MainLoop Dispatched sourced out to other thread.

4.24.1.2 THMainLoopTwoThreads_ProxyGetsFunctionResponse()

```
void THMainLoopTwoThreads_ProxyGetsFunctionResponse ( )
```

Test Proxy gets function response when MainLoop Dispatched sourced out to other thread.

4.24.1.3 main()

```
int main (  
    int argc,  
    char ** argv )
```

4.24.2 Variable Documentation

4.24.2.1 domain

```
const std::string domain = "local"
```

4.24.2.2 instance

```
const std::string instance = "my.test.commonapi.address"
```

Index

[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/27](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/29](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/38](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/42](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/48](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/45](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/59](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/62](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/65](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/71](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/74](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/76](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/77](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/79](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/81](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/84](#)

[examples-for-windows/org.genivi.commonapi.core.verification/src/AFExtended.cpp,](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/AFManaged.cpp,](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/AFPolymorph.cpp,](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/AFSelective.cpp,](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/CMAAttributeSubscription.cpp,](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/CMAAttributes.cpp,](#)
[/home/guojunfeng/SourceCode/wapeasy_github/commonapi-examples-for-windows/org.genivi.commonapi.core.verification/src/CMBroadcasts.cpp,](#)
[AFExtended_Broadcast,](#) [28](#)
[AFExtended_MethodCall,](#) [27](#)
[clientId,](#) [28](#)
[domain,](#) [28](#)
[main,](#) [28](#)
[serviceId,](#) [28](#)
[tasync,](#) [29](#)
[testAddressBase,](#) [28](#)
[testAddressOnce,](#) [28](#)
[testAddressTwice,](#) [29](#)
[AFExtended_Attributes](#)
[AFExtended_Constants.cpp,](#) [27](#)
[AFExtended_Broadcast](#)
[AFExtended.cpp,](#) [28](#)
[AFExtended_MethodCall](#)
[AFExtended.cpp,](#) [27](#)
[AFManaged.cpp](#)
[AFManaged_AddRemoveHierarchicalManagedInterface,](#) [37](#)
[AFManaged_AddRemoveManagedInterfaceMultiple,](#) [31](#)
[AFManaged_AddRemoveManagedInterfaceSingle,](#) [31](#)
[AFManaged_AddRemoveMultipleManagedInterfacesMultiple,](#) [32](#)
[AFManaged_AddRemoveMultipleManagedInterfacesMultipleProxyNo](#)

- AFSelective.cpp
 - AFSelective_Fire_Selective_Within_Subscription_Changed_Attributes.cpp, 44
 - AFSelective_Multiple_Subscriptions_SameConnection_CallHandle.cpp, 43
 - AFSelective_ProxyBuildAndDestroy, 43
 - AFSelective_SelectiveBroadcast, 42
 - AFSelective_SelectiveBroadcastRejected, 42
 - AFSelective_SelectiveMultiBroadcast, 43
 - AFSelective_SelectiveRejectedMultiBroadcast, 43
 - AFSelective_Two_proxies_subscribe_delete_one_proxy, 44
 - AFSelective_Two_proxies_subscribe_delete_one_proxy_error_Attributes.cpp, 44
 - clientId, 44
 - domain, 45
 - main, 44
 - otherclientId, 44
 - servicId, 44
 - tasync, 45
 - testAddress, 45
- AFSelective_Fire_Selective_Within_Subscription_Changed_Attributes.cpp, 44
- AFSelective_Multiple_Subscriptions_SameConnection_CallHandle.cpp, 43
- AFSelective_ProxyBuildAndDestroy
 - AFSelective.cpp, 43
- AFSelective_SelectiveBroadcast
 - AFSelective.cpp, 42
- AFSelective_SelectiveBroadcastRejected
 - AFSelective.cpp, 42
- AFSelective_SelectiveMultiBroadcast
 - AFSelective.cpp, 43
- AFSelective_SelectiveRejectedMultiBroadcast
 - AFSelective.cpp, 43
- AFSelective_Two_proxies_subscribe_delete_one_proxy
 - AFSelective.cpp, 44
- AFSelective_Two_proxies_subscribe_delete_one_proxy_error_Attributes.cpp, 44
- applicationNameClient
 - RTBuildProxiesAndStubs.cpp, 92
- applicationNameService
 - RTBuildProxiesAndStubs.cpp, 92
- BOTTOM_INTERFACE
 - AFManaged.cpp, 31
- clientId
 - AFExtended.cpp, 28
 - AFSelective.cpp, 44
 - CMAAttributes.cpp, 47
 - CMAAttributeSubscription.cpp, 57
 - CMBlockingCalls.cpp, 61
 - CMBroadcasts.cpp, 64
 - CMMethodCalls.cpp, 70
 - PFComplex.cpp, 85
 - PFPrimitive.cpp, 87
 - StabilitySP.cpp, 96
- clientId2
 - CMBlockingCalls.cpp, 61
 - CMAAttributes.cpp
 - clientId, 47
 - CMAAttributes_AttributeGetAsynchronous, 46
 - CMAAttributes_AttributeGetSynchronous, 45
 - CMAAttributes_AttributeSetAsynchronous, 46
 - CMAAttributes_AttributeSetSynchronous, 46
 - CMAAttributes_AttributeSubscription, 47
 - domain, 48
 - main, 47
 - servicId, 47
 - tasync, 48
 - testAddress, 48
 - CMAAttributes_AttributeGetAsynchronous
 - CMAAttributes.cpp, 46
 - CMAAttributes_AttributeGetSynchronous
 - CMAAttributes.cpp, 45
 - CMAAttributes_AttributeSetAsynchronous
 - CMAAttributes.cpp, 46
 - CMAAttributes_AttributeSetSynchronous
 - CMAAttributes.cpp, 46
 - CMAAttributes_AttributeSubscription
 - CMAAttributes.cpp, 47
 - CMAAttributeSubscription.cpp
 - clientId, 57
 - CMAAttributeSubscription_DISABLED_SubscribeAndUnsubscribeImp, 51
 - CMAAttributeSubscription_SubscribeAndUnsubscribeAndReSubscribe, 55
 - CMAAttributeSubscription_SubscribeAndUnsubscribeSequentially, 51
 - CMAAttributeSubscription_SubscribeAndUnsubscribeTwoCallbacksCo, 50
 - CMAAttributeSubscription_SubscribeAndUnsubscribeUnsubscribe, 52
 - CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllRes, 55
 - CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAllRes, 56
 - CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeOneR, 56
 - CMAAttributeSubscription_SubscribeSecondProxyLater, 54
 - CMAAttributeSubscription_SubscribeServiceNotAvailable, 52
 - CMAAttributeSubscription_SubscribeThreeCallbacksServiceAvailable, 54
 - CMAAttributeSubscription_SubscribeThreeCallbacksServiceNotAvaila, 54
 - CMAAttributeSubscription_SubscribeUnregisterNoValueSetRegisterSe, 53
 - CMAAttributeSubscription_SubscribeUnregisterSetValueRegisterServi, 53
 - CMAAttributeSubscription_SubscriptionMultithreading, 50
 - CMAAttributeSubscription_SubscriptionOnAvailable, 49
 - CMAAttributeSubscription_SubscriptionStandard,

- 49
- CMAAttributeSubscription_SubscriptionUnsubscribeFromCallback, 50
- daemonAddress, 58
- daemonId, 57
- data_cond, 58
- data_queue, 58
- domain, 58
- main, 57
- mut, 58
- ProxyPtr, 49
- servicId, 57
- testAddress, 58
- testSubscription, 49
- wf, 58
- wt, 58
- CMAAttributeSubscription_DISABLED_SubscribeAndUnsubscribe, 51
- CMAAttributeSubscription.cpp, 51
- CMAAttributeSubscription_SubscribeAndUnsubscribeAndReceive, 55
- CMAAttributeSubscription.cpp, 55
- CMAAttributeSubscription_SubscribeAndUnsubscribeSequential, 51
- CMAAttributeSubscription.cpp, 51
- CMAAttributeSubscription_SubscribeAndUnsubscribeTwoCallbacks, 50
- CMAAttributeSubscription.cpp, 50
- CMAAttributeSubscription_SubscribeAndUnsubscribeUnsubscribe, 52
- CMAAttributeSubscription.cpp, 52
- CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribe, 55
- CMAAttributeSubscription.cpp, 55
- CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeAll, 56
- CMAAttributeSubscription.cpp, 56
- CMAAttributeSubscription_SubscribeMultipleProxysUnsubscribeOne, 56
- CMAAttributeSubscription.cpp, 56
- CMAAttributeSubscription_SubscribeSecondProxyLater, 54
- CMAAttributeSubscription.cpp, 54
- CMAAttributeSubscription_SubscribeServiceNotAvailable, 52
- CMAAttributeSubscription.cpp, 52
- CMAAttributeSubscription_SubscribeThreeCallbacksServiceAvailable, 54
- CMAAttributeSubscription.cpp, 54
- CMAAttributeSubscription_SubscribeThreeCallbacksServiceNotAvailable, 54
- CMAAttributeSubscription.cpp, 54
- CMAAttributeSubscription_SubscribeUnregisterNoValueSetRegisters, 53
- CMAAttributeSubscription.cpp, 53
- CMAAttributeSubscription_SubscribeUnregisterSetValueRegisters, 53
- CMAAttributeSubscription.cpp, 53
- CMAAttributeSubscription_SubscriptionMultithreading, 50
- CMAAttributeSubscription.cpp, 50
- CMAAttributeSubscription_SubscriptionOnAvailable, 49
- CMAAttributeSubscription.cpp, 49
- CMAAttributeSubscription_SubscriptionStandard, 49
- CMAAttributeSubscription.cpp, 49
- CMAAttributeSubscription_SubscriptionUnsubscribeFromCallback, 50
- CMAAttributeSubscription.cpp, 50
- CMBlockingCalls.cpp
 - clientId, 61
 - clientId2, 61
 - CMBlockingCalls_BlockInAvailabilityHandler, 60
 - CMBlockingCalls_BlockInAvailabilityHandlerAndReceiveCallbacks, 60
 - CMBlockingCalls_BlockInProxyCallback, 59
 - CMBlockingCalls_BlockInStubMethod, 59
 - CMBlockingCalls_NestedBlockInStubMethods, 60
 - domain, 61
 - main, 60
 - maxTimeoutCalls, 62
 - servicId, 61
 - tasync, 61
 - testAddress, 61
 - testAddress2, 61
 - timeout, 61
 - wf, 62
- CMBlockingCalls_BlockInAvailabilityHandler
 - CMBlockingCalls.cpp, 60
- CMBlockingCalls_BlockInAvailabilityHandlerAndReceiveCallbacks
 - CMBlockingCalls.cpp, 60
- CMBlockingCalls_BlockInProxyCallback
 - CMBlockingCalls.cpp, 59
- CMBlockingCalls_BlockInStubMethod
 - CMBlockingCalls.cpp, 59
- CMBlockingCalls_NestedBlockInStubMethods
 - CMBlockingCalls.cpp, 60
- CMBlockingCalls.cpp
 - clientId, 64
- CMBroadcasts_BroadcastStubGoesOfflineOnlineAgain, 63
- CMBroadcasts_NormalBroadcast, 62
- CMBroadcasts_NormalBroadcast_Two_proxies_subscribe_and_one_resubscribeSameEventgroup, 64
- CMBroadcasts_SelectiveBroadcast, 63
- CMBroadcasts_SelectiveBroadcastStubGoesOfflineOnlineAgain, 63
- CMBroadcasts_SelectiveBroadcastStubGoesOfflineOnlineAgain, 63
- CMBroadcasts_Two_proxies_subscribe_delete_one_proxy_status_listen, 64
- domain, 65
- main, 64
- otherclientId, 65
- Available, 64
- tasync, 65
- testAddress, 65
- wf, 65
- CMBroadcasts_BroadcastStubGoesOfflineOnlineAgain
 - CMBroadcasts.cpp, 63
- CMBroadcasts_NormalBroadcast
 - CMBroadcasts.cpp, 62
- CMBroadcasts_NormalBroadcast_Two_proxies_subscribe_and_one_resubscribe, 64
- CMBroadcasts.cpp, 64
- CMBroadcasts_SelectiveBroadcast
 - CMBroadcasts.cpp, 63
- CMBroadcasts_SelectiveBroadcastRejected
 - CMBroadcasts.cpp, 63
- CMBroadcasts_SelectiveBroadcastStubGoesOfflineOnlineAgain
 - CMBroadcasts.cpp, 63
- CMBroadcasts_Two_proxies_subscribe_delete_one_proxy_status_listen, 64
- CMBroadcasts.cpp, 64
- CMAAttributeCalls.cpp
 - clientId, 70

- CMMethodCalls_AsyncronousMethodCall, [66](#)
- CMMethodCalls_AsyncronousMethodCallProxyBecomesAvailable, [68](#)
- CMMethodCalls_AsyncronousMethodCallProxyNotAvailable, [67](#)
- CMMethodCalls_AsyncronousMethodCallProxyNotAvailableDeleteProxy, [69](#)
- CMMethodCalls_AsyncronousMethodCallsProxyBecomesAvailable, [69](#)
- CMMethodCalls_AsyncronousMethodCallsReceiveNotAvailable, [69](#)
- CMMethodCalls_FireAndForget, [66](#)
- CMMethodCalls_NestedAsyncronousMethodCall, [67](#)
- CMMethodCalls_NestedAsyncronousMethodCallProxyBecomesAvailable, [68](#)
- CMMethodCalls_NestedAsyncronousMethodCallProxyNotAvailable, [68](#)
- CMMethodCalls_NestedAsyncronousMethodCallsTimedOut, [67](#)
- CMMethodCalls_NestedSynchronousMethodCall, [67](#)
- CMMethodCalls_SynchronousMethodCall, [66](#)
- domain, [70](#)
- main, [70](#)
- maxTimeoutCalls, [71](#)
- serviceld, [70](#)
- tasync, [71](#)
- testAddress, [70](#)
- testAddress2, [70](#)
- timeout, [71](#)
- wf, [71](#)
- CMMethodCalls_AsyncronousMethodCall
 - CMMethodCalls.cpp, [66](#)
- CMMethodCalls_AsyncronousMethodCallProxyBecomesAvailable
 - CMMethodCalls.cpp, [68](#)
- CMMethodCalls_AsyncronousMethodCallProxyNotAvailable
 - CMMethodCalls.cpp, [67](#)
- CMMethodCalls_AsyncronousMethodCallProxyNotAvailableDeleteProxy
 - CMMethodCalls.cpp, [69](#)
- CMMethodCalls_AsyncronousMethodCallsProxyBecomesAvailable
 - CMMethodCalls.cpp, [69](#)
- CMMethodCalls_AsyncronousMethodCallsReceiveNotAvailable
 - CMMethodCalls.cpp, [69](#)
- CMMethodCalls_FireAndForget
 - CMMethodCalls.cpp, [66](#)
- CMMethodCalls_NestedAsyncronousMethodCall
 - CMMethodCalls.cpp, [67](#)
- CMMethodCalls_NestedAsyncronousMethodCallProxyBecomesAvailable
 - CMMethodCalls.cpp, [68](#)
- CMMethodCalls_NestedAsyncronousMethodCallProxyNotAvailable
 - CMMethodCalls.cpp, [68](#)
- CMMethodCalls_NestedAsyncronousMethodCallsTimedOut
 - CMMethodCalls.cpp, [67](#)
- CMMethodCalls_NestedSynchronousMethodCall
 - CMMethodCalls.cpp, [67](#)
- CMMethodCalls_SynchronousMethodCall
 - CMMethodCalls.cpp, [66](#)
- COMMONAPI_CONFIG_SUFFIX
 - StabilitySP.cpp, [96](#)
- connection_client
 - THMainLoopIntegration.cpp, [102](#)
- connection_service
 - THMainLoopIntegration.cpp, [102](#)
- connectionId_client
 - DTDerived.cpp, [80](#)
- DTDerived
 - AFPolymorph.cpp, [41](#)
 - DTDerived.cpp, [81](#)
- connectionIdClient
 - DTAdvanced.cpp, [74](#)
 - DTDeployment.cpp, [79](#)
- DTAdvanced
 - DTAdvanced.cpp, [74](#)
- DTCombined
 - DTCombined.cpp, [76](#)
- DTDeployment
 - DTDeployment.cpp, [79](#)
- DTPrimitive
 - DTPrimitive.cpp, [83](#)
- daemonAddress
 - CMAAttributeSubscription.cpp, [58](#)
- daemonId
 - CMAAttributeSubscription.cpp, [57](#)
- data_cond
 - CMAAttributeSubscription.cpp, [58](#)
- data_queue
 - CMAAttributeSubscription.cpp, [58](#)
- domain
 - AFExtended.cpp, [28](#)
 - AFManaged.cpp, [38](#)
 - AFPolymorph.cpp, [41](#)
 - AFSelective.cpp, [45](#)
 - CMAAttributes.cpp, [48](#)
 - CMAAttributeSubscription.cpp, [58](#)
 - CMBlockingCalls.cpp, [61](#)
 - CMBroadcasts.cpp, [65](#)
 - CMMethodCalls.cpp, [70](#)
 - DTAdvanced.cpp, [74](#)
 - DTCombined.cpp, [76](#)
 - DTDeployment.cpp, [78](#)
 - DTDerived.cpp, [80](#)
 - DTPrimitive.cpp, [83](#)
 - PFCComplex.cpp, [85](#)
 - PFPPrimitive.cpp, [88](#)
 - RTBuildProxiesAndStubs.cpp, [91](#)
 - StabilitySP.cpp, [96](#)
 - THMainLoopIndependence.cpp, [99](#)
 - THMainLoopIntegration.cpp, [102](#)
 - THMainLoopTwoThreads.cpp, [104](#)
- DTAdvanced
 - connectionIdClient, [74](#)
 - connectionIdService, [74](#)
 - domain, [74](#)
 - DTAdvanced_AttributeSet, [73](#)
 - DTAdvanced_AttributeSetAsyncInvalid, [73](#)

- DTAdvanced_AttributeSetInvalid, 72
- DTAdvanced_BroadcastReceive, 73
- DTAdvanced_DISABLED_AttributeSetInvalidMapLength, 72
- DTAdvanced_DISABLED_SendAndReceiveMapInvalid, 72
- DTAdvanced_SendAndReceive, 72
- DTAdvanced_SendAndReceiveInvalid, 72
- main, 73
- tasync, 74
- testAddress, 74
- DTAdvanced_AttributeSet
 - DTAdvanced.cpp, 73
- DTAdvanced_AttributeSetAsyncInvalid
 - DTAdvanced.cpp, 73
- DTAdvanced_AttributeSetInvalid
 - DTAdvanced.cpp, 72
- DTAdvanced_BroadcastReceive
 - DTAdvanced.cpp, 73
- DTAdvanced_DISABLED_AttributeSetInvalidMapLength
 - DTAdvanced.cpp, 72
- DTAdvanced_DISABLED_SendAndReceiveMapInvalid
 - DTAdvanced.cpp, 72
- DTAdvanced_SendAndReceive
 - DTAdvanced.cpp, 72
- DTAdvanced_SendAndReceiveInvalid
 - DTAdvanced.cpp, 72
- DTCombined.cpp
 - connectionIdClient, 76
 - connectionIdService, 76
 - domain, 76
 - DTCombined2_VariantWithLiteralEnum, 75
 - DTCombined_CheckInitialValue, 75
 - DTCombined_SendAndReceive, 75
 - main, 75
 - testAddress, 76
- DTCombined2_VariantWithLiteralEnum
 - DTCombined.cpp, 75
- DTCombined_CheckInitialValue
 - DTCombined.cpp, 75
- DTCombined_SendAndReceive
 - DTCombined.cpp, 75
- DTConstants.cpp
 - DTConstants_InterfaceConstants, 76
 - DTConstants_TypeCollectionConstants, 77
 - main, 77
- DTConstants_InterfaceConstants
 - DTConstants.cpp, 76
- DTConstants_TypeCollectionConstants
 - DTConstants.cpp, 77
- DTDeployment.cpp
 - connectionIdClient, 79
 - connectionIdService, 79
 - domain, 78
 - DTDeployment_TryGetAttributeWithGetterIDSetToZeroInDeployment, 78
 - DTDeployment_TryGetNoSubscriptionAttributeWithGetterIDSetToZeroInDeployment, 77
- main, 78
- testAddress, 78
- DTDeployment_TryGetAttributeWithGetterIDSetToZeroInDeployment
 - DTDeployment.cpp, 78
- DTDeployment_TryGetNoSubscriptionAttributeWithGetterIDSetToZeroInDeployment
 - DTDeployment.cpp, 77
- DTDerived.cpp
 - connectionId_client, 80
 - connectionId_service, 81
 - domain, 80
 - DTDerived_AttributeSet, 79
 - DTDerived_BroadcastReceive, 80
 - DTDerived_SendAndReceive, 79
 - main, 80
 - tasync, 81
 - testAddress, 80
- DTDerived_AttributeSet
 - DTDerived.cpp, 79
- DTDerived_BroadcastReceive
 - DTDerived.cpp, 80
- DTDerived_SendAndReceive
 - DTDerived.cpp, 79
- DTPrimitive.cpp
 - connectionIdClient, 83
 - connectionIdService, 83
 - domain, 83
 - DTPrimitive_AttributeSet, 82
 - DTPrimitive_BroadcastReceive, 82
 - DTPrimitive_EmptyBroadcastReceive, 82
 - DTPrimitive_RangedIntegers, 82
 - DTPrimitive_SendAndReceive, 81
 - main, 83
 - tasync, 83
 - testAddress, 83
- DTPrimitive_AttributeSet
 - DTPrimitive.cpp, 82
- DTPrimitive_BroadcastReceive
 - DTPrimitive.cpp, 82
- DTPrimitive_EmptyBroadcastReceive
 - DTPrimitive.cpp, 82
- DTPrimitive_RangedIntegers
 - DTPrimitive.cpp, 82
- DTPrimitive_SendAndReceive
 - DTPrimitive.cpp, 81
- instance
 - THMainLoopIntegration.cpp, 102
 - THMainLoopTwoThreads.cpp, 104
- instance6
 - THMainLoopIndependence.cpp, 99
- instance7
 - THMainLoopIndependence.cpp, 99
- instance8
 - THMainLoopIndependence.cpp, 99
- INTERFACE_DEVICE
 - AFManaged.cpp, 30
- INTERFACE_SPECIAL_DEVICE
 - AFManaged.cpp, 30

- loopCountPerPayload
 - PFCComplex.cpp, [86](#)
 - PFPrimitive.cpp, [88](#)
- main
 - AFExtended.cpp, [28](#)
 - AFManaged.cpp, [38](#)
 - AFPolymorph.cpp, [40](#)
 - AFSelective.cpp, [44](#)
 - CMAAttributes.cpp, [47](#)
 - CMAAttributeSubscription.cpp, [57](#)
 - CMBlockingCalls.cpp, [60](#)
 - CMBroadcasts.cpp, [64](#)
 - CMMethodCalls.cpp, [70](#)
 - DTAdvanced.cpp, [73](#)
 - DTCombined.cpp, [75](#)
 - DTConstants.cpp, [77](#)
 - DTDeployment.cpp, [78](#)
 - DTDerived.cpp, [80](#)
 - DTPrimitive.cpp, [83](#)
 - PFCComplex.cpp, [85](#)
 - PFPrimitive.cpp, [87](#)
 - RTBuildProxiesAndStubs.cpp, [91](#)
 - RTLLoadingRuntime.cpp, [92](#)
 - StabilitySP.cpp, [95](#)
 - THMainLoopIndependence.cpp, [98](#)
 - THMainLoopIntegration.cpp, [102](#)
 - THMainLoopTwoThreads.cpp, [103](#)
- mainloopName1
 - THMainLoopIndependence.cpp, [99](#)
- mainloopName2
 - THMainLoopIndependence.cpp, [99](#)
- mainpagetests/01_mainpage.dox, [27](#)
- maxArraySize
 - PFCComplex.cpp, [86](#)
- MAXMETHODCALLS
 - StabilitySP.cpp, [97](#)
- maxPrimitiveArraySize
 - PFPrimitive.cpp, [88](#)
- MAXREGCOUNT
 - StabilitySP.cpp, [97](#)
- MAXREGLOOPS
 - StabilitySP.cpp, [97](#)
- MAXSERVERCOUNT
 - StabilitySP.cpp, [96](#)
- MAXSUBSCRIPTIONSETS
 - StabilitySP.cpp, [97](#)
- MAXTHREADCOUNT
 - StabilitySP.cpp, [96](#)
- maxTimeoutCalls
 - CMBlockingCalls.cpp, [62](#)
 - CMMethodCalls.cpp, [71](#)
- MESSAGESIZE
 - StabilitySP.cpp, [97](#)
- MIDDLE_INTERFACE
 - AFManaged.cpp, [30](#)
- mut
 - CMAAttributeSubscription.cpp, [58](#)
- otherclientId
 - AFSelective.cpp, [44](#)
 - CMBroadcasts.cpp, [65](#)
- PFCComplex.cpp
 - clientId, [85](#)
 - domain, [85](#)
 - loopCountPerPayload, [86](#)
 - main, [85](#)
 - maxArraySize, [86](#)
 - PFCComplex_Ping_Pong_Complex_Asynchronous, [84](#)
 - PFCComplex_Ping_Pong_Complex_Synchronous, [84](#)
 - serviceld, [85](#)
 - testAddress, [86](#)
 - usecPerSecond, [85](#)
- PFCComplex_Ping_Pong_Complex_Asynchronous
 - PFCComplex.cpp, [84](#)
- PFCComplex_Ping_Pong_Complex_Synchronous
 - PFCComplex.cpp, [84](#)
- PFPrimitive.cpp
 - clientId, [87](#)
 - domain, [88](#)
 - loopCountPerPayload, [88](#)
 - main, [87](#)
 - maxPrimitiveArraySize, [88](#)
 - PFPrimitive_Ping_Pong_Primitive_Asynchronous, [87](#)
 - PFPrimitive_Ping_Pong_Primitive_Synchronous, [86](#)
 - serviceld, [87](#)
 - testAddress, [88](#)
 - usecPerSecond, [88](#)
- PFPrimitive_Ping_Pong_Primitive_Asynchronous
 - PFPrimitive.cpp, [87](#)
- PFPrimitive_Ping_Pong_Primitive_Synchronous
 - PFPrimitive.cpp, [86](#)
- ProxyPtr
 - CMAAttributeSubscription.cpp, [49](#)
- RTBuildProxiesAndStubs.cpp
 - applicationNameClient, [92](#)
 - applicationNameService, [92](#)
 - domain, [91](#)
 - main, [91](#)
 - RTBuildProxiesAndStubs_BuildProxiesAndStubsTwoTimes, [89](#)
 - RTBuildProxiesAndStubs_BuildProxySubscribeToProxyStatusEventB, [91](#)
 - RTBuildProxiesAndStubs_BuildProxyTwoTimesWithReassigningAndS, [89](#)
 - RTBuildProxiesAndStubs_LoadedRuntimeCanBuildProxiesAndStubs, [89](#)
 - RTBuildProxiesAndStubs_WaitForProxyDestruction, [90](#)
 - RTBuildProxiesAndStubs_WaitForProxyDestructionCreatedInThread, [90](#)

- RTBuildProxiesAndStubs_WaitForProxyDestructionInTwoThreads
 - 90
 - tasync, 92
 - testAddress, 91
- RTBuildProxiesAndStubs_BuildProxiesAndStubsTwoTimes
 - RTBuildProxiesAndStubs.cpp, 89
- RTBuildProxiesAndStubs_BuildProxySubscribeToProxyStatusEventAndShutdown
 - RTBuildProxiesAndStubs.cpp, 91
- RTBuildProxiesAndStubs_BuildProxyTwoTimesWithReassigningAndStubs
 - RTBuildProxiesAndStubs.cpp, 89
- RTBuildProxiesAndStubs_LoadedRuntimeCanBuildProxiesAndStubs
 - RTBuildProxiesAndStubs.cpp, 89
- RTBuildProxiesAndStubs_WaitForProxyDestruction
 - RTBuildProxiesAndStubs.cpp, 90
- RTBuildProxiesAndStubs_WaitForProxyDestructionCreatedInThread
 - RTBuildProxiesAndStubs.cpp, 90
- RTBuildProxiesAndStubs_WaitForProxyDestructionInTwoThreads
 - RTBuildProxiesAndStubs.cpp, 90
- RTLLoadingRuntime.cpp
 - main, 92
 - RTLLoadingRuntime_LoadsDefaultRuntime, 92
- RTLLoadingRuntime_LoadsDefaultRuntime
 - RTLLoadingRuntime.cpp, 92
- serviceld
 - AFExtended.cpp, 28
 - AFSelective.cpp, 44
 - CMAAttributes.cpp, 47
 - CMAAttributeSubscription.cpp, 57
 - CMBlockingCalls.cpp, 61
 - CMBroadcasts.cpp, 64
 - CMMethodCalls.cpp, 70
 - PFCOMPLEX.cpp, 85
 - PFPPrimitive.cpp, 87
 - StabilitySP.cpp, 96
- StabilitySP.cpp
 - clientId, 96
 - COMMONAPI_CONFIG_SUFFIX, 96
 - domain, 96
 - main, 95
 - MAXMETHODCALLS, 97
 - MAXREGCOUNT, 97
 - MAXREGLOOPS, 97
 - MAXSERVERCOUNT, 96
 - MAXSUBSCRIPTIONSETS, 97
 - MAXTHREADCOUNT, 96
 - MESSAGESIZE, 97
 - serviceld, 96
 - StabilitySP_MultipleAttributeGetAsyncs, 94
 - StabilitySP_MultipleAttributeGets, 94
 - StabilitySP_MultipleAttributeSetAsyncs, 95
 - StabilitySP_MultipleAttributeSets, 94
 - StabilitySP_MultipleAttributeSubscriptions, 95
 - StabilitySP_MultipleMethodCalls, 93
 - StabilitySP_RepeatedRegistrations, 93
 - testAddress, 96
- StabilitySP_MultipleAttributeGetAsyncs
 - StabilitySP.cpp, 94
- StabilitySP_MultipleAttributeGets
 - StabilitySP.cpp, 94
- StabilitySP_MultipleAttributeSetAsyncs
 - StabilitySP.cpp, 95
- StabilitySP_MultipleAttributeSets
 - StabilitySP.cpp, 94
- StabilitySP_MultipleAttributeSubscriptions
 - StabilitySP.cpp, 95
- StabilitySP_MultipleMethodCalls
 - StabilitySP.cpp, 93
- StabilitySP_RepeatedRegistrations
 - StabilitySP.cpp, 93
- StabilitySP.cpp, 94
 - StabilitySP_MultipleAttributeSetAsyncs
 - StabilitySP.cpp, 95
 - StabilitySP_MultipleAttributeSets
 - StabilitySP_MultipleAttributeSubscriptions
 - StabilitySP_MultipleMethodCalls
 - StabilitySP_RepeatedRegistrations
 - StabilitySP.cpp, 93
 - tasync
 - AFExtended.cpp, 29
 - AFPOLYmorph.cpp, 41
 - AFSelective.cpp, 45
 - CMAAttributes.cpp, 48
 - CMBlockingCalls.cpp, 61
 - CMBroadcasts.cpp, 65
 - CMMethodCalls.cpp, 71
 - DTAdvanced.cpp, 74
 - DTDerived.cpp, 81
 - DTPRIMITIVE.cpp, 83
 - RTBuildProxiesAndStubs.cpp, 92
 - THMainLoopIndependence.cpp, 100
 - THMainLoopIntegration.cpp, 103
- testAddress
 - AFPOLYmorph.cpp, 41
 - AFSelective.cpp, 45
 - CMAAttributes.cpp, 48
 - CMAAttributeSubscription.cpp, 58
 - CMBlockingCalls.cpp, 61
 - CMBroadcasts.cpp, 65
 - CMMethodCalls.cpp, 70
 - DTAdvanced.cpp, 74
 - DTCombined.cpp, 76
 - DTDeployment.cpp, 78
 - DTDerived.cpp, 80
 - DTPRIMITIVE.cpp, 83
 - PFCOMPLEX.cpp, 86
 - PFPPrimitive.cpp, 88
 - RTBuildProxiesAndStubs.cpp, 91
 - StabilitySP.cpp, 96
- testAddress2
 - CMBlockingCalls.cpp, 61
 - CMMethodCalls.cpp, 70
- testAddressBase
 - AFExtended.cpp, 28
- testAddressOnce
 - AFExtended.cpp, 29
- testAddressTwice
 - AFExtended.cpp, 29
- testSubscription
 - CMAAttributeSubscription.cpp, 49
- thirdPartyServiceld
 - THMainLoopIndependence.cpp, 99
- THMainLoopIndependence.cpp
 - domain, 99
 - instance6, 99

instance7, [99](#)
 instance8, [99](#)
 main, [98](#)
 mainloopName1, [99](#)
 mainloopName2, [99](#)
 tasync, [100](#)
 thirdPartyServiceId, [99](#)
 THMainLoopIndependence_ProxyReceivesAnswerOnlyIfStubMainLoopRuns,
 [98](#)
 THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersAsync,
 [98](#)
 THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersSync,
 [98](#)
 THMainLoopIndependence_ProxyReceivesAnswerOnlyIfStubMainLoopRuns
 THMainLoopIndependence.cpp, [98](#)
 THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersAsync
 THMainLoopIndependence.cpp, [98](#)
 THMainLoopIndependence_ProxyReceivesJustHisOwnAnswersSync
 THMainLoopIndependence.cpp, [98](#)
 THMainLoopIntegration.cpp
 connection_client, [102](#)
 connection_service, [102](#)
 domain, [102](#)
 instance, [102](#)
 main, [102](#)
 tasync, [103](#)
 THMainLoopIntegration_AsynchronousMethodCallsReceiveNotAvailable,
 [101](#)
 THMainLoopIntegration_CreateProxyToManagerInSameProcess,
 [102](#)
 THMainLoopIntegration_SelectiveErrorHandlerWithMainLoop,
 [101](#)
 THMainLoopIntegration_VerifyCommunicationWithMainLoop,
 [100](#)
 THMainLoopIntegration_VerifySyncCallMessageHandlingOrder,
 [101](#)
 THMainLoopIntegration_VerifyTransportReading,
 [100](#)
 THMainLoopIntegration_AsynchronousMethodCallsReceiveNotAvailable
 THMainLoopIntegration.cpp, [101](#)
 THMainLoopIntegration_CreateProxyToManagerInSameProcess
 THMainLoopIntegration.cpp, [102](#)
 THMainLoopIntegration_SelectiveErrorHandlerWithMainLoop
 THMainLoopIntegration.cpp, [101](#)
 THMainLoopIntegration_VerifyCommunicationWithMainLoop
 THMainLoopIntegration.cpp, [100](#)
 THMainLoopIntegration_VerifySyncCallMessageHandlingOrder
 THMainLoopIntegration.cpp, [101](#)
 THMainLoopIntegration_VerifyTransportReading
 THMainLoopIntegration.cpp, [100](#)
 THMainLoopTwoThreads.cpp
 domain, [104](#)
 instance, [104](#)
 main, [103](#)
 THMainLoopTwoThreads_ProxyGetsAvailableStatus,
 [103](#)
 THMainLoopTwoThreads_ProxyGetsFunctionResponse,
 [103](#)
 THMainLoopTwoThreads_ProxyGetsAvailableStatus
 THMainLoopTwoThreads.cpp, [103](#)
 THMainLoopTwoThreads_ProxyGetsFunctionResponse
 THMainLoopTwoThreads.cpp, [103](#)
 timeout
 CMBlockingCalls.cpp, [61](#)
 CMMethodCalls.cpp, [71](#)
 usecPerSecond
 PFCComplex.cpp, [85](#)
 PFPrimitive.cpp, [88](#)
 wf
 CMAAttributeSubscription.cpp, [58](#)
 CMBlockingCalls.cpp, [62](#)
 CMBroadcasts.cpp, [65](#)
 CMMethodCalls.cpp, [71](#)
 CMAAttributeSubscription.cpp, [58](#)