

VICI
0.11.815

Generated by Doxygen 1.8.5

Sun Feb 3 2019 16:15:57

Contents

1	Main Page	1
2	Todo List	3
3	Namespace Index	5
3.1	Namespace List	5
4	Hierarchical Index	7
4.1	Class Hierarchy	7
5	Class Index	11
5.1	Class List	11
6	File Index	19
6.1	File List	19
7	Namespace Documentation	21
7.1	VICI Namespace Reference	21
7.1.1	Detailed Description	24
7.1.2	Typedef Documentation	24
7.1.2.1	AsyncTestEventFn	24
7.1.3	Enumeration Type Documentation	24
7.1.3.1	Severity	24
7.1.4	Function Documentation	24
7.1.4.1	clean	24
7.1.4.2	defaultAsyncTestEvent	24
7.1.4.3	expandMacros	25
7.1.4.4	split	25
7.1.4.5	trim	25
7.2	VICI::Admin Namespace Reference	25
7.2.1	Detailed Description	26
7.3	VICI::Canvas Namespace Reference	26
7.3.1	Detailed Description	26
7.4	VICI::cdi Namespace Reference	26

7.4.1	Detailed Description	27
7.5	VICI::cfi Namespace Reference	27
7.5.1	Detailed Description	29
7.6	VICI::Cmnd Namespace Reference	29
7.6.1	Detailed Description	30
7.7	VICI::Cron Namespace Reference	30
7.7.1	Detailed Description	30
7.8	VICI::EBNF Namespace Reference	30
7.8.1	Detailed Description	31
7.9	VICI::Ed Namespace Reference	31
7.9.1	Detailed Description	31
7.10	VICI::gth Namespace Reference	31
7.10.1	Detailed Description	33
7.10.2	Enumeration Type Documentation	33
7.10.2.1	WidgetType	33
7.10.3	Function Documentation	34
7.10.3.1	RegnFn	34
7.11	VICI::Inst Namespace Reference	34
7.11.1	Detailed Description	34
7.12	VICI::Interp Namespace Reference	34
7.12.1	Detailed Description	35
7.13	VICI::Search Namespace Reference	35
7.13.1	Detailed Description	35
7.14	VICI::Sec Namespace Reference	35
7.14.1	Detailed Description	36
7.15	VICI::stub Namespace Reference	36
7.15.1	Detailed Description	37
7.16	VICI::Symbol Namespace Reference	37
7.16.1	Detailed Description	38
7.16.2	Enumeration Type Documentation	38
7.16.2.1	Style	38
7.17	VICI::Syntax Namespace Reference	39
7.17.1	Detailed Description	39
8	Class Documentation	41
8.1	VICI::AboutDialog Class Reference	41
8.1.1	Detailed Description	41
8.2	VICI::cfi::AbstractChildProcess Class Reference	41
8.2.1	Detailed Description	42
8.3	VICI::cdi::AbstractScenario Class Reference	42

8.3.1	Detailed Description	43
8.3.2	Constructor & Destructor Documentation	43
8.3.2.1	AbstractScenario	43
8.3.3	Member Function Documentation	43
8.3.3.1	willRunTests	43
8.4	VICI::cdi::AbstractTest Class Reference	43
8.4.1	Detailed Description	44
8.5	VICI::cdi::AbstractTestCase Class Reference	44
8.5.1	Detailed Description	45
8.5.2	Constructor & Destructor Documentation	45
8.5.2.1	AbstractTestCase	45
8.5.2.2	~AbstractTestCase	45
8.5.3	Member Function Documentation	45
8.5.3.1	operator()	45
8.5.3.2	runTest	45
8.5.3.3	test	45
8.6	VICI::gth::Adaptor Class Reference	46
8.6.1	Detailed Description	47
8.6.2	Member Function Documentation	47
8.6.2.1	action	47
8.6.2.2	getType	47
8.7	VICI::gth::AdaptorST< wt > Class Template Reference	47
8.7.1	Detailed Description	48
8.8	VICI::gth::AdaptorT< wt, T > Class Template Reference	48
8.8.1	Detailed Description	49
8.8.2	Constructor & Destructor Documentation	49
8.8.2.1	AdaptorT	49
8.8.3	Member Function Documentation	49
8.8.3.1	getType	49
8.9	VICI::stub::AnEvent< T > Class Template Reference	49
8.9.1	Detailed Description	50
8.9.2	Constructor & Destructor Documentation	50
8.9.2.1	AnEvent	50
8.10	VICI::cdi::AsyncTestCase Class Reference	50
8.10.1	Detailed Description	51
8.10.2	Constructor & Destructor Documentation	51
8.10.2.1	AsyncTestCase	51
8.10.3	Member Function Documentation	51
8.10.3.1	handleEvent	51
8.10.3.2	initTest	51

8.10.3.3	timedOut	51
8.11	VICI::cdi::AsyncTestCaseT< T > Class Template Reference	52
8.11.1	Detailed Description	52
8.11.2	Constructor & Destructor Documentation	52
8.11.2.1	AsyncTestCaseT	52
8.11.3	Member Function Documentation	53
8.11.3.1	install	53
8.12	VICI::cfi::AutoRunPlugIn Class Reference	54
8.12.1	Detailed Description	54
8.13	VICI::cdi::CallTrace Class Reference	54
8.13.1	Detailed Description	55
8.13.2	Constructor & Destructor Documentation	55
8.13.2.1	CallTrace	55
8.14	VICI::Canvas::Canvas Class Reference	55
8.14.1	Detailed Description	56
8.14.2	Member Function Documentation	56
8.14.2.1	load	56
8.14.2.2	save	56
8.14.2.3	selection	56
8.14.2.4	setCommand	56
8.14.2.5	setExecution	57
8.14.2.6	symbolAttr	58
8.14.2.7	textAttr	58
8.15	VICI::Canvas::CanvasClient Class Reference	58
8.15.1	Detailed Description	59
8.15.2	Member Function Documentation	59
8.15.2.1	newSymbol	59
8.16	VICI::Canvas::CanvasFactory Class Reference	59
8.16.1	Detailed Description	60
8.17	VICI::CanvasScene Class Reference	60
8.17.1	Detailed Description	60
8.17.2	Constructor & Destructor Documentation	61
8.17.2.1	CanvasScene	61
8.18	VICI::stub::CanvasStub Class Reference	62
8.18.1	Detailed Description	63
8.18.2	Constructor & Destructor Documentation	63
8.18.2.1	CanvasStub	63
8.18.3	Member Function Documentation	63
8.18.3.1	breakReached	63
8.18.3.2	dataReady	63

8.18.3.3	load	63
8.18.3.4	reportError	64
8.18.3.5	save	64
8.18.3.6	selection	64
8.18.3.7	setCommand	64
8.18.3.8	setCursor	64
8.18.3.9	setExecution	65
8.18.3.10	setFile	66
8.18.3.11	setValue	66
8.18.3.12	symbolAttr	66
8.18.3.13	textAttr	66
8.19	VICI::stub::CanvasStubFactory Class Reference	67
8.19.1	Detailed Description	67
8.20	VICI::cfi::ChildProcess Class Reference	67
8.20.1	Detailed Description	68
8.20.2	Constructor & Destructor Documentation	68
8.20.2.1	ChildProcess	68
8.20.3	Member Function Documentation	68
8.20.3.1	finished	68
8.20.3.2	setArgs	68
8.21	VICI::cfi::ChildProcessMgr Class Reference	69
8.21.1	Detailed Description	69
8.21.2	Member Function Documentation	69
8.21.2.1	deregisterChild	69
8.21.2.2	numberOfChildren	70
8.21.2.3	numberOfLiveChildren	70
8.21.2.4	registerChild	70
8.22	VICI::Cmnd::Command Class Reference	70
8.22.1	Detailed Description	71
8.22.2	Member Function Documentation	71
8.22.2.1	selection	71
8.22.2.2	setCommand	71
8.23	VICI::Cmnd::CommandClient Class Reference	71
8.23.1	Detailed Description	72
8.23.2	Member Function Documentation	72
8.23.2.1	cmndError	72
8.23.2.2	optionsAndParameters	72
8.24	VICI::Cmnd::CommandFactory Class Reference	72
8.24.1	Detailed Description	73
8.25	VICI::stub::CommandStub Class Reference	73

8.25.1	Detailed Description	74
8.25.2	Constructor & Destructor Documentation	74
8.25.2.1	CommandStub	74
8.25.3	Member Function Documentation	74
8.25.3.1	selection	74
8.25.3.2	setCommand	74
8.26	VICI::stub::CommandStubFactory Class Reference	74
8.26.1	Detailed Description	75
8.27	VICI::Cron::Cron Class Reference	75
8.27.1	Detailed Description	75
8.27.2	Member Function Documentation	76
8.27.2.1	setCurrentFile	76
8.28	VICI::Cron::CronFactory Class Reference	76
8.28.1	Detailed Description	76
8.28.2	Member Function Documentation	76
8.28.2.1	makeCron	76
8.29	VICI::stub::CronStub Class Reference	77
8.29.1	Detailed Description	77
8.29.2	Constructor & Destructor Documentation	77
8.29.2.1	CronStub	77
8.29.3	Member Function Documentation	78
8.29.3.1	setCurrentFile	78
8.30	VICI::stub::CronStubFactory Class Reference	79
8.30.1	Detailed Description	79
8.30.2	Member Function Documentation	79
8.30.2.1	makeCron	79
8.31	VICI::cdi::DefaultScenario Class Reference	80
8.31.1	Detailed Description	80
8.32	VICI::cdi::DefaultTest Class Reference	80
8.32.1	Detailed Description	81
8.33	VICI::gth::DefaultTestCase Class Reference	81
8.33.1	Detailed Description	81
8.33.2	Member Function Documentation	82
8.33.2.1	runTest	82
8.34	VICI::cfi::Discoverable Class Reference	82
8.34.1	Detailed Description	82
8.35	VICI::cfi::DiscoverPointer Struct Reference	82
8.35.1	Detailed Description	83
8.35.2	Constructor & Destructor Documentation	83
8.35.2.1	DiscoverPointer	83

8.36	VICI::cfi::DiscoveryMgr Class Reference	83
8.36.1	Detailed Description	83
8.36.2	Member Function Documentation	84
8.36.2.1	fetch	84
8.36.2.2	save	85
8.37	VICI::stub::Dispatcher Class Reference	85
8.37.1	Detailed Description	85
8.37.2	Member Function Documentation	86
8.37.2.1	enableTracing	86
8.37.2.2	getTrace	86
8.37.2.3	instance	86
8.37.2.4	registerEvent	86
8.37.2.5	sendEvent	86
8.37.2.6	trace	86
8.38	VICI::EBNF::EBNF Class Reference	87
8.38.1	Detailed Description	87
8.38.2	Member Function Documentation	87
8.38.2.1	getError	87
8.38.2.2	parse	87
8.38.2.3	validate	88
8.39	VICI::EBNF::EBNF_Factory Class Reference	88
8.39.1	Detailed Description	88
8.39.2	Member Function Documentation	88
8.39.2.1	makeEBNF	88
8.40	VICI::stub::EBNF_Stub Class Reference	89
8.40.1	Detailed Description	89
8.40.2	Member Function Documentation	89
8.40.2.1	getError	89
8.40.2.2	parse	89
8.40.2.3	validate	90
8.41	VICI::stub::EBNF_Stub_Factory Class Reference	90
8.41.1	Detailed Description	90
8.41.2	Member Function Documentation	90
8.41.2.1	makeEBNF	90
8.42	VICI::EbnfNode Class Reference	91
8.42.1	Detailed Description	92
8.42.2	Member Enumeration Documentation	92
8.42.2.1	NodeType	92
8.42.3	Member Function Documentation	92
8.42.3.1	typeOfNode	92

8.42.3.2	typeOfNode	92
8.43	VICI::EbnfTree Class Reference	93
8.43.1	Detailed Description	93
8.43.2	Member Function Documentation	93
8.43.2.1	exportXml	93
8.43.2.2	importXml	93
8.44	VICI::EbnfXml Class Reference	94
8.44.1	Detailed Description	94
8.44.2	Member Function Documentation	94
8.44.2.1	exportTree	94
8.44.2.2	importTree	95
8.45	VICI::stub::Event Class Reference	96
8.45.1	Detailed Description	96
8.46	VICI::Factory Class Reference	96
8.46.1	Detailed Description	97
8.47	VICI::FactoryFactory Class Reference	97
8.47.1	Detailed Description	98
8.47.2	Member Function Documentation	98
8.47.2.1	getFactory	98
8.47.2.2	registerFactory	98
8.48	VICI::cfi::FD Class Reference	98
8.48.1	Detailed Description	99
8.48.2	Constructor & Destructor Documentation	99
8.48.2.1	FD	99
8.48.2.2	FD	99
8.48.3	Member Function Documentation	99
8.48.3.1	report	99
8.49	VICI::cfi::fdinbuf Class Reference	99
8.49.1	Detailed Description	100
8.49.2	Constructor & Destructor Documentation	101
8.49.2.1	fdinbuf	101
8.49.3	Member Data Documentation	102
8.49.3.1	mBuffer	102
8.50	VICI::cfi::fdistream Class Reference	102
8.50.1	Detailed Description	103
8.50.2	Constructor & Destructor Documentation	103
8.50.2.1	fdistream	103
8.51	VICI::cfi::fdostream Class Reference	103
8.51.1	Detailed Description	104
8.51.2	Constructor & Destructor Documentation	104

8.51.2.1	fdostream	104
8.52	VICI::cfi::fdoutbuf Class Reference	104
8.52.1	Detailed Description	105
8.52.2	Constructor & Destructor Documentation	105
8.52.2.1	fdoutbuf	105
8.52.3	Member Function Documentation	106
8.52.3.1	close	106
8.52.3.2	overflow	106
8.53	VICI::cfi::FileLogger Class Reference	106
8.53.1	Detailed Description	107
8.53.2	Constructor & Destructor Documentation	107
8.53.2.1	FileLogger	107
8.53.3	Member Function Documentation	107
8.53.3.1	log	107
8.53.3.2	log	107
8.54	VICI::cfi::FormattingLogger Class Reference	108
8.54.1	Detailed Description	108
8.54.2	Member Function Documentation	108
8.54.2.1	format	108
8.54.2.2	timeStamp	109
8.55	VICI::gth::GTHTest Class Reference	109
8.55.1	Detailed Description	110
8.56	VICI::gth::GTHTestCase Class Reference	110
8.56.1	Detailed Description	111
8.57	VICI::GTHWindowWidget Class Reference	111
8.57.1	Detailed Description	112
8.58	VICI::Inst::Installer Class Reference	112
8.58.1	Detailed Description	112
8.58.2	Member Function Documentation	112
8.58.2.1	setCurrentFile	112
8.59	VICI::Inst::InstallerFactory Class Reference	113
8.59.1	Detailed Description	113
8.59.2	Member Function Documentation	113
8.59.2.1	makeInstaller	113
8.60	VICI::stub::InstallerStub Class Reference	114
8.60.1	Detailed Description	114
8.60.2	Constructor & Destructor Documentation	114
8.60.2.1	InstallerStub	114
8.60.3	Member Function Documentation	114
8.60.3.1	setCurrentFile	114

8.61	VICI::stub::InstallerStubFactory Class Reference	115
8.61.1	Detailed Description	115
8.61.2	Member Function Documentation	115
8.61.2.1	makeInstaller	115
8.62	VICI::Interp::Interpreter Class Reference	115
8.62.1	Detailed Description	117
8.62.2	Member Function Documentation	117
8.62.2.1	dataAck	117
8.62.2.2	debugMode	117
8.62.2.3	loadSnapshot	117
8.62.2.4	openDisplay	117
8.62.2.5	run	118
8.62.2.6	saveSnapshot	118
8.62.2.7	setBreak	118
8.62.2.8	setInterval	118
8.62.2.9	setPosn	118
8.62.2.10	setValue	119
8.62.2.11	step	119
8.63	VICI::Interp::InterpreterClient Class Reference	119
8.63.1	Detailed Description	120
8.63.2	Member Function Documentation	120
8.63.2.1	breakReached	120
8.63.2.2	dataReady	120
8.63.2.3	reportError	120
8.63.2.4	setCursor	120
8.63.2.5	setFile	121
8.63.2.6	setValue	121
8.64	VICI::Interp::InterpreterFactory Class Reference	121
8.64.1	Detailed Description	122
8.64.2	Member Function Documentation	122
8.64.2.1	makeInterpreter	122
8.65	VICI::stub::InterpreterStub Class Reference	122
8.65.1	Detailed Description	123
8.65.2	Constructor & Destructor Documentation	123
8.65.2.1	InterpreterStub	123
8.65.3	Member Function Documentation	123
8.65.3.1	dataAck	123
8.65.3.2	debugMode	124
8.65.3.3	loadSnapshot	125
8.65.3.4	openDisplay	125

8.65.3.5	run	125
8.65.3.6	saveSnapshot	125
8.65.3.7	setBreak	125
8.65.3.8	setInterval	126
8.65.3.9	setPosn	126
8.65.3.10	setValue	126
8.65.3.11	step	126
8.66	VICI::stub::InterpreterStubFactory Class Reference	126
8.66.1	Detailed Description	127
8.66.2	Member Function Documentation	127
8.66.2.1	makeInterpreter	127
8.67	VICI::ItemDelegate Class Reference	127
8.67.1	Detailed Description	128
8.68	logbuff Class Reference	128
8.68.1	Detailed Description	129
8.68.2	Member Function Documentation	129
8.68.2.1	flushBuffer	129
8.68.2.2	overflow	129
8.68.2.3	sync	129
8.69	VICI::cfi::logger Class Reference	130
8.69.1	Detailed Description	130
8.69.2	Member Function Documentation	130
8.69.2.1	log	130
8.69.2.2	log	131
8.70	VICI::cfi::logstream Class Reference	131
8.70.1	Detailed Description	132
8.70.2	Constructor & Destructor Documentation	132
8.70.2.1	logstream	132
8.70.3	Member Function Documentation	132
8.70.3.1	instance	132
8.70.3.2	instance	132
8.70.3.3	severity	133
8.71	VICI::gth::LuaScript Class Reference	133
8.71.1	Detailed Description	133
8.72	VICI::Metrics Class Reference	133
8.72.1	Detailed Description	134
8.73	VICI::gth::MouseEventReporter Class Reference	134
8.73.1	Detailed Description	134
8.74	VICI::cdi::NullTrace Class Reference	135
8.74.1	Detailed Description	135

8.75	VICI::EBNF::ParseTree Class Reference	135
8.75.1	Detailed Description	135
8.76	VICI::Path Class Reference	136
8.76.1	Detailed Description	137
8.76.2	Constructor & Destructor Documentation	137
8.76.2.1	Path	137
8.76.3	Member Function Documentation	137
8.76.3.1	absolute	137
8.76.3.2	append	137
8.76.3.3	appendExt	137
8.76.3.4	base	138
8.76.3.5	defined	138
8.76.3.6	dir	138
8.76.3.7	ext	138
8.76.3.8	isChildOf	138
8.76.3.9	name	138
8.76.3.10	noExt	138
8.76.3.11	split	138
8.76.3.12	up	139
8.77	VICI::cfi::PlainFileLogger Class Reference	139
8.77.1	Detailed Description	139
8.77.2	Constructor & Destructor Documentation	139
8.77.2.1	PlainFileLogger	139
8.77.3	Member Function Documentation	140
8.77.3.1	log	140
8.77.3.2	log	140
8.78	VICI::cfi::PlugIn Class Reference	140
8.78.1	Detailed Description	141
8.79	VICI::cfi::PlugInDescriptor Struct Reference	141
8.79.1	Detailed Description	142
8.80	VICI::cfi::PlugInDetails Struct Reference	142
8.80.1	Detailed Description	142
8.81	VICI::cfi::PlugInFactory Class Reference	142
8.81.1	Detailed Description	143
8.81.2	Member Function Documentation	143
8.81.2.1	make	143
8.82	VICI::cfi::PlugInFactoryT< F, P > Class Template Reference	143
8.82.1	Detailed Description	144
8.82.2	Member Function Documentation	144
8.82.2.1	make	144

8.83	VICI::cfi::PlugInFamilyFactoryT< F > Class Template Reference	144
8.83.1	Detailed Description	145
8.83.2	Member Function Documentation	145
8.83.2.1	make	145
8.84	VICI::cfi::PlugInLib Class Reference	145
8.84.1	Detailed Description	146
8.84.2	Member Enumeration Documentation	146
8.84.2.1	Mode	146
8.84.3	Constructor & Destructor Documentation	146
8.84.3.1	PlugInLib	146
8.84.4	Member Function Documentation	146
8.84.4.1	getMode	146
8.84.4.2	getPath	146
8.84.4.3	inUse	147
8.84.4.4	loaded	147
8.84.4.5	setMode	147
8.85	VICI::cfi::PlugInMgr Class Reference	147
8.85.1	Detailed Description	148
8.85.2	Member Function Documentation	148
8.85.2.1	load	148
8.86	VICI::cfi::ProcessOwner Class Reference	148
8.86.1	Detailed Description	148
8.86.2	Member Function Documentation	148
8.86.2.1	processTerminated	148
8.87	VICI::cdi::scenario_exception Class Reference	149
8.87.1	Detailed Description	149
8.88	VICI::cdi::ScenarioFactory Class Reference	149
8.88.1	Detailed Description	150
8.89	VICI::cdi::ScenarioFT< T > Class Template Reference	150
8.89.1	Detailed Description	150
8.89.2	Member Function Documentation	150
8.89.2.1	make	150
8.90	VICI::cdi::ScenarioResults Struct Reference	151
8.90.1	Detailed Description	151
8.91	VICI::cdi::ScenarioT< T > Class Template Reference	151
8.91.1	Detailed Description	152
8.91.2	Constructor & Destructor Documentation	152
8.91.2.1	ScenarioT	152
8.91.3	Member Function Documentation	152
8.91.3.1	install	152

8.92	VICI::Scene Class Reference	152
8.92.1	Detailed Description	153
8.93	VICI::gth::ScriptXML Class Reference	153
8.93.1	Detailed Description	154
8.94	VICI::Search::Search Class Reference	154
8.94.1	Detailed Description	154
8.95	VICI::Search::SearchClient Class Reference	154
8.95.1	Detailed Description	155
8.95.2	Member Function Documentation	155
8.95.2.1	selectedCommand	155
8.96	VICI::Search::SearchFactory Class Reference	155
8.96.1	Detailed Description	156
8.96.2	Member Function Documentation	156
8.96.2.1	makeSearch	156
8.97	VICI::stub::SearchStub Class Reference	156
8.97.1	Detailed Description	157
8.97.2	Constructor & Destructor Documentation	157
8.97.2.1	SearchStub	157
8.98	VICI::stub::SearchStubFactory Class Reference	157
8.98.1	Detailed Description	157
8.98.2	Member Function Documentation	157
8.98.2.1	makeSearch	157
8.99	VICI::Sec::Secure Class Reference	158
8.99.1	Detailed Description	158
8.99.2	Member Function Documentation	158
8.99.2.1	addSignature	158
8.99.2.2	verifySignature	159
8.100	VICI::Sec::SecureFactory Class Reference	159
8.100.1	Detailed Description	159
8.100.2	Member Function Documentation	159
8.100.2.1	makeSecure	160
8.101	VICI::stub::SecureStub Class Reference	160
8.101.1	Detailed Description	160
8.101.2	Member Function Documentation	160
8.101.2.1	addSignature	160
8.101.2.2	verifySignature	160
8.102	VICI::stub::SecureStubFactory Class Reference	161
8.102.1	Detailed Description	161
8.102.2	Member Function Documentation	161
8.102.2.1	makeSecure	161

8.103VICI::cfi::Semaphore Class Reference	162
8.103.1 Detailed Description	162
8.103.2 Constructor & Destructor Documentation	162
8.103.2.1 Semaphore	162
8.104VICI::cfi::SemaphoreLock Class Reference	162
8.104.1 Detailed Description	163
8.104.2 Constructor & Destructor Documentation	163
8.104.2.1 SemaphoreLock	163
8.105VICI::SignalToQtSignal Class Reference	163
8.105.1 Detailed Description	163
8.106VICI::cfi::StdLogger Class Reference	164
8.106.1 Detailed Description	164
8.106.2 Constructor & Destructor Documentation	164
8.106.2.1 StdLogger	164
8.106.3 Member Function Documentation	164
8.106.3.1 log	164
8.106.3.2 log	165
8.107VICI::Symbol::Symbol Class Reference	165
8.107.1 Detailed Description	166
8.107.2 Member Function Documentation	166
8.107.2.1 attachCommand	166
8.107.2.2 clone	166
8.107.2.3 draw	166
8.107.2.4 getStyle	167
8.107.2.5 isCommand	167
8.107.2.6 setAttributes	167
8.107.2.7 setHighlight	167
8.107.2.8 setNodeId	167
8.108VICI::Symbol::SymbolAttributes Class Reference	167
8.108.1 Detailed Description	168
8.109VICI::Symbol::SymbolClient Class Reference	168
8.109.1 Detailed Description	169
8.109.2 Member Function Documentation	169
8.109.2.1 selection	169
8.109.2.2 symbolAttr	169
8.109.2.3 textAttr	169
8.110VICI::Symbol::SymbolFactory Class Reference	169
8.110.1 Detailed Description	170
8.110.2 Member Function Documentation	170
8.110.2.1 makeSymbolMgr	170

8.111VICI::Symbol::SymbolMgr Class Reference	170
8.111.1 Detailed Description	171
8.111.2 Member Function Documentation	171
8.111.2.1 addClient	171
8.111.2.2 getDefaultAttr	171
8.111.2.3 getDefaultTextAttr	171
8.111.2.4 getSymbol	171
8.112VICI::stub::SymbolMgrStub Class Reference	172
8.112.1 Detailed Description	172
8.112.2 Constructor & Destructor Documentation	172
8.112.2.1 SymbolMgrStub	172
8.112.3 Member Function Documentation	173
8.112.3.1 addClient	173
8.112.3.2 getDefaultAttr	173
8.112.3.3 getDefaultTextAttr	173
8.112.3.4 getSymbol	173
8.113VICI::Symbol::SymbolOwner Class Reference	174
8.113.1 Detailed Description	174
8.113.2 Member Function Documentation	174
8.113.2.1 dragged	174
8.113.2.2 opened	174
8.114VICI::stub::SymbolStub Class Reference	174
8.114.1 Detailed Description	175
8.114.2 Constructor & Destructor Documentation	175
8.114.2.1 SymbolStub	175
8.114.3 Member Function Documentation	175
8.114.3.1 attachCommand	175
8.114.3.2 clone	175
8.114.3.3 draw	176
8.114.3.4 getStyle	176
8.114.3.5 isCommand	176
8.114.3.6 setAttributes	176
8.114.3.7 setHighlight	176
8.114.3.8 setNodeIId	177
8.115VICI::stub::SymbolStubFactory Class Reference	177
8.115.1 Detailed Description	177
8.115.2 Member Function Documentation	177
8.115.2.1 makeSymbolMgr	177
8.116VICI::Syntax::Syntax Class Reference	178
8.116.1 Detailed Description	178

8.116.2 Member Function Documentation	178
8.116.2.1 show	178
8.117VICI::Syntax::SyntaxFactory Class Reference	179
8.117.1 Detailed Description	179
8.117.2 Member Function Documentation	179
8.117.2.1 makeSyntax	179
8.118VICI::stub::SyntaxStub Class Reference	179
8.118.1 Detailed Description	180
8.118.2 Constructor & Destructor Documentation	180
8.118.2.1 SyntaxStub	180
8.118.3 Member Function Documentation	180
8.118.3.1 show	180
8.119VICI::stub::SyntaxStubFactory Class Reference	181
8.119.1 Detailed Description	181
8.119.2 Member Function Documentation	181
8.119.2.1 makeSyntax	181
8.120VICI::cfi::SystemLogger Class Reference	181
8.120.1 Detailed Description	182
8.120.2 Constructor & Destructor Documentation	182
8.120.2.1 SystemLogger	182
8.120.3 Member Function Documentation	182
8.120.3.1 log	182
8.120.3.2 log	182
8.121VICI::cdi::test_exception Class Reference	183
8.121.1 Detailed Description	183
8.122VICI::gth::TestAction Struct Reference	183
8.122.1 Detailed Description	184
8.123VICI::cdi::TestCaseFactory Class Reference	184
8.123.1 Detailed Description	184
8.123.2 Member Function Documentation	185
8.123.2.1 make	185
8.124VICI::cdi::TestCaseFT< T > Class Template Reference	185
8.124.1 Detailed Description	185
8.124.2 Member Function Documentation	185
8.124.2.1 make	185
8.125VICI::cdi::TestCaseT< T > Class Template Reference	186
8.125.1 Detailed Description	186
8.125.2 Constructor & Destructor Documentation	186
8.125.2.1 TestCaseT	186
8.125.3 Member Function Documentation	187

8.125.3.1 install	187
8.126VICI::cdi::Tester Class Reference	187
8.126.1 Detailed Description	188
8.126.2 Member Function Documentation	188
8.126.2.1 addScenario	188
8.126.2.2 addTestCase	189
8.126.2.3 configure	189
8.126.2.4 getScenario	189
8.126.2.5 getTest	189
8.126.2.6 getTestName	189
8.126.2.7 instance	189
8.126.2.8 log	190
8.126.2.9 setTest	190
8.126.2.10summary	190
8.126.2.11testScenario	190
8.127VICI::cdi::TestEvent Class Reference	190
8.127.1 Detailed Description	191
8.127.2 Constructor & Destructor Documentation	191
8.127.2.1 TestEvent	191
8.127.3 Member Function Documentation	191
8.127.3.1 id	191
8.128VICI::cdi::TestEventQueue Class Reference	191
8.128.1 Detailed Description	192
8.128.2 Member Function Documentation	192
8.128.2.1 enqueueEvent	192
8.128.2.2 event	192
8.128.2.3 instance	192
8.129VICI::cdi::TestFactory Class Reference	193
8.129.1 Detailed Description	193
8.129.2 Member Function Documentation	193
8.129.2.1 make	193
8.130VICI::cdi::TestFT< T > Class Template Reference	193
8.130.1 Detailed Description	194
8.130.2 Member Function Documentation	194
8.130.2.1 make	194
8.131VICI::gth::TestsForWindow Class Reference	194
8.131.1 Detailed Description	195
8.132VICI::cdi::TestT< T > Class Template Reference	195
8.132.1 Detailed Description	196
8.133VICI::Symbol::TextAttributes Class Reference	196

8.133.1 Detailed Description	196
8.134VICI::cdi::Trace Class Reference	196
8.134.1 Detailed Description	197
8.134.2 Constructor & Destructor Documentation	197
8.134.2.1 Trace	197
8.134.2.2 Trace	197
8.135VICI::cfi::TraceLogger Class Reference	197
8.135.1 Detailed Description	198
8.135.2 Constructor & Destructor Documentation	198
8.135.2.1 TraceLogger	198
8.135.3 Member Function Documentation	198
8.135.3.1 log	198
8.135.3.2 log	198
8.136VICI::cdi::Tracer Class Reference	199
8.136.1 Detailed Description	199
8.136.2 Member Function Documentation	199
8.136.2.1 log	199
8.137VICI::cfi::UDPClientSocket Class Reference	200
8.137.1 Detailed Description	200
8.137.2 Constructor & Destructor Documentation	200
8.137.2.1 UDPClientSocket	200
8.138VICI::cfi::UDPLogger Class Reference	201
8.138.1 Detailed Description	201
8.138.2 Constructor & Destructor Documentation	201
8.138.2.1 UDPLogger	201
8.138.3 Member Function Documentation	201
8.138.3.1 log	201
8.138.3.2 log	202
8.139VICI::cfi::UDPServerSocket Class Reference	202
8.139.1 Detailed Description	203
8.139.2 Constructor & Destructor Documentation	203
8.139.2.1 UDPServerSocket	203
8.139.3 Member Function Documentation	203
8.139.3.1 send	203
8.140VICI::cfi::UDPSSocket Class Reference	203
8.140.1 Detailed Description	204
8.140.2 Member Function Documentation	204
8.140.2.1 recv	204
8.140.2.2 send	204
8.140.3 Member Data Documentation	205

8.140.3.1 BUFFER_SIZE	205
8.141 VICI::gth::UserEscaper Class Reference	205
8.141.1 Detailed Description	205
8.142 VICI::VDialog Class Reference	205
8.142.1 Detailed Description	206
8.143 VICI::VEditList Class Reference	206
8.143.1 Detailed Description	207
8.144 VICI::Vici Class Reference	207
8.144.1 Detailed Description	207
8.145 VICI::Admin::ViciAdmin Class Reference	208
8.145.1 Detailed Description	208
8.146 VICI::Admin::ViciAdminFactory Class Reference	208
8.146.1 Detailed Description	209
8.146.2 Member Function Documentation	209
8.146.2.1 makeViciAdmin	209
8.147 VICI::stub::ViciAdminStub Class Reference	209
8.147.1 Detailed Description	209
8.148 VICI::stub::ViciAdminStubFactory Class Reference	209
8.148.1 Detailed Description	210
8.148.2 Member Function Documentation	210
8.148.2.1 makeViciAdmin	210
8.149 VICI::Ed::ViciEditor Class Reference	210
8.149.1 Detailed Description	211
8.150 VICI::Ed::ViciEditorFactory Class Reference	211
8.150.1 Detailed Description	211
8.151 VICI::stub::ViciEditorStub Class Reference	212
8.151.1 Detailed Description	212
8.151.2 Member Function Documentation	213
8.151.2.1 cmdnError	213
8.151.2.2 newSymbol	214
8.151.2.3 optionsAndParameters	214
8.151.2.4 selectedCommand	214
8.151.2.5 selection	214
8.151.2.6 symbolAttr	214
8.151.2.7 textAttr	215
8.152 VICI::stub::ViciEditorStubFactory Class Reference	216
8.152.1 Detailed Description	216
8.153 VICI::ViciFactory Class Reference	216
8.153.1 Detailed Description	217
8.154 VICI::stub::ViciStub Class Reference	217

8.154.1 Detailed Description	218
8.154.2 Member Function Documentation	218
8.154.2.1 breakReached	218
8.154.2.2 dataReady	218
8.154.2.3 reportError	218
8.154.2.4 setCursor	218
8.154.2.5 setFile	219
8.154.2.6 setValue	219
8.155VICI::stub::ViciStubFactory Class Reference	219
8.155.1 Detailed Description	220
8.156VICI::VMainWindow Class Reference	220
8.156.1 Detailed Description	220
8.157VICI::VWindow Class Reference	221
8.157.1 Detailed Description	221
8.157.2 Constructor & Destructor Documentation	221
8.157.2.1 VWindow	221
8.158VICI::vxt< N > Class Template Reference	221
8.158.1 Detailed Description	222
8.158.2 Constructor & Destructor Documentation	223
8.158.2.1 vxt	223
8.158.2.2 vxt	223
8.159VICI::WidgetMgr Class Reference	223
8.159.1 Detailed Description	224
8.160VICI::WidgetMgrClient Class Reference	224
8.160.1 Detailed Description	224
8.161VICI::Window Class Reference	224
8.161.1 Detailed Description	225
8.162VICI::stub::WindowStub Class Reference	225
8.162.1 Detailed Description	225
8.163VICI::cfi::XINI Class Reference	225
8.163.1 Detailed Description	226
8.163.2 Member Function Documentation	227
8.163.2.1 configure	227
8.163.2.2 configure	227
8.163.2.3 getConfigFilename	227
8.163.2.4 getPath	227
8.163.2.5 getVal	227
8.163.2.6 getVals	228
8.163.3 Member Data Documentation	228
8.163.3.1 config	228

8.164	VICI::cfi::Xml Class Reference	228
8.164.1	Detailed Description	231
8.164.2	Member Function Documentation	231
8.164.2.1	create	231
8.164.2.2	deleteNode	231
8.164.2.3	dirty	231
8.164.2.4	find	231
8.164.2.5	getChild	231
8.164.2.6	getChildren	232
8.164.2.7	getDtdIdentifiers	232
8.164.2.8	getNodeContent	232
8.164.2.9	getNodeName	232
8.164.2.10	getPropDouble	232
8.164.2.11	getPropInt	232
8.164.2.12	getPropShort	233
8.164.2.13	getPropString	233
8.164.2.14	newNode	233
8.164.2.15	newTextChild	233
8.164.2.16	open	233
8.164.2.17	registerNamespace	234
8.164.2.18	safeToSave	234
8.164.2.19	save	234
8.164.2.20	setCDATAContent	234
8.164.2.21	setCompression	234
8.164.2.22	setContent	234
8.164.2.23	setDirty	234
8.164.2.24	setDtd	235
8.164.2.25	setDtd	235
8.164.2.26	setProp	235
8.164.2.27	setProp	235
8.164.2.28	setProp	235
8.165	VICI::cfi::XmlBuffer Class Reference	236
8.165.1	Detailed Description	236
9	File Documentation	237
9.1	canvas.h File Reference	237
9.1.1	Detailed Description	237
9.2	discover.h File Reference	237
9.2.1	Detailed Description	238
9.2.2	Macro Definition Documentation	238

9.2.2.1	DISCOVERABLE	238
9.3	fdstream.h File Reference	238
9.3.1	Detailed Description	239
9.4	gth.h File Reference	239
9.4.1	Detailed Description	240
9.5	ipc.h File Reference	240
9.5.1	Detailed Description	240
9.6	libgth.h File Reference	240
9.6.1	Detailed Description	242
9.7	libgui.h File Reference	242
9.7.1	Detailed Description	243
9.8	libfstubs.h File Reference	243
9.8.1	Detailed Description	245
9.9	log.h File Reference	245
9.9.1	Detailed Description	246
9.10	parseTree.h File Reference	246
9.10.1	Detailed Description	246
9.11	plugin.h File Reference	246
9.11.1	Detailed Description	247
9.11.2	Function Documentation	248
9.11.2.1	closeViciPlugin	248
9.11.2.2	initViciPlugin	248
9.12	proc.h File Reference	248
9.12.1	Detailed Description	248
9.13	stringy.h File Reference	249
9.13.1	Detailed Description	249
9.14	test.h File Reference	249
9.14.1	Detailed Description	250
9.15	testmgr.h File Reference	250
9.15.1	Detailed Description	251
9.16	trace.h File Reference	251
9.16.1	Detailed Description	252
9.17	udpsocket.h File Reference	252
9.17.1	Detailed Description	253
9.18	vici.h File Reference	253
9.18.1	Detailed Description	257
9.19	vx.h File Reference	257
9.19.1	Detailed Description	258
9.20	window.h File Reference	258
9.20.1	Detailed Description	258

9.21 xini.h File Reference	259
9.21.1 Detailed Description	259
9.22 xml.h File Reference	259
9.22.1 Detailed Description	260
Index	261

Chapter 1

Main Page

[VICI](#) aims to make the UNIX command line programs easily available to the users of GUI desktop environments.

A flow chart can be built with commands and their parameters. This can then be saved as a script which can be installed into the user's desktop as an application.

The script supports all Linux commands, pipelines, file redirection, variables, functions, and multiple threads of control.

Chapter 2

Todo List

Member `VICI::Canvas::Canvas::setExecution` (bool active, csr node)=0

node should be NodeId type

Member `VICI::Canvas::Canvas::symbolAttr` (Symbol::SymbolAttributes &att)=0

this appears to be meaningless without the symbol

Member `VICI::Symbol::SymbolClient::selection` (Symbol *sym)=0

do we need to notify of deselection ?

Class `VICI::Symbol::TextAttributes`

should also include the style - bold, italic, normal

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

VICI	The namespace for the Visual Chart Interpreter project	21
VICI::Admin	An API for the vici-adm program	25
VICI::Canvas	An API for the Canvas which handles the drawing and layout	26
VICI::cdi	The namespace for the Common Development Infrastructure	26
VICI::cfi	The namespace for the Common Facilities Infrastructure components	27
VICI::Cmnd	An API for libcommand	29
VICI::Cron	API for libcron	30
VICI::EBNF	An API for the libebnf library	30
VICI::Ed	An API for the vici editor	31
VICI::gth	The namespace for the GUI Test Harness	31
VICI::Inst	An API for the installer	34
VICI::Interp	An API for the vici script interpreter	34
VICI::Search	An API for libsearch	35
VICI::Sec	An API for libsecure	35
VICI::stub	The namespace for stub versions of modules	36
VICI::Symbol	An API for libsymbols	37
VICI::Syntax	An API for libsyntax	39

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

VICI::cfi::AbstractChildProcess	41
VICI::cfi::ChildProcess	67
VICI::cdi::AbstractScenario	42
VICI::cdi::ScenarioT< DefaultScenario >	151
VICI::cdi::DefaultScenario	80
VICI::cdi::ScenarioT< GTHTest >	151
VICI::gth::GTHTest	109
VICI::cdi::ScenarioT< T >	151
VICI::cdi::AbstractTest	43
VICI::cdi::TestT< DefaultTest >	195
VICI::cdi::DefaultTest	80
VICI::cdi::TestT< T >	195
VICI::cdi::AbstractTestCase	44
VICI::cdi::TestCaseT< DefaultTestCase >	186
VICI::gth::DefaultTestCase	81
VICI::cdi::AsyncTestCase	50
VICI::cdi::AsyncTestCaseT< T >	52
VICI::cdi::TestCaseT< T >	186
VICI::gth::Adaptor	46
VICI::gth::AdaptorST< wt >	47
VICI::gth::AdaptorT< wt, T >	48
std::basic_string< Char >	
std::string	
VICI::Path	136
VICI::cdi::CallTrace	54
VICI::Canvas::CanvasClient	58
VICI::Ed::ViciEditor	210
VICI::stub::ViciEditorStub	212
VICI::cfi::ChildProcessMgr	69
VICI::Cmnd::CommandClient	71
VICI::Ed::ViciEditor	210
VICI::Cron::Cron	75
VICI::stub::CronStub	77
VICI::cfi::Discoverable	82
VICI::cfi::DiscoverPointer	82

VICI::cfi::DiscoveryMgr	83
VICI::stub::Dispatcher	85
VICI::EBNF::EBNF	87
VICI::stub::EBNF_Stub	89
VICI::EbnfNode	91
VICI::stub::Event	96
VICI::stub::AnEvent< T >	49
std::exception	
std::runtime_error	
VICI::vxt< N >	221
VICI::cdi::scenario_exception	149
VICI::cdi::test_exception	183
VICI::Factory	96
VICI::Admin::ViciAdminFactory	208
VICI::stub::ViciAdminStubFactory	209
VICI::Canvas::CanvasFactory	59
VICI::stub::CanvasStubFactory	67
VICI::Cmnd::CommandFactory	72
VICI::stub::CommandStubFactory	74
VICI::Cron::CronFactory	76
VICI::stub::CronStubFactory	79
VICI::EBNF::EBNF_Factory	88
VICI::stub::EBNF_Stub_Factory	90
VICI::Ed::ViciEditorFactory	211
VICI::stub::ViciEditorStubFactory	216
VICI::Inst::InstallerFactory	113
VICI::stub::InstallerStubFactory	115
VICI::Interp::InterpreterFactory	121
VICI::stub::InterpreterStubFactory	126
VICI::Search::SearchFactory	155
VICI::stub::SearchStubFactory	157
VICI::Sec::SecureFactory	159
VICI::stub::SecureStubFactory	161
VICI::Symbol::SymbolFactory	169
VICI::stub::SymbolStubFactory	177
VICI::Syntax::SyntaxFactory	179
VICI::stub::SyntaxStubFactory	181
VICI::ViciFactory	216
VICI::stub::ViciStubFactory	219
VICI::FactoryFactory	97
VICI::cfi::FD	98
VICI::gth::GTHTestCase	110
VICI::gth::DefaultTestCase	81
VICI::GTHWindowWidget	111
VICI::VDialog	205
VICI::AboutDialog	41
VICI::VMainWindow	220
VICI::Inst::Installer	112
VICI::stub::InstallerStub	114
VICI::Interp::Interpreter	115
VICI::stub::InterpreterStub	122
VICI::Interp::InterpreterClient	119
VICI::Canvas::Canvas	55
VICI::stub::CanvasStub	62

VICI::Vici	207
VICI::stub::ViciStub	217
std::ios_base	
std::basic_ios	
std::basic_istream	
std::istream	
VICI::cfi::fdistream	102
std::basic_ostream	
std::ostream	
VICI::cfi::fdostream	103
VICI::cfi::logstream	131
VICI::cfi::logger	130
VICI::cfi::FormattingLogger	108
VICI::cfi::FileLogger	106
VICI::cfi::StdLogger	164
VICI::cfi::UDPLLogger	201
VICI::cfi::PlainFileLogger	139
VICI::cfi::SystemLogger	181
VICI::cfi::TraceLogger	197
VICI::gth::LuaScript	133
VICI::Metrics	133
VICI::cdi::NullTrace	135
VICI::EBNF::ParseTree	135
VICI::EbnfTree	93
VICI::cfi::PlugIn	140
VICI::cfi::AutoRunPlugIn	54
VICI::cfi::PlugInDescriptor	141
VICI::cfi::PlugInDetails	142
VICI::cfi::PlugInFactory	142
VICI::cfi::PlugInFamilyFactoryT< F >	144
VICI::cfi::PlugInFactoryT< F, P >	143
VICI::cfi::PlugInLib	145
VICI::cfi::PlugInMgr	147
VICI::cfi::ProcessOwner	148
QDialog	
VICI::VDialog	205
QItemDelegate	
VICI::ItemDelegate	127
QListWidget	
VICI::VEditList	206
QMainWindow	
VICI::VMainWindow	220
QObject	
VICI::gth::GTHTest	109
VICI::gth::MouseEventReporter	134
VICI::gth::UserEscaper	205
VICI::SignalToQtSignal	163
QThread	
VICI::gth::TestsForWindow	194
VICI::cdi::ScenarioFactory	149
VICI::cdi::ScenarioFT< T >	150
VICI::cdi::ScenarioResults	151
VICI::Scene	152
VICI::CanvasScene	60
VICI::Search::Search	154
VICI::stub::SearchStub	156

VICI::Search::SearchClient	154
VICI::Ed::ViciEditor	210
VICI::Sec::Secure	158
VICI::stub::SecureStub	160
VICI::cfi::Semaphore	162
VICI::cfi::SemaphoreLock	162
streambuf	
logbuff	128
VICI::cfi::fdinbuf	99
VICI::cfi::fdoutbuf	104
VICI::Symbol::Symbol	165
VICI::stub::SymbolStub	174
VICI::Symbol::SymbolAttributes	167
VICI::Symbol::SymbolClient	168
VICI::Canvas::Canvas	55
VICI::Cmnd::Command	70
VICI::stub::CommandStub	73
VICI::Ed::ViciEditor	210
VICI::Symbol::SymbolMgr	170
VICI::stub::SymbolMgrStub	172
VICI::Symbol::SymbolOwner	174
VICI::Syntax::Syntax	178
VICI::stub::SyntaxStub	179
VICI::gth::TestAction	183
VICI::cdi::TestCaseFactory	184
VICI::cdi::TestCaseFT< T >	185
VICI::cdi::Tester	187
VICI::cdi::TestEvent	190
VICI::cdi::TestEventQueue	191
VICI::cdi::TestFactory	193
VICI::cdi::TestFT< T >	193
VICI::Symbol::TextAttributes	196
VICI::cdi::Trace	196
VICI::cdi::Tracer	199
VICI::cfi::UDPSocket	203
VICI::cfi::UDPClientSocket	200
VICI::cfi::UDPServerSocket	202
VICI::Admin::ViciAdmin	208
VICI::stub::ViciAdminStub	209
VICI::WidgetMgr	223
VICI::WidgetMgrClient	224
VICI::gth::GTHTest	109
VICI::Window	224
VICI::stub::WindowStub	225
VICI::VWindow	221
VICI::cfi::Xml	228
VICI::cfi::XINI	225
VICI::EbnfXml	94
VICI::gth::ScriptXML	153
VICI::cfi::XmlBuffer	236

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

VICI::AboutDialog	A standard About Vici dialog for the project	41
VICI::cfi::AbstractChildProcess	Abstract child process interface for ChildProcessMgr	41
VICI::cdi::AbstractScenario	Define a type for scenarios	42
VICI::cdi::AbstractTest	Responsible for managing resources needed for the entire test	43
VICI::cdi::AbstractTestCase	Base class for test cases	44
VICI::gth::Adaptor	Defines an abstract base class for widget adaptors	46
VICI::gth::AdaptorST< wt >	Adds a static method to Adaptor to describe it;	47
VICI::gth::AdaptorT< wt, T >	The interpreter for the script commands	48
VICI::stub::AnEvent< T >	A template wrapper for function calls	49
VICI::cdi::AsyncTestCase	Responsible for handling asynchronous tests	50
VICI::cdi::AsyncTestCaseT< T >	Responsible for installing the factory for the test case type	52
VICI::cfi::AutoRunPlugIn	Base class for plug ins that are run immediately that the library is loaded	54
VICI::cdi::CallTrace	Class to create a call trace	54
VICI::Canvas::Canvas	The facade for canvas library	55
VICI::Canvas::CanvasClient	An interface that is notified of canvas actions	58
VICI::Canvas::CanvasFactory	An abstract factory for making an instance of Canvas	59
VICI::CanvasScene	An implementation of the Scene abstract class for holding a QGraphicsScene	60
VICI::stub::CanvasStub	A stub version of the Canvas module	62
VICI::stub::CanvasStubFactory	A factory for creating stub instances of Canvas objects	67

VICI::cfi::ChildProcess	Represents the state of a child process	67
VICI::cfi::ChildProcessMgr	Manage the child processes	69
VICI::Cmnd::Command	The facade for the command library	70
VICI::Cmnd::CommandClient	An interface that is notified of a command selection	71
VICI::Cmnd::CommandFactory	An abstract factory for making an instance of Command	72
VICI::stub::CommandStub	A stub version of the Command module	73
VICI::stub::CommandStubFactory	A factory for creating stub instances of Command objects	74
VICI::Cron::Cron	Allow scripts to be scheduled for later running	75
VICI::Cron::CronFactory	An abstract factory for making an instance of Cron	76
VICI::stub::CronStub	A stub version of the Cron module	77
VICI::stub::CronStubFactory	A factory for creating stub instances of Cron objects	79
VICI::cdi::DefaultScenario	Provide a default scenario object	80
VICI::cdi::DefaultTest	Provide a default version of the AbstractTest object	80
VICI::gth::DefaultTestCase	Provides a test case to use for actions that don't have an explicit test case	81
VICI::cfi::Discoverable	A mixin class that makes its owner discoverable	82
VICI::cfi::DiscoverPointer	Holds a pointer to a discoverable object	82
VICI::cfi::DiscoveryMgr	Manager for discoverable objects	83
VICI::stub::Dispatcher	Send events to registered objects	85
VICI::EBNF::EBNF	A parser for EBNF	87
VICI::EBNF::EBNF_Factory	An abstract factory for making an instance of EBNF	88
VICI::stub::EBNF_Stub	A stub version of the EBNF module interface	89
VICI::stub::EBNF_Stub_Factory	A factory for creating instances of the EBNF_Stub	90
VICI::EbnfNode	A node of the EBNF parse tree	91
VICI::EbnfTree	An implementation of the ParseTree type	93
VICI::EbnfXml	A specialization of the Xml class for the parse tree	94
VICI::stub::Event	Define an Abstract base type for Dispatcher events	96
VICI::Factory	An abstract type for factories	96
VICI::FactoryFactory	Responsible for creating and supplying factories for the main modules	97
VICI::cfi::FD	Wrap file descriptors in a class to ensure closed	98

VICI::cfi::fdinbuf	An input stream buffer	99
VICI::cfi::fdistream	A file descriptor input stream	102
VICI::cfi::fdostream	A file descriptor output stream	103
VICI::cfi::fdoutbuf	An output stream buffer	104
VICI::cfi::FileLogger	Class for logging to a file	106
VICI::cfi::FormattingLogger	Class for producing a formatted log message	108
VICI::gth::GTHTest	An AbstractTest derived class for testing GUI programs	109
VICI::gth::GTHTestCase	Interface for test cases that do gui testing	110
VICI::GTHWindowWidget	A mix-in class with the functions required for the gui test harness	111
VICI::Inst::Installer	Install a script into the user's desktop	112
VICI::Inst::InstallerFactory	An abstract factory for making an instance of Installer	113
VICI::stub::InstallerStub	A stub version of the Installer module	114
VICI::stub::InstallerStubFactory	A factory for creating stub instances of Installer objects	115
VICI::Interp::Interpreter	The API for interpreter library	115
VICI::Interp::InterpreterClient	The interface that must be implemented by clients of the interpreter	119
VICI::Interp::InterpreterFactory	An abstract factory for making an instance of Interpreter	121
VICI::stub::InterpreterStub	A stub version of the Interpreter module	122
VICI::stub::InterpreterStubFactory	A factory for creating stub instances of Interpreter objects	126
VICI::ItemDelegate	An item delegate that uses QLineEdit for editing list and table entries	127
logbuff	A streambuf class specialized for logging	128
VICI::cfi::logger	An abstract base class used by the log stream to write logs	130
VICI::cfi::logstream	A stream class specialized for logging	131
VICI::gth::LuaScript	Provides a wrapper for a lua_State object	133
VICI::Metrics	A class to encapsulate measurements	133
VICI::gth::MouseEventReporter	Provides a means of getting scene coordinates in a QGraphicsView	134
VICI::cdi::NullTrace	A class for dummy trace objects	135
VICI::EBNF::ParseTree	A type for the parsed form of EBNF	135
VICI::Path	Manipulate path strings	136
VICI::cfi::PlainFileLogger		139

VICI::cfi::PlugIn	Base class for objects loaded from dynamically loaded libraries	140
VICI::cfi::PlugInDescriptor	Descriptor for plug-ins that can be used by the application	141
VICI::cfi::PlugInDetails	Details of plug-ins as provided by the loaded library	142
VICI::cfi::PlugInFactory	Base class for factories that create plug-in objects	142
VICI::cfi::PlugInFactoryT< F, P >	Template class for factories	143
VICI::cfi::PlugInFamilyFactoryT< F >	Template base class for families of plug-in factories	144
VICI::cfi::PlugInLib	Manages an instance of a dynamically loaded shared library	145
VICI::cfi::PlugInMgr	Manage the handling of plug-in shared libraries	147
VICI::cfi::ProcessOwner	Interface that allows the owner of a child process to be notified	148
VICI::cdi::scenario_exception	Throw this to abandon an entire scenario	149
VICI::cdi::ScenarioFactory	Define a type for scenario factories	149
VICI::cdi::ScenarioFT< T >	Responsible for creating a scenario of some type	150
VICI::cdi::ScenarioResults	Responsible for storing the results of testing for a scenario	151
VICI::cdi::ScenarioT< T >	Responsible for installing a factory to create scenarios of the required type	151
VICI::Scene	A wrapper for QGraphicsScene class	152
VICI::gth::ScriptXML	Provides an interface to the script XML file	153
VICI::Search::Search	Allow the user to search for, and organise commands	154
VICI::Search::SearchClient	The interface that must implemented for clients of Search	154
VICI::Search::SearchFactory	An abstract factory for making an instance of Search	155
VICI::stub::SearchStub	A stub version of the Search module	156
VICI::stub::SearchStubFactory	A factory for creating stub instances of Search objects	157
VICI::Sec::Secure	Provide security for the scripts	158
VICI::Sec::SecureFactory	An abstract factory for making an instance of Secure	159
VICI::stub::SecureStub	A stub version of the Secure module	160
VICI::stub::SecureStubFactory	A factory for creating stub instances of Security objects	161
VICI::cfi::Semaphore	A semaphore for managing exclusive access to resources across multiple processes	162
VICI::cfi::SemaphoreLock	Mutual exclusion lock	162
VICI::SignalToQtSignal	A class to convert operating system signals into Qt signals	163
VICI::cfi::StdLogger	Class for logging to the std output streams	164

VICI::Symbol::Symbol	Represents something that is drawn on the canvas	165
VICI::Symbol::SymbolAttributes	Holds the attributes of a symbol	167
VICI::Symbol::SymbolClient	An interface that is notified of changes to symbol manager	168
VICI::Symbol::SymbolFactory	An abstract factory for making an instance of SymbolMgr	169
VICI::Symbol::SymbolMgr	The facade for the symbol library	170
VICI::stub::SymbolMgrStub	A stub version of the Symbol Manager	172
VICI::Symbol::SymbolOwner	Represents something that is notified about events occurring on a symbol	174
VICI::stub::SymbolStub	A stub version of the Symbol class	174
VICI::stub::SymbolStubFactory	A factory for creating stub instances of Symbol Manager objects	177
VICI::Syntax::Syntax	Display a syntax chart of command options	178
VICI::Syntax::SyntaxFactory	An abstract factory for making an instance of Syntax	179
VICI::stub::SyntaxStub	A stub version of the Syntax module interface	179
VICI::stub::SyntaxStubFactory	A factory for creating SyntaxStub objects	181
VICI::cfi::SystemLogger	Logger that interfaces to the Linux syslog	181
VICI::cdi::test_exception	Throw this to abandon a particular test case	183
VICI::gth::TestAction	A single action that can be applied to a GUI	183
VICI::cdi::TestCaseFactory	Define a base type for test case factories	184
VICI::cdi::TestCaseFT< T >	Responsible for creating a test case of some type	185
VICI::cdi::TestCaseT< T >	Responsible for installing a factory for the test case	186
VICI::cdi::Tester	Responsible for managing the testing	187
VICI::cdi::TestEvent	Represent an event in the object under test	190
VICI::cdi::TestEventQueue	Responsible for queuing TestEvents	191
VICI::cdi::TestFactory	Responsible for creating an object that manages the resources for the entire test	193
VICI::cdi::TestFT< T >	Responsible for creating a test object of the required type	193
VICI::gth::TestsForWindow	Provides a thread for applying test cases to a window	194
VICI::cdi::TestT< T >	Responsible for installing a factory for making test objects	195
VICI::Symbol::TextAttributes	Hold the attributes of a text comment	196
VICI::cdi::Trace	This class is used to create trace log entries	196
VICI::cfi::TraceLogger	Class for logging tracing output	197

VICI::cdi::Tracer	This class manages the tracing for an application	199
VICI::cfi::UDPClientSocket	A UDP client socket	200
VICI::cfi::UDPLogger	Class for logging to a logging server	201
VICI::cfi::UDPServerSocket	A UDP Server socket	202
VICI::cfi::UDPSocket	An abstract base class with common stuff for UDP sockets	203
VICI::gth::UserEscaper	A class for allowing the user to suspend the test	205
VICI::VDialog	A dialog class that automatically registers itself	205
VICI::VEditList	Builds on QListWidget to provide better editing ability	206
VICI::Vici	An API for the vici runtime gui	207
VICI::Admin::ViciAdmin	Application for preparing commands	208
VICI::Admin::ViciAdminFactory	An abstract factory for making an instance of ViciAdmin	208
VICI::stub::ViciAdminStub	A stub version of the ViciAdmin module interface	209
VICI::stub::ViciAdminStubFactory	A factory for creating stub instances of ViciAdmin	209
VICI::Ed::ViciEditor	The flowchart editor	210
VICI::Ed::ViciEditorFactory	An abstract factory for making an instance of ViciEditor	211
VICI::stub::ViciEditorStub	A stub version of the Vici Editor module	212
VICI::stub::ViciEditorStubFactory	A factory for creating stub instances of Vici Editor objects	216
VICI::ViciFactory	An abstract factory for making an instance of Vici	216
VICI::stub::ViciStub	A stub version of the Vici module	217
VICI::stub::ViciStubFactory	A factory for creating stub instances of Vici objects	219
VICI::VMainWindow	A main window class that automatically registers itself	220
VICI::VWindow	An implementation of Window for holding Qt's QWidget objects	221
VICI::vxt< N >	An exception object with severity levels	221
VICI::WidgetMgr	A singleton class that window widgets register with	223
VICI::WidgetMgrClient	An interface for the WidgetMgr to call its clients when a window registers itself	224
VICI::Window	A wrapper class for windows	224
VICI::stub::WindowStub	A stub version of window objects	225
VICI::cfi::XINI	Load an XML configuration file	225
VICI::cfi::Xml	A C++ wrapper for libxml2	228

[VICI::cfi::XmlBuffer](#)

An abstract class defining a buffer object [236](#)

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

canvas.h	The API for handling graphics scenes between modules	237
discover.h	Provides the API for objects that need to be discoverable for testing	237
fdstream.h	Provide C++ stream interface to file descriptor integers	238
gth.h	Provide the interface between the GUI applications and the GUI Test Harness	239
ipc.h	Declarations for the semaphore component of libcfi	240
libgth.h	Provides the API for the GUI Test Harness library	240
libgui.h	Provides project wide GUI components	242
libfstubs.h	Provide a stub implementation for all VICI modules	243
log.h	Declarations for the logging component of libcfi	245
parseTree.h	The API for implementation of the EBNF Parse Tree	246
plugin.h	Declarations for the plug-in component of libcfi	246
proc.h	Declarations for classes that manage child processes	248
stringy.h	Useful string functions	249
test.h	Interface between applications and the test harness	249
testmgr.h	Declarations for the test harness	250
trace.h	Support for tracing the execution path	251
udpsocket.h	Declarations for a wrapper for the UDP socket	252
vici.h	Project wide declarations and definitions	253
vx.h	An exception object with stream semantics	257

window.h	The API for handling windows between modules	258
xini.h	A class for XML based configuration file	259
xml.h	A simple C++ wrapper for libxml2	259

Chapter 7

Namespace Documentation

7.1 VICI Namespace Reference

The namespace for the Visual Chart Interpreter project.

Namespaces

- [Admin](#)
An API for the vici-adm program.
- [Canvas](#)
An API for the [Canvas](#) which handles the drawing and layout.
- [cdi](#)
The namespace for the Common Development Infrastructure.
- [cfi](#)
The namespace for the Common Facilities Infrastructure components.
- [Cmnd](#)
An API for libcommand.
- [Cron](#)
an API for libcron
- [EBNF](#)
An API for the libebnf library.
- [Ed](#)
An API for the vici editor.
- [gth](#)
The namespace for the GUI Test Harness.
- [Inst](#)
An API for the installer.
- [Interp](#)
An API for the vici script interpreter.
- [Search](#)
An API for libsearch.
- [Sec](#)
An API for libsecure.
- [stub](#)
The namespace for stub versions of modules.
- [Symbol](#)
An API for libsymbol.
- [Syntax](#)
An API for libsyntax.

Classes

- class [Path](#)
Manipulate path strings.
- class [vxt](#)
An exception object with severity levels.
- class [GTHWindowWidget](#)
A mix-in class with the functions required for the gui test harness.
- class [WidgetMgrClient](#)
An interface for the [WidgetMgr](#) to call its clients when a window registers itself.
- class [WidgetMgr](#)
A singleton class that window widgets register with.
- class [VMainWindow](#)
A main window class that automatically registers itself.
- class [VDialog](#)
A dialog class that automatically registers itself.
- class [ItemDelegate](#)
An item delegate that uses [QLineEdit](#) for editing list and table entries.
- class [VEditList](#)
Builds on [QListWidget](#) to provide better editing ability.
- class [Metrics](#)
A class to encapsulate measurements.
- class [SignalToQtSignal](#)
A class to convert operating system signals into Qt signals.
- class [AboutDialog](#)
A standard About [Vici](#) dialog for the project.
- class [CanvasScene](#)
An implementation of the [Scene](#) abstract class for holding a [QGraphicsScene](#).
- class [EbnfNode](#)
A node of the [EBNF](#) parse tree.
- class [EbnfTree](#)
An implementation of the [ParseTree](#) type.
- class [EbnfXml](#)
A specialization of the [Xml](#) class for the parse tree.
- class [Factory](#)
An abstract type for factories.
- class [FactoryFactory](#)
Responsible for creating and supplying factories for the main modules.
- class [Window](#)
A wrapper class for windows.
- class [Scene](#)
A wrapper for [QGraphicsScene](#) class.
- class [Vici](#)
An API for the vici runtime gui.
- class [ViciFactory](#)
An abstract factory for making an instance of [Vici](#).
- class [VWindow](#)
An implementation of [Window](#) for holding Qt's [QWidget](#) objects.

Typedefs

- typedef void(* [AsyncTestEventFn](#))(const std::string &s)
Pointer to function used to enqueue a test event.
- typedef const std::string & [csr](#)
short cut for string constants
- typedef int [NodeId](#)
Type for identifying a node of the flowchart.
- typedef int [ThreadId](#)
type for identifying a thread in the running script
- typedef std::vector< std::string > [ArgList](#)
Type for a list of command arguments and options.
- typedef std::shared_ptr< [Factory](#) > [FactoryPtr](#)
Shared pointer for factory.
- typedef std::shared_ptr
< [ViciFactory](#) > [ViciFactoryPtr](#)
Shared pointer for Vici Factory.

Enumerations

- enum [Severity](#) {
[Emergency](#), [Alert](#), [Critical](#), [Error](#),
[Code](#), [Warning](#), [Notice](#), [Info](#),
[Debug](#) }
Severity levels for log messages.
- enum [Module](#) {
[EBNF_Module](#), **[Syntax_Module](#)**, **[Admin_Module](#)**, **[Search_Module](#)**,
[Command_Module](#), **[Symbol_Module](#)**, **[Canvas_Module](#)**, **[Secure_Module](#)**,
[Cron_Module](#), **[Installer_Module](#)**, **[Interpreter_Module](#)**, **[Editor_Module](#)**,
[Vici_Module](#) }
An enum that lists the main modules of VICI.

Functions

- void [trim](#) (std::string &s)
rip off leading and trailing white spaces
- int [split](#) ([csr](#) text, std::vector< std::string > &result)
split a string into sub-strings at spaces
- std::string [expandMacros](#) ([csr](#) s)
expand a string containing \$ macros
- std::string [clean](#) ([csr](#) s)
Remove everything except alpha, digit, space, dot, hyphen and underscore.
- void [defaultAsyncTestEvent](#) (const std::string &s)
Function used to enqueue a test event.

Variables

- [AsyncTestEventFn](#) [asyncTestEvent](#) = [VICI::defaultAsyncTestEvent](#)
Pointer to function used to enqueue a test event.

7.1.1 Detailed Description

The namespace for the Visual Chart Interpreter project. The namespace for the project.

7.1.2 Typedef Documentation

7.1.2.1 typedef void(* VICI::AsyncTestEventFn)(const std::string &s)

Pointer to function used to enqueue a test event.

Parameters

s	Identity of the event.
---	------------------------

7.1.3 Enumeration Type Documentation

7.1.3.1 enum VICI::Severity

Severity levels for log messages.

Enumerator

Emergency A fault has been detected which may compromise the computer.

Alert A configuration error has been detected.

Critical The program cannot continue and may have corrupted its data.

Error The program cannot continue.

Code A programming error has been detected.

Warning There is a problem but the program can continue.

Notice Something is odd.

Info Information messages.

Debug Debugging messages.

7.1.4 Function Documentation

7.1.4.1 std::string VICI::clean (csr s)

Remove everything except alpha, digit, space, dot, hyphen and underscore.

Parameters

s	The string to clean
---	---------------------

Returns

A string devoid of all but the above

7.1.4.2 void VICI::defaultAsyncTestEvent (const std::string & s)

Function used to enqueue a test event.

The default function is defined in cfi/log.cpp and does nothing

Parameters

<i>s</i>	Identity of the event.
----------	------------------------

7.1.4.3 string VICI::expandMacros (csr s)

expand a string containing \$ macros

Parameters

<i>s</i>	The string to expand.
----------	-----------------------

Returns

A string with \$ macros replaced.

7.1.4.4 int VICI::split (csr text, std::vector< std::string > & result)

split a string into sub-strings at spaces

Parameters

<i>text</i>	the string to split
<i>result</i>	the vector of sub strings

Returns

a number of sub-strings

7.1.4.5 void VICI::trim (std::string & s)

rip off leading and trailing white spaces

Parameters

<i>s</i>	the string to trim
----------	--------------------

7.2 VICI::Admin Namespace Reference

An API for the vici-adm program.

Classes

- class [ViciAdmin](#)
Application for preparing commands.
- class [ViciAdminFactory](#)
An abstract factory for making an instance of [ViciAdmin](#).

Typedefs

- typedef std::shared_ptr
< [ViciAdminFactory](#) > [ViciAdminFactoryPtr](#)
Shared pointer for [Admin Factory](#).

7.2.1 Detailed Description

An API for the vici-adm program.

7.3 VICI::Canvas Namespace Reference

An API for the [Canvas](#) which handles the drawing and layout.

Classes

- class [CanvasClient](#)
An interface that is notified of canvas actions.
- class [Canvas](#)
The facade for canvas library.
- class [CanvasFactory](#)
An abstract factory for making an instance of [Canvas](#).

Typedefs

- typedef std::shared_ptr
< [CanvasFactory](#) > [CanvasFactoryPtr](#)
Shared pointer for [Canvas Factory](#).

7.3.1 Detailed Description

An API for the [Canvas](#) which handles the drawing and layout.

7.4 VICI::cdi Namespace Reference

The namespace for the Common Development Infrastructure.

Classes

- class [Tester](#)
Responsible for managing the testing.
- class [test_exception](#)
throw this to abandon a particular test case
- class [scenario_exception](#)
throw this to abandon an entire scenario
- class [TestEvent](#)
Represent an event in the object under test.
- class [TestEventQueue](#)
Responsible for queuing TestEvents.
- class [TestFactory](#)
Responsible for creating an object that manages the resources for the entire test.
- class [TestFT](#)
Responsible for creating a test object of the required type.
- class [AbstractTest](#)

- Responsible for managing resources needed for the entire test.*

 - class [TestT](#)

Responsible for installing a factory for making test objects.
 - class [DefaultTest](#)

Provide a default version of the [AbstractTest](#) object.
 - class [ScenarioFactory](#)

Define a type for scenario factories.
 - class [ScenarioFT](#)

Responsible for creating a scenario of some type.
 - struct [ScenarioResults](#)

Responsible for storing the results of testing for a scenario.
 - class [AbstractScenario](#)

Define a type for scenarios.
 - class [ScenarioT](#)

Responsible for installing a factory to create scenarios of the required type.
 - class [DefaultScenario](#)

Provide a default scenario object.
 - class [TestCaseFactory](#)

Define a base type for test case factories.
 - class [TestCaseFT](#)

Responsible for creating a test case of some type.
 - class [AbstractTestCase](#)

Base class for test cases.
 - class [TestCaseT](#)

Responsible for installing a factory for the test case.
 - class [AsyncTestCase](#)

Responsible for handling asynchronous tests.
 - class [AsyncTestCaseT](#)

Responsible for installing the factory for the test case type.
 - class [CallTrace](#)

Class to create a call trace.
 - class [NullTrace](#)

A class for dummy trace objects.
 - class [Trace](#)

This class is used to create trace log entries.
 - class [Tracer](#)

This class manages the tracing for an application.

7.4.1 Detailed Description

The namespace for the Common Development Infrastructure. The cdi namespace is for components that are used during development of VICI but are not part of the product used by a normal user.

7.5 VICI::cfi Namespace Reference

The namespace for the Common Facilities Infrastructure components.

Classes

- struct [DiscoverPointer](#)
Holds a pointer to a discoverable object.
- class [Discoverable](#)
A mixin class that makes its owner discoverable.
- class [DiscoveryMgr](#)
Manager for discoverable objects.
- class [FD](#)
Wrap file descriptors in a class to ensure closed.
- class [fdoutbuf](#)
An output stream buffer.
- class [fdostream](#)
A file descriptor output stream.
- class [fdinbuf](#)
An input stream buffer.
- class [fdistream](#)
A file descriptor input stream.
- class [Semaphore](#)
A semaphore for managing exclusive access to resources across multiple processes.
- class [SemaphoreLock](#)
Mutual exclusion lock.
- class [logstream](#)
A stream class specialized for logging.
- class [logger](#)
An abstract base class used by the log stream to write logs.
- class [FormattingLogger](#)
Class for producing a formatted log message.
- class [StdLogger](#)
Class for logging to the std output streams.
- class [FileLogger](#)
Class for logging to a file.
- class [PlainFileLogger](#)
- class [UDPLogger](#)
Class for logging to a logging server.
- class [TraceLogger](#)
Class for logging tracing output.
- class [SystemLogger](#)
logger that interfaces to the Linux syslog
- class [PlugIn](#)
Base class for objects loaded from dynamically loaded libraries.
- class [AutoRunPlugIn](#)
Base class for plug ins that are run immediately that the library is loaded.
- class [PlugInFactory](#)
Base class for factories that create plug-in objects.
- class [PlugInFamilyFactoryT](#)
Template base class for families of plug-in factories.
- class [PlugInFactoryT](#)
Template class for factories.
- struct [PlugInDescriptor](#)
Descriptor for plug-ins that can be used by the application.

- struct [PlugInDetails](#)
Details of plug-ins as provided by the loaded library.
- class [PlugInLib](#)
Manages an instance of a dynamically loaded shared library.
- class [PlugInMgr](#)
Manage the handling of plug-in shared libraries.
- class [AbstractChildProcess](#)
Abstract child process interface for [ChildProcessMgr](#).
- class [ChildProcess](#)
Represents the state of a child process.
- class [ProcessOwner](#)
Interface that allows the owner of a child process to be notified.
- class [ChildProcessMgr](#)
Manage the child processes.
- class [UDPSocket](#)
An abstract base class with common stuff for UDP sockets.
- class [UDPClientSocket](#)
A UDP client socket.
- class [UDPServerSocket](#)
A UDP Server socket.
- class [XINI](#)
Load an XML configuration file.
- class [XmlBuffer](#)
An abstract class defining a buffer object.
- class [Xml](#)
A C++ wrapper for libxml2.

Typedefs

- typedef std::shared_ptr
< [DiscoverPointer](#) > [DiscoverSharedPointer](#)
A shared pointer that will be owned by a discoverable object.
- typedef std::weak_ptr
< [DiscoverPointer](#) > [DiscoverWeakPointer](#)
A weak pointer that will be held by the [DiscoveryMgr](#).
- typedef std::shared_ptr
< [PlugInLib](#) > [PlugInLibPtr](#)
Shared pointer to [PlugInLib](#).
- typedef std::unique_ptr
< [AutoRunPlugIn](#) > [AutoRunPlugInPtr](#)
Unique pointer to an [AutoRunPlugIn](#).

7.5.1 Detailed Description

The namespace for the Common Facilities Infrastructure components. The cfi namespace is for components that are basic infrastructure for almost any large project.

7.6 VICI::Cmnd Namespace Reference

An API for libcommand.

Classes

- class [CommandClient](#)
An interface that is notified of a command selection.
- class [Command](#)
The facade for the command library.
- class [CommandFactory](#)
An abstract factory for making an instance of [Command](#).

Typedefs

- typedef std::shared_ptr
< [CommandFactory](#) > [CommandFactoryPtr](#)
Shared pointer for [Command Factory](#).

7.6.1 Detailed Description

An API for libcommand.

7.7 VICI::Cron Namespace Reference

an API for libcron

Classes

- class [Cron](#)
Allow scripts to be scheduled for later running.
- class [CronFactory](#)
An abstract factory for making an instance of [Cron](#).

Typedefs

- typedef std::shared_ptr
< [CronFactory](#) > [CronFactoryPtr](#)
Shared pointer for [Cron Factory](#).

7.7.1 Detailed Description

an API for libcron

7.8 VICI::EBNF Namespace Reference

An API for the libebnf library.

Classes

- class [ParseTree](#)
A type for the parsed form of [EBNF](#).
- class [EBNF](#)
A parser for [EBNF](#).
- class [EBNF_Factory](#)
An abstract factory for making an instance of [EBNF](#).

Typedefs

- typedef std::shared_ptr
< [EBNF_Factory](#) > [EBNF_FactoryPtr](#)
Shared pointer for [EBNF_Factory](#).

7.8.1 Detailed Description

An API for the libebnf library.

7.9 VICI::Ed Namespace Reference

An API for the vici editor.

Classes

- class [ViciEditor](#)
The flowchart editor.
- class [ViciEditorFactory](#)
An abstract factory for making an instance of [ViciEditor](#).

Typedefs

- typedef std::shared_ptr
< [ViciEditorFactory](#) > [ViciEditorFactoryPtr](#)
Shared pointer for Editor [Factory](#).

7.9.1 Detailed Description

An API for the vici editor.

7.10 VICI::gth Namespace Reference

The namespace for the GUI Test Harness.

Classes

- class [Adaptor](#)
Defines an abstract base class for widget adaptors.
- class [LuaScript](#)
Provides a wrapper for a lua_State object.
- struct [TestAction](#)
A single action that can be applied to a GUI.
- class [TestsForWindow](#)
Provides a thread for applying test cases to a window.
- class [UserEscaper](#)
A class for allowing the user to suspend the test.
- class [GHTTest](#)
An AbstractTest derived class for testing GUI programs.
- class [GHTTestCase](#)
Interface for test cases that do gui testing.
- class [DefaultTestCase](#)
Provides a test case to use for actions that don't have an explicit test case.
- class [ScriptXML](#)
Provides an interface to the script XML file.
- class [MouseEventReporter](#)
Provides a means of getting scene coordinates in a QGraphicsView.
- class [AdaptorST](#)
Adds a static method to [Adaptor](#) to describe it;.
- class [AdaptorT](#)
The interpreter for the script commands.

Typedefs

- typedef std::vector< std::string > [ParamList](#)
ParamList is used for the parameters of a command.
- typedef std::vector< std::list
< std::string > > [ListOfLists](#)
ListOfLists is used to describe the commands and parameters of an [Adaptor](#).
- typedef std::shared_ptr
< [ScriptXML](#) > [ScriptXmlPtr](#)
Provide for automatic destruction when owner is deleted.
- typedef std::shared_ptr
< [LuaScript](#) > [LuaScriptPtr](#)
Provides for automatic destruction when the owner is deleted.
- typedef std::shared_ptr
< [TestsForWindow](#) > [TestsForWindowPtr](#)
Provides for automatic destruction when the owner is deleted.

Enumerations

- enum [WidgetType](#) {
[Action](#), [Button](#), [CheckBox](#), [ComboBox](#),
[Dock](#), [List](#), [ListView](#), [Label](#),
[LineEdit](#), [SpinBox](#), [Splitter](#), [Table](#),
[Tabs](#), [TextEdit](#), [Tree](#), [View](#),
[Window](#), [MessageBox](#), [FileDialog](#), [FontDialog](#),
[ColorDialog](#), [Script](#) }

List the types of widgets that we can interact with during testing.

- enum [ConDes](#) { **Constructor**, **Destructor** }

Enumerates the stages at which the actions for a test case can be run.

Functions

- void [RegnFn](#) (std::function< void(void *, [WidgetType](#), [csr](#))> reg)

Register the widgets for use by the GUI Test Harness.

7.10.1 Detailed Description

The namespace for the GUI Test Harness.

7.10.2 Enumeration Type Documentation

7.10.2.1 enum VICI::gth::WidgetType

List the types of widgets that we can interact with during testing.

Enumerator

- Action** For objects of type QAction.
- Button** For objects of type QPushButton.
- CheckBox** For objects of type QCheckBox.
- ComboBox** For objects of type QComboBox.
- Dock** For objects of type QDockWidget.
- List** For objects of type QListWidget.
- ListView** For objects of type QListView.
- Label** For objects of type QLabel.
- LineEdit** For objects of type QLineEdit.
- SpinBox** For objects of type QSpinBox.
- Splitter** For objects of type QSplitter.
- Table** For objects of type QTableWidgetItem.
- Tabs** For objects of type QTabWidget.
- TextEdit** For objects of type QTextEdit.
- Tree** For objects of type QTreeView.
- View** For objects of type QGraphicsView.
- Window** For objects of type QMainWindow.
- MessageBox** For objects of type QMessageBox.
- FileDialog** For objects of type QFileDialog.
- FontDialog** For objects of type QFontDialog.
- ColorDialog** For objects of type QColorDialog.
- Script** For LUA Scripts.

7.10.3 Function Documentation

7.10.3.1 void VICI::gth::RegnFn (std::function< void(void *, WidgetType, csr)> reg)

Register the widgets for use by the GUI Test Harness.

The function is passed as a parameter so that the application code will be able to link even when the test plug-in has not been loaded.

The parameters for the function pointer are the address of the widget, the type of the widget, and a name that will refer to the widget in the test script.

Parameters

<i>reg</i>	The function that should be called for each widget.
------------	---

7.11 VICI::Inst Namespace Reference

An API for the installer.

Classes

- class [Installer](#)
Install a script into the user's desktop.
- class [InstallerFactory](#)
An abstract factory for making an instance of [Installer](#).

Typedefs

- typedef std::shared_ptr
< [InstallerFactory](#) > [InstallerFactoryPtr](#)
Shared pointer for [Installer Factory](#).

7.11.1 Detailed Description

An API for the installer.

7.12 VICI::Interp Namespace Reference

An API for the vici script interpreter.

Classes

- class [InterpreterClient](#)
The interface that must be implemented by clients of the interpreter.
- class [Interpreter](#)
The API for interpreter library.
- class [InterpreterFactory](#)
An abstract factory for making an instance of [Interpreter](#).

Typedefs

- typedef std::shared_ptr< [InterpreterFactory](#) > [InterpreterFactoryPtr](#)
Shared pointer for [Interpreter Factory](#).

7.12.1 Detailed Description

An API for the vici script interpreter.

7.13 VICI::Search Namespace Reference

An API for libsearch.

Classes

- class [SearchClient](#)
The interface that must implemented for clients of [Search](#).
- class [Search](#)
Allow the user to search for, and organise commands.
- class [SearchFactory](#)
An abstract factory for making an instance of [Search](#).

Typedefs

- typedef std::shared_ptr< [SearchFactory](#) > [SearchFactoryPtr](#)
Shared pointer for [Search Factory](#).

7.13.1 Detailed Description

An API for libsearch.

7.14 VICI::Sec Namespace Reference

An API for libsecure.

Classes

- class [Secure](#)
Provide security for the scripts.
- class [SecureFactory](#)
An abstract factory for making an instance of [Secure](#).

Typedefs

- typedef std::shared_ptr< [SecureFactory](#) > [SecureFactoryPtr](#)
Shared pointer for [Secure Factory](#).

7.14.1 Detailed Description

An API for libsecure.

7.15 VICI::stub Namespace Reference

The namespace for stub versions of modules.

Classes

- class [Event](#)
Define an Abstract base type for [Dispatcher](#) events.
- class [Dispatcher](#)
Send events to registered objects.
- class [AnEvent](#)
A template wrapper for function calls.
- class [WindowStub](#)
A stub version of window objects.
- class [EBNF_Stub](#)
A stub version of the [EBNF](#) module interface.
- class [EBNF_Stub_Factory](#)
A factory for creating instances of the [EBNF_Stub](#).
- class [SyntaxStub](#)
A stub version of the [Syntax](#) module interface.
- class [SyntaxStubFactory](#)
A factory for creating [SyntaxStub](#) objects.
- class [ViciAdminStub](#)
A stub version of the [ViciAdmin](#) module interface.
- class [ViciAdminStubFactory](#)
A factory for creating stub instances of [ViciAdmin](#).
- class [SearchStub](#)
A stub version of the [Search](#) module.
- class [SearchStubFactory](#)
A factory for creating stub instances of [Search](#) objects.
- class [CommandStub](#)
A stub version of the [Command](#) module.
- class [CommandStubFactory](#)
A factory for creating stub instances of [Command](#) objects.
- class [SymbolStub](#)
A stub version of the [Symbol](#) class.
- class [SymbolMgrStub](#)
A stub version of the [Symbol](#) Manager.
- class [SymbolStubFactory](#)
A factory for creating stub instances of [Symbol](#) Manager objects.
- class [CanvasStub](#)
A stub version of the [Canvas](#) module.
- class [CanvasStubFactory](#)
A factory for creating stub instances of [Canvas](#) objects.
- class [SecureStub](#)
A stub version of the [Secure](#) module.

- class [SecureStubFactory](#)
A factory for creating stub instances of Security objects.
- class [CronStub](#)
A stub version of the [Cron](#) module.
- class [CronStubFactory](#)
A factory for creating stub instances of [Cron](#) objects.
- class [InstallerStub](#)
A stub version of the [Installer](#) module.
- class [InstallerStubFactory](#)
A factory for creating stub instances of [Installer](#) objects.
- class [InterpreterStub](#)
A stub version of the [Interpreter](#) module.
- class [InterpreterStubFactory](#)
A factory for creating stub instances of [Interpreter](#) objects.
- class [ViciEditorStub](#)
A stub version of the [Vici Editor](#) module.
- class [ViciEditorStubFactory](#)
A factory for creating stub instances of [Vici Editor](#) objects.
- class [ViciStub](#)
A stub version of the [Vici](#) module.
- class [ViciStubFactory](#)
A factory for creating stub instances of [Vici](#) objects.

7.15.1 Detailed Description

The namespace for stub versions of modules.

7.16 VICI::Symbol Namespace Reference

An API for libsymbol.

Classes

- class [SymbolAttributes](#)
Holds the attributes of a symbol.
- class [TextAttributes](#)
Hold the attributes of a text comment.
- class [SymbolOwner](#)
Represents something that is notified about events occurring on a symbol.
- class [Symbol](#)
Represents something that is drawn on the canvas.
- class [SymbolClient](#)
An interface that is notified of changes to symbol manager.
- class [SymbolMgr](#)
The facade for the symbol library.
- class [SymbolFactory](#)
An abstract factory for making an instance of [SymbolMgr](#).

Typedefs

- typedef long [Colour](#)
Specialisation for holding colour values.
- typedef std::shared_ptr
< [SymbolFactory](#) > [SymbolFactoryPtr](#)
Shared pointer for [Symbol Factory](#).

Enumerations

- enum [Style](#) {
[SymCommand](#), [SymChoice](#), [SymFuncRef](#), [SymFunc](#),
[SymVar](#), [SymConst](#), [SymMutex](#), [SymSem](#),
[SymFile](#), [SymInline](#), [SymPipe](#), [SymDisplay](#),
[SymLock](#), [SymUnlock](#), [SymWait](#), [SymPost](#),
[SymFlow](#), [SymSuccess](#), [SymFail](#), [SymSignal](#),
[SymStdIn](#), [SymStdOut](#), [SymStdErr](#) }
The possible styles for symbols and lines.

7.16.1 Detailed Description

An API for libsymbols.

7.16.2 Enumeration Type Documentation

7.16.2.1 enum [VICI::Symbol::Style](#)

The possible styles for symbols and lines.

Enumerator

- SymCommand*** box for a command
- SymChoice*** diamond for a command
- SymFuncRef*** function call
- SymFunc*** entry point for a function
- SymVar*** variable
- SymConst*** constant
- SymMutex*** mutex variable
- SymSem*** semaphore variable
- SymFile*** file
- SymInline*** inline file
- SymPipe*** named pipe
- SymDisplay*** output display
- SymLock*** lock a mutex
- SymUnlock*** unlock a mutex
- SymWait*** wait for a semaphore
- SymPost*** signal a semaphore
- SymFlow*** flow of execution
- SymSuccess*** execution flow for a success result
- SymFail*** execution flow for a failure result

SymSignal send a signal
SymStdIn standard input stream
SymStdOut standard output stream
SymStdErr standard error stream

7.17 VICI::Syntax Namespace Reference

An API for libsyntax.

Classes

- class [Syntax](#)
Display a syntax chart of command options.
- class [SyntaxFactory](#)
An abstract factory for making an instance of [Syntax](#).

Typedefs

- typedef std::shared_ptr
< [SyntaxFactory](#) > [SyntaxFactoryPtr](#)
Shared pointer for [SyntaxFactory](#).

7.17.1 Detailed Description

An API for libsyntax.

Chapter 8

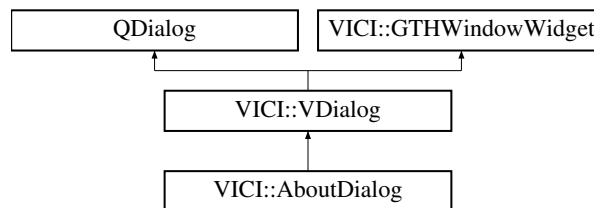
Class Documentation

8.1 VICI::AboutDialog Class Reference

A standard About [Vici](#) dialog for the project.

```
#include <vici/libgui.h>
```

Inheritance diagram for VICI::AboutDialog:



Public Member Functions

- [AboutDialog](#) (QWidget *parent, [csr](#) progName, [csr](#) description)
Constructor.
- virtual void [RegnFn](#) (std::function< void(void *, [gth::WidgetType](#), [csr](#))> reg)
This may be called by the test harness to get pointers to the individual widgets.

Additional Inherited Members

8.1.1 Detailed Description

A standard About [Vici](#) dialog for the project.

The documentation for this class was generated from the following files:

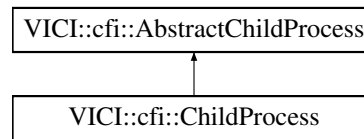
- [libgui.h](#)
- [libgui.cpp](#)

8.2 VICI::cfi::AbstractChildProcess Class Reference

Abstract child process interface for [ChildProcessMgr](#).

```
#include <vici/proc.h>
```

Inheritance diagram for VICI::cfi::AbstractChildProcess:



Public Member Functions

- virtual [~AbstractChildProcess](#) ()
Destructor.
- virtual int [getld](#) ()=0
Returns the pid of the child process.
- virtual void [kill](#) ()=0
Terminates the child process.
- virtual bool [done](#) ()=0
get the running state of the process.
- virtual void [finished](#) (int exit_status)=0
Called by the manager when the child completes.

8.2.1 Detailed Description

Abstract child process interface for [ChildProcessMgr](#).

The documentation for this class was generated from the following file:

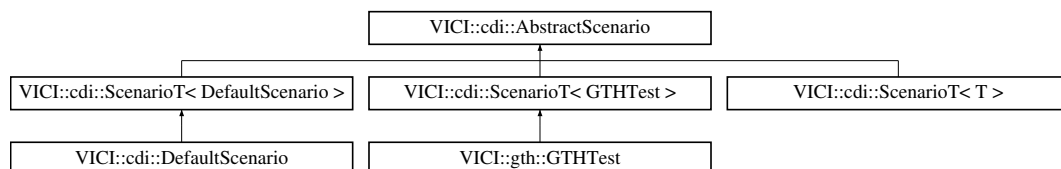
- [proc.h](#)

8.3 VICI::cdi::AbstractScenario Class Reference

Define a type for scenarios.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::AbstractScenario:



Public Member Functions

- [AbstractScenario](#) (csr name)
Constructor.
- virtual [~AbstractScenario](#) ()
Destructor.
- virtual bool [willRunTests](#) ()

- virtual void `runTests` (`csr scenarioName`, `ScenarioResults *`)

Runs the tests within the scenario.

Protected Attributes

- `std::string scenarioName`

The name of the scenario, as used when installed.

8.3.1 Detailed Description

Define a type for scenarios.

A scenario is responsible for setting up and taking down the resources required for a particular testing scenario.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 VICI::cdi::AbstractScenario::AbstractScenario (`csr name`) `[inline]`, `[explicit]`

Constructor.

Parameters

<code>name</code>	The name of the scenario.
-------------------	---------------------------

8.3.3 Member Function Documentation

8.3.3.1 virtual bool VICI::cdi::AbstractScenario::willRunTests () `[inline]`, `[virtual]`

Returns true if the scenario is going to manage the order in which the tests are run.

Reimplemented in `VICI::gth::GTHTest`.

The documentation for this class was generated from the following file:

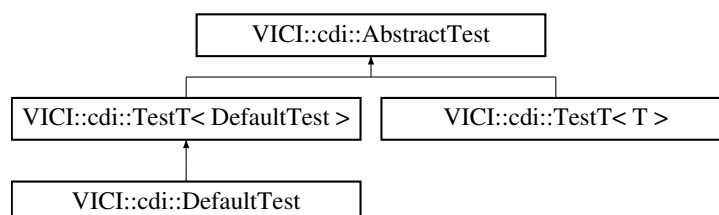
- `testmgr.h`

8.4 VICI::cdi::AbstractTest Class Reference

Responsible for managing resources needed for the entire test.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::AbstractTest:



Public Member Functions

- virtual [~AbstractTest](#) ()
Destructor.

8.4.1 Detailed Description

Responsible for managing resources needed for the entire test.

The documentation for this class was generated from the following file:

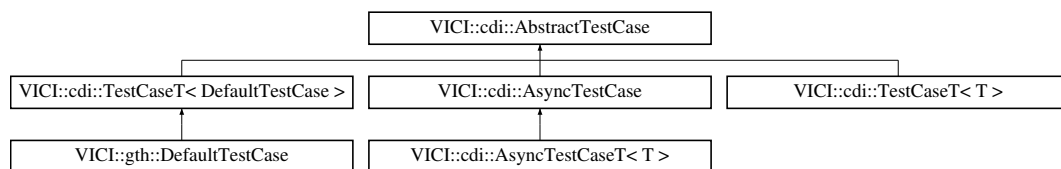
- [testmgr.h](#)

8.5 VICI::cdi::AbstractTestCase Class Reference

Base class for test cases.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::AbstractTestCase:



Public Member Functions

- [AbstractTestCase](#) (csr nm)
Constructor.
- virtual [~AbstractTestCase](#) ()
Destructor.
- virtual bool [operator\(\)](#) ([ScenarioResults](#) *)
Execute the tests.

Static Public Attributes

- static const bool [EXPECTED](#) = true
Value for expected parameter of [test\(\)](#)

Protected Member Functions

- bool [test](#) (bool cond, [csr](#) testMsg, bool expected=false)
Perform or record a test.
- virtual bool [runTest](#) ()=0
Run the tests for the test case.

Protected Attributes

- `std::string name`
The name of the test case.
- `ScenarioResults * scenario`
Results for the scenario are placed in here.

8.5.1 Detailed Description

Base class for test cases.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 VICI::cdi::AbstractTestCase::AbstractTestCase (`csr nm`) `[inline]`,`[explicit]`

Constructor.

Derived classes should perform any test case specific initialisations.

Parameters

<code>nm</code>	The name of the test case.
-----------------	----------------------------

8.5.2.2 virtual VICI::cdi::AbstractTestCase::~~AbstractTestCase () `[inline]`,`[virtual]`

Destructor.

Derived classes should perform test case specific clean up.

8.5.3 Member Function Documentation

8.5.3.1 bool AbstractTestCase::operator() (`ScenarioResults * sr`) `[virtual]`

Execute the tests.

Returns

true if tests passed.

8.5.3.2 virtual bool VICI::cdi::AbstractTestCase::runTest () `[protected]`,`[pure virtual]`

Run the tests for the test case.

Derived classes implement this to perform the required tests.

Returns

true if all tests passed.

Implemented in [VICI::gth::DefaultTestCase](#).

8.5.3.3 bool AbstractTestCase::test (`bool cond`, `csr testMsg`, `bool expected = false`) `[protected]`

Perform or record a test.

Parameters

<i>cond</i>	Set to true if test passes.
<i>testMsg</i>	Message to report when test fails
<i>expected</i>	Set to EXPECTED if test is expected to fail.

Returns

true if test passes, or failed and was expected to fail

The documentation for this class was generated from the following files:

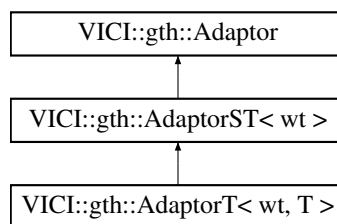
- [testmgr.h](#)
- [testmgr.cpp](#)

8.6 VICI::gth::Adaptor Class Reference

Defines an abstract base class for widget adaptors.

```
#include <vici/libgth.h>
```

Inheritance diagram for VICI::gth::Adaptor:



Public Member Functions

- [Adaptor](#) (*csr* name, [GTHTest](#) *gth)
Constructor.
- virtual [~Adaptor](#) ()
Destructor.
- virtual std::string [action](#) (*csr* tname, *csr* cmdnd, const [ParamList](#) ¶ms)=0
Interpreter function.
- virtual [VICI::gth::WidgetType](#) [getType](#) ()=0
Get the type of the adaptor.
- [csr](#) [getWidgetName](#) ()
Get the name of the widget.

Protected Member Functions

- void [mouseDoubleClick](#) (QWidget *widget, Qt::MouseButton button, Qt::KeyboardModifiers stateKey=0, QPoint pos=QPoint(), int delay=-1)
Utility functions.

Protected Attributes

- [GTHTest * gthTest](#)
Reference to the owner.
- `std::string` [widgetName](#)
The name of the widget as set when it was registered.

8.6.1 Detailed Description

Defines an abstract base class for widget adaptors.

The adaptor objects contain simple command interpreters that modify or access the contained widget (or other object)

8.6.2 Member Function Documentation

8.6.2.1 `virtual std::string VICI::gth::Adaptor::action (csr tname, csr cmd, const ParamList & params)` [pure virtual]

Interpreter function.

Parameters

<i>tname</i>	The name of the test case.
<i>cmd</i>	The type of action to perform
<i>params</i>	Any additional parameters for the command

Returns

Either OK or some value extracted from the widget

Implemented in [VICI::gth::AdaptorT< wt, T >](#).

8.6.2.2 `virtual VICI::gth::WidgetType VICI::gth::Adaptor::getType ()` [pure virtual]

Get the type of the adaptor.

Returns

The WidgetType

Implemented in [VICI::gth::AdaptorT< wt, T >](#).

The documentation for this class was generated from the following file:

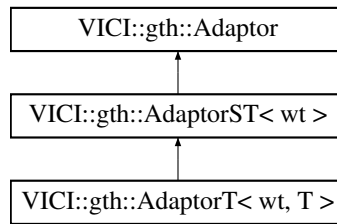
- [libgth.h](#)

8.7 VICI::gth::AdaptorST< wt > Class Template Reference

Adds a static method to [Adaptor](#) to describe it;

```
#include <vici/libgth.h>
```

Inheritance diagram for VICI::gth::AdaptorST< wt >:



Public Member Functions

- [AdaptorST](#) ([csr](#) name, [GTHTest](#) *gth)
Constructor.

Static Public Member Functions

- static void [describe](#) ([ListOfLists](#) &)
Get a list of commands and their parameters for the [Adaptor](#).

Additional Inherited Members

8.7.1 Detailed Description

```
template<VICI::gth::WidgetType wt>class VICI::gth::AdaptorST< wt >
```

Adds a static method to [Adaptor](#) to describe it;.

The documentation for this class was generated from the following file:

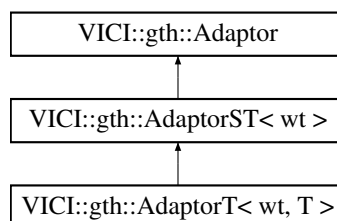
- [libgth.h](#)

8.8 VICI::gth::AdaptorT< wt, T > Class Template Reference

The interpreter for the script commands.

```
#include <libgth.h>
```

Inheritance diagram for VICI::gth::AdaptorT< wt, T >:



Public Member Functions

- [AdaptorT](#) (void *p, [csr](#) name, [GTHTest](#) *gth)
Constructor.
- [VICI::gth::WidgetType](#) [getType](#) ()

Get the type of the adaptor.

- virtual std::string [action](#) (csr tname, csr cmnd, const [ParamList](#) ¶ms)

Run the command on the [Adaptor's](#) widget.

Additional Inherited Members

8.8.1 Detailed Description

```
template<VICI::gth::WidgetType wt, class T>class VICI::gth::AdaptorT< wt, T >
```

The interpreter for the script commands.

class [AdaptorT](#) [libgth.h](#) [vici/libgth.h](#)

8.8.2 Constructor & Destructor Documentation

8.8.2.1 `template<VICI::gth::WidgetType wt, class T > VICI::gth::AdaptorT< wt, T >::AdaptorT (void * p, csr name, GTHTest * gth) [inline]`

Constructor.

Parameters

<i>p</i>	Pointer to a widget
<i>name</i>	The name of the widget.
<i>gth</i>	The Scenario object.

8.8.3 Member Function Documentation

8.8.3.1 `template<VICI::gth::WidgetType wt, class T > VICI::gth::WidgetType VICI::gth::AdaptorT< wt, T >::getType () [inline], [virtual]`

Get the type of the adaptor.

Returns

The WidgetType

Implements [VICI::gth::Adaptor](#).

The documentation for this class was generated from the following file:

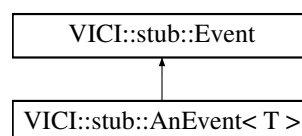
- [libgth.h](#)

8.9 VICI::stub::AnEvent< T > Class Template Reference

A template wrapper for function calls.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::AnEvent< T >:



Public Member Functions

- [AnEvent](#) (void(T::*f)(), T *x)
Constructor.
- virtual void [execute](#) ()
Execute the saved function pointer.

8.9.1 Detailed Description

```
template<class T>class VICI::stub::AnEvent< T >
```

A template wrapper for function calls.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 `template<class T > VICI::stub::AnEvent< T >::AnEvent (void(T::*)() f, T * x)` `[inline]`

Constructor.

Parameters

<i>f</i>	Pointer to member function.
<i>x</i>	Object to execute the method on.

The documentation for this class was generated from the following file:

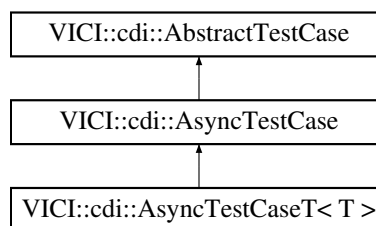
- [libifstubs.h](#)

8.10 VICI::cdi::AsyncTestCase Class Reference

Responsible for handling asynchronous tests.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::AsyncTestCase:



Public Member Functions

- [AsyncTestCase](#) (csr name)
Constructor.

Protected Member Functions

- virtual void [initTest](#) ()=0
Initiate the test case.

- virtual void `handleEvent (TestEvent *ev)=0`
Handle the events.
- virtual void `timedOut ()=0`
Handle time out.

Protected Attributes

- bool `done`
This should be set by the handleEvent function.
- std::chrono::steady_clock::time_point `startTime`
The time when the test is started.
- std::chrono::steady_clock::duration `timeOut`
The time to wait. The default is 10 seconds.

Additional Inherited Members

8.10.1 Detailed Description

Responsible for handling asynchronous tests.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 AsyncTestCase::AsyncTestCase (csr name)

Constructor.

Parameters

<i>name</i>	The name of the test case.
-------------	----------------------------

8.10.3 Member Function Documentation

8.10.3.1 virtual void VICI::cdi::AsyncTestCase::handleEvent (TestEvent * ev) [protected],[pure virtual]

Handle the events.

When the object under test gets its call back it will queue an event which is passed to this function. Set done to true when the last expected event arrives.

Parameters

<i>ev</i>	The test event.
-----------	-----------------

8.10.3.2 virtual void VICI::cdi::AsyncTestCase::initTest () [protected],[pure virtual]

Initiate the test case.

The derived class should use this to start the test function. It should return immediately the test has been started.

8.10.3.3 virtual void VICI::cdi::AsyncTestCase::timedOut () [protected],[pure virtual]

Handle time out.

This gets called if the events don't arrive in time.

The documentation for this class was generated from the following files:

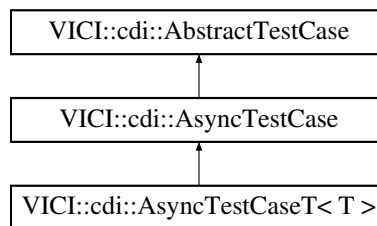
- [testmgr.h](#)
- [testmgr.cpp](#)

8.11 VICI::cdi::AsyncTestCaseT< T > Class Template Reference

Responsible for installing the factory for the test case type.

```
#include <testmgr.h>
```

Inheritance diagram for VICI::cdi::AsyncTestCaseT< T >:



Public Member Functions

- [AsyncTestCaseT](#) (`csr nm`)
Constructor.

Static Public Member Functions

- static void [install](#) (`csr nm`, `csr scn`)
Install the factory for the test case.

Additional Inherited Members

8.11.1 Detailed Description

```
template<class T>class VICI::cdi::AsyncTestCaseT< T >
```

Responsible for installing the factory for the test case type.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 `template<class T > VICI::cdi::AsyncTestCaseT< T >::AsyncTestCaseT (csr nm) [inline]`

Constructor.

Parameters

<i>nm</i>	The name of the test case
-----------	---------------------------

8.11.3 Member Function Documentation

8.11.3.1 `template<class T > static void VICI::cdi::AsyncTestCaseT< T >::install (csr nm, csr scn)` [inline],
[static]

Install the factory for the test case.

Parameters

<i>nm</i>	The name of the test case
<i>scn</i>	The name of the enclosing scenario.

The documentation for this class was generated from the following file:

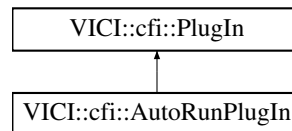
- [testmgr.h](#)

8.12 VICI::cfi::AutoRunPlugIn Class Reference

Base class for plug ins that are run immediately that the library is loaded.

```
#include <vici/plugin.h>
```

Inheritance diagram for VICI::cfi::AutoRunPlugIn:



Public Member Functions

- virtual void [execute](#) ()=0

Called by the [PlugInMgr](#) to run the plug-in's code.

8.12.1 Detailed Description

Base class for plug ins that are run immediately that the library is loaded.

The documentation for this class was generated from the following file:

- [plugin.h](#)

8.13 VICI::cdi::CallTrace Class Reference

Class to create a call trace.

```
#include <vici/trace.h>
```

Public Member Functions

- [CallTrace](#) (int level, [csr](#) methodName, [csr](#) file)

Constructor.

- [~CallTrace](#) ()

Destructor.

8.13.1 Detailed Description

Class to create a call trace.

This [CallTrace](#) class is used to produce a call graph. Use the macro `FN_TRACE` to create an object of this class on the stack at the beginning of each method. The destructor will be automatically called when the method ends.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 CallTrace::CallTrace (int level, csr methodName, csr file)

Constructor.

This should be used via the `FN_TRACE(level,method)` macro.

Parameters

<i>level</i>	used to determine if tracing output should be generated.
<i>methodName</i>	name of the function to report
<i>file</i>	name of the source file

The documentation for this class was generated from the following files:

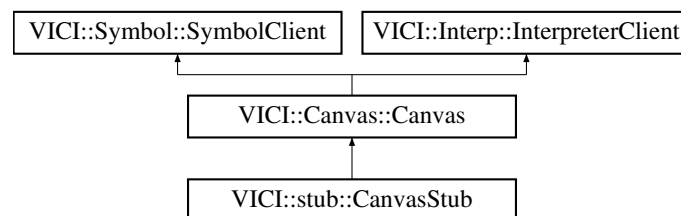
- [trace.h](#)
- [trace.cpp](#)

8.14 VICI::Canvas::Canvas Class Reference

The facade for canvas library.

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

Inheritance diagram for VICI::Canvas::Canvas:



Public Member Functions

- virtual `~Canvas ()`
virtual destructor
- virtual void `load (csr filename)=0`
restore the chart from the specified XML file.
- virtual void `save (csr filename)=0`
save the chart
- virtual void `setExecution (bool active, csr node)=0`
display of executing script
- virtual void `setCommand (NodeId nid, const ArgList &args)=0`
assigning a command to the current symbol
- virtual void `selection (Symbol::Symbol *sym)=0`

- notification that a symbol has been selected*

 - virtual void `symbolAttr (Symbol::SymbolAttributes &att)=0`

notification that the default symbol attributes have changed
- virtual void `textAttr (Symbol::TextAttributes &att)=0`

notification of new text attributes
- virtual `ArgList getNames ()=0`

get a list of names of variables etc for use in command options

8.14.1 Detailed Description

The facade for canvas library.

The implementation of this class will provide a canvas on which the user constructs a flow chart.

8.14.2 Member Function Documentation

8.14.2.1 virtual void VICI::Canvas::Canvas::load (*csr filename*) [pure virtual]

restore the chart from the specified XML file.

Parameters

<i>filename</i>	the XML VICI script file.
-----------------	---

Implemented in [VICI::stub::CanvasStub](#).

8.14.2.2 virtual void VICI::Canvas::Canvas::save (*csr filename*) [pure virtual]

save the chart

Parameters

<i>filename</i>	the XML VICI script file.
-----------------	---

Implemented in [VICI::stub::CanvasStub](#).

8.14.2.3 virtual void VICI::Canvas::Canvas::selection ([Symbol::Symbol](#) * *sym*) [pure virtual]

notification that a symbol has been selected

Parameters

<i>sym</i>	the symbol to place on the canvas
------------	-----------------------------------

Implements [VICI::Symbol::SymbolClient](#).

Implemented in [VICI::stub::CanvasStub](#).

8.14.2.4 virtual void VICI::Canvas::Canvas::setCommand ([Nodeld](#) *nid*, const [ArgList](#) & *args*) [pure virtual]

assigning a command to the current symbol

Parameters

<i>nid</i>	The node to set the args of.
------------	------------------------------

<i>args</i>	The command (arg[0]) and options.
-------------	-----------------------------------

Implemented in [VICI::stub::CanvasStub](#).

8.14.2.5 virtual void VICI::Canvas::Canvas::setExecution (bool *active*, csr *node*) [pure virtual]

display of executing script

Parameters

<i>active</i>	true if the command is being executed
<i>node</i>	the node being executed

Todo node should be NodeId type

Implemented in [VICI::stub::CanvasStub](#).

8.14.2.6 virtual void VICI::Canvas::Canvas::symbolAttr (Symbol::SymbolAttributes & *att*) [pure virtual]

notification that the default symbol attributes have changed

Parameters

<i>att</i>	the new default attributes
------------	----------------------------

Todo this appears to be meaningless without the symbol

Implements [VICI::Symbol::SymbolClient](#).

Implemented in [VICI::stub::CanvasStub](#).

8.14.2.7 virtual void VICI::Canvas::Canvas::textAttr (Symbol::TextAttributes & *att*) [pure virtual]

notification of new text attributes

Parameters

<i>att</i>	the new text attributes
------------	-------------------------

Implements [VICI::Symbol::SymbolClient](#).

Implemented in [VICI::stub::CanvasStub](#).

The documentation for this class was generated from the following file:

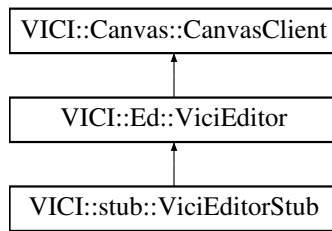
- [vici.h](#)

8.15 VICI::Canvas::CanvasClient Class Reference

An interface that is notified of canvas actions.

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

Inheritance diagram for VICI::Canvas::CanvasClient:



Public Member Functions

- virtual `~CanvasClient ()`
virtual destructor
- virtual void `newSymbol (Symbol::Symbol *sym)=0`
called when a symbol is placed on the canvas.
- virtual void `newChart ()=0`
called when a new diagram has been started
- virtual void `changedChart ()=0`
called when the chart is first changed
- virtual void `searchAction (NodeId)=0`
notification that the user wants the search tab
- virtual void `commandAction (NodeId, const ArgList &)=0`
notification that the command tab is to be displayed
- virtual void `breakAction (NodeId, bool set)=0`
notification that a breakpoint has been set or cleared

8.15.1 Detailed Description

An interface that is notified of canvas actions.

This interface is implemented by objects that need to be notified of actions by the canvas.

8.15.2 Member Function Documentation

8.15.2.1 virtual void VICI::Canvas::CanvasClient::newSymbol (Symbol::Symbol * sym) [pure virtual]

called when a symbol is placed on the canvas.

Parameters

<code>sym</code>	the symbol that was placed on the canvas.
------------------	---

Implemented in [VICI::stub::ViciEditorStub](#).

The documentation for this class was generated from the following file:

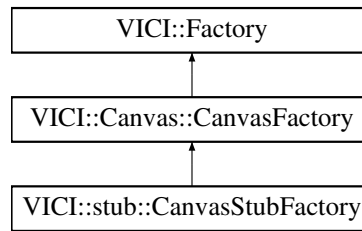
- [vici.h](#)

8.16 VICI::Canvas::CanvasFactory Class Reference

An abstract factory for making an instance of [Canvas](#).

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

Inheritance diagram for VICI::Canvas::CanvasFactory:



Public Member Functions

- virtual `~CanvasFactory ()`
virtual destructor
- virtual `Canvas * makeCanvas (Window *, Symbol::SymbolMgr *, Interp::Interpreter *, CanvasClient *)=0`
create an instance of the Canvas class

8.16.1 Detailed Description

An abstract factory for making an instance of `Canvas`.

The implementation will create either the production version, or a stub, or a test version of the `Canvas` class.

The documentation for this class was generated from the following file:

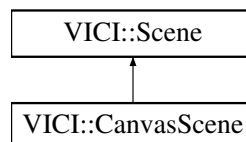
- [vici.h](#)

8.17 VICI::CanvasScene Class Reference

An implementation of the `Scene` abstract class for holding a `QGraphicsScene`.

```
#include <vici/canvas.h>
```

Inheritance diagram for `VICI::CanvasScene`:



Public Member Functions

- `CanvasScene (QGraphicsScene *s)`
constructor
- `operator QGraphicsScene * ()`
type conversion operator
- `QGraphicsScene * operator-> ()`
pointer operator

Protected Attributes

- `QGraphicsScene * scene`
the graphics scene being handled.

8.17.1 Detailed Description

An implementation of the [Scene](#) abstract class for holding a QGraphicsScene.

This class implements the [Scene](#) type for the QGraphicsScene. It allows references to scenes to be passed between modules without the Qt libraries leaking into the general interface definitions defined in [vici.h](#).

8.17.2 Constructor & Destructor Documentation

8.17.2.1 VICI::CanvasScene::CanvasScene (QGraphicsScene * s) [inline]

constructor

Parameters

s	the graphics scene object.
---	----------------------------

The documentation for this class was generated from the following file:

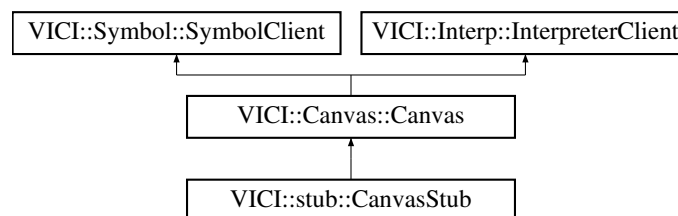
- [canvas.h](#)

8.18 VICI::stub::CanvasStub Class Reference

A stub version of the [Canvas](#) module.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::CanvasStub:



Public Member Functions

- [CanvasStub](#) ([Window](#) *w, [Symbol::SymbolMgr](#) *s, [Interp::Interpreter](#) *i, [VICI::Canvas::CanvasClient](#) *c)
Constructor.
- virtual void [load](#) ([csr](#) filename)
restore the chart from the specified XML file.
- virtual void [save](#) ([csr](#) filename)
save the chart
- virtual void [setExecution](#) (bool active, [csr](#) node)
display of executing script
- virtual void [setCommand](#) ([Nodeld](#), const [ArgList](#) &)
assigning a command to the current symbol
- virtual void [selection](#) ([VICI::Symbol::Symbol](#) *)
notification that a symbol has been selected
- virtual void [symbolAttr](#) ([VICI::Symbol::SymbolAttributes](#) &)
notification that the default symbol attributes have changed
- virtual void [textAttr](#) ([VICI::Symbol::TextAttributes](#) &)

- notification of new text attributes*
- virtual [VICI::ArgList](#) `getNames ()`
get a list of names of variables etc for use in command options
- virtual void `setValue (csr varName, csr value)`
this gets called when a script variable has a change of value
- virtual void `setFile (int state, csr filename)`
this gets called when a monitored file changes state.
- virtual void `setCursor (ThreadId, Nodeld)`
this gets called as the shell steps through the script
- virtual void `breakReached (ThreadId, Nodeld)`
this gets called when a break point is reached
- virtual void `dataReady (Nodeld)`
this gets called when data is available on a display
- virtual void `reportError (Severity, csr)`
This gets called if an error is detected.
- virtual void `done ()`
this gets called when the script completes

8.18.1 Detailed Description

A stub version of the [Canvas](#) module.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 `CanvasStub::CanvasStub (Window * w, Symbol::SymbolMgr * s, Interp::Interpreter * i, VICI::Canvas::CanvasClient * c)`

Constructor.

Parameters

<i>w</i>	The window to display into
<i>s</i>	The symbol manager responsible for the symbol palette
<i>i</i>	The Vici interpreter.
<i>c</i>	The owner of the canvas.

8.18.3 Member Function Documentation

8.18.3.1 `void CanvasStub::breakReached (ThreadId tid, Nodeld node) [virtual]`

this gets called when a break point is reached

Parameters

<i>tid</i>	the thread which reached the breakpoint
<i>node</i>	the location of the breakpoint

Implements [VICI::Interp::InterpreterClient](#).

8.18.3.2 `void CanvasStub::dataReady (Nodeld node) [virtual]`

this gets called when data is available on a display

Parameters

<i>node</i>	the node of the display object in the flowchart
-------------	---

Implements [VICI::Interp::InterpreterClient](#).

8.18.3.3 void CanvasStub::load (*csr filename*) [virtual]

restore the chart from the specified XML file.

Parameters

<i>filename</i>	the XML VICI script file.
-----------------	---

Implements [VICI::Canvas::Canvas](#).

8.18.3.4 void CanvasStub::reportError (Severity *sev*, *csr msg*) [virtual]

This gets called if an error is detected.

Parameters

<i>msg</i>	The text of the error message.
<i>sev</i>	The severity of the error

Implements [VICI::Interp::InterpreterClient](#).

8.18.3.5 void CanvasStub::save (*csr filename*) [virtual]

save the chart

Parameters

<i>filename</i>	the XML VICI script file.
-----------------	---

Implements [VICI::Canvas::Canvas](#).

8.18.3.6 void CanvasStub::selection (VICI::Symbol::Symbol * *sym*) [virtual]

notification that a symbol has been selected

Parameters

<i>sym</i>	the symbol to place on the canvas
------------	-----------------------------------

Implements [VICI::Canvas::Canvas](#).

8.18.3.7 void CanvasStub::setCommand (Nodeld *nid*, const ArgList & *args*) [virtual]

assigning a command to the current symbol

Parameters

<i>nid</i>	The node to set the args of.
<i>args</i>	The command (arg[0]) and options.

Implements [VICI::Canvas::Canvas](#).

8.18.3.8 void CanvasStub::setCursor (ThreadId *tid*, Nodeld *node*) [virtual]

this gets called as the shell steps through the script

Parameters

<i>tid</i>	the thread which changed to executing this node
<i>node</i>	the node that is currently being executed

Implements [VICI::Interp::InterpreterClient](#).

8.18.3.9 void CanvasStub::setExecution (bool *active*, *csr node*) [virtual]

display of executing script

Parameters

<i>active</i>	true if the command is being executed
<i>node</i>	the node being executed

Todo node should be NodeId type

Implements [VICI::Canvas::Canvas](#).

8.18.3.10 void CanvasStub::setFile (int *state*, *csr filename*) [virtual]

this gets called when a monitored file changes state.

Parameters

<i>state</i>	this needs to be defined
<i>filename</i>	the name of the file that changed

Implements [VICI::Interp::InterpreterClient](#).

8.18.3.11 void CanvasStub::setValue (*csr varName*, *csr value*) [virtual]

this gets called when a script variable has a change of value

Parameters

<i>varName</i>	the name of the variable that changed value
<i>value</i>	the new value of the variable

Implements [VICI::Interp::InterpreterClient](#).

8.18.3.12 void CanvasStub::symbolAttr (VICI::Symbol::SymbolAttributes & *att*) [virtual]

notification that the default symbol attributes have changed

Parameters

<i>att</i>	the new default attributes
------------	----------------------------

Todo this appears to be meaningless without the symbol

Implements [VICI::Canvas::Canvas](#).

8.18.3.13 void CanvasStub::textAttr (VICI::Symbol::TextAttributes & *att*) [virtual]

notification of new text attributes

Parameters

<i>att</i>	the new text attributes
------------	-------------------------

Implements [VICI::Canvas::Canvas](#).

The documentation for this class was generated from the following files:

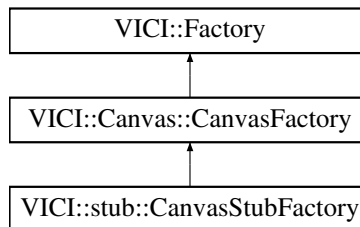
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.19 VICI::stub::CanvasStubFactory Class Reference

A factory for creating stub instances of [Canvas](#) objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::CanvasStubFactory:



Additional Inherited Members

8.19.1 Detailed Description

A factory for creating stub instances of [Canvas](#) objects.

The documentation for this class was generated from the following files:

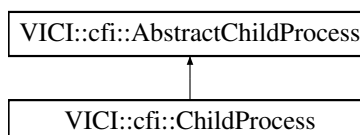
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.20 VICI::cfi::ChildProcess Class Reference

Represents the state of a child process.

```
#include <vici/proc.h>
```

Inheritance diagram for VICI::cfi::ChildProcess:



Public Member Functions

- [ChildProcess](#) (const std::string &cmdnd, int *fdin=nullptr, int *fdout=nullptr, int *fderr=nullptr)

- constructor*
- [ChildProcess](#) ()
- constructor for fork*
- virtual [~ChildProcess](#) ()
- destructor*
- virtual void [setArgs](#) (const std::vector< std::string > &args)
- set the args for the process*
- virtual void [run](#) ()
- start the command running*
- virtual void [kill](#) ()
- terminate the process*
- virtual int [signal](#) (int sig)
- send a signal to the process*
- virtual void [finished](#) (int result)
- set the exit status of the process*
- bool [done](#) ()
- get the running status of the program*
- virtual int [getid](#) ()
- get the process id*
- int [getExitSignal](#) ()
- get the signal that caused the process to exit.*
- int [getExitStatus](#) ()
- get the exit status*
- void [reportExitErrors](#) (bool x)
- turn on or off the reporting of error exits*

8.20.1 Detailed Description

Represents the state of a child process.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 [ChildProcess::ChildProcess](#) (const std::string & *cmd*, int * *fdin* = nullptr, int * *fdout* = nullptr, int * *fderr* = nullptr)

constructor

Parameters

<i>cmd</i>	the command to run
<i>fdin</i>	File descriptor for stdin for the child
<i>fdout</i>	File descriptor for stdout for the child
<i>fderr</i>	File descriptor for stderr for the child

8.20.3 Member Function Documentation

8.20.3.1 [void ChildProcess::finished](#) (int *result*) [virtual]

set the exit status of the process

called by the process manager when the process exits

Parameters

<i>result</i>	the exit status and code of the process
---------------	---

Implements [VICI::cfi::AbstractChildProcess](#).

8.20.3.2 void ChildProcess::setArgs (const std::vector< std::string > & args) [virtual]

set the args for the process

Parameters

<i>args</i>	the args for the command
-------------	--------------------------

The documentation for this class was generated from the following files:

- [proc.h](#)
- [proc.cpp](#)

8.21 VICI::cfi::ChildProcessMgr Class Reference

Manage the child processes.

```
#include <vici/proc.h>
```

Public Member Functions

- void [registerOwner](#) ([ProcessOwner](#) *)
register an object that is to be notified of a child's death
- void [registerChild](#) ([AbstractChildProcess](#) *cp)
tell the manager about a child
- void [deregisterChild](#) ([AbstractChildProcess](#) *cp)
forget about a child
- int [numberOfChildren](#) () const
get the number of child processes
- int [numberOfLiveChildren](#) () const
get the number of running children
- void [shutDown](#) ()
terminate all the child processes

Static Public Member Functions

- static [ChildProcessMgr](#) & [instance](#) ()
get reference to the child process manager

8.21.1 Detailed Description

Manage the child processes.

Manage the child processes at a global level which is mostly about handling the SIGCHLD signals and ensuring the correct [ChildProcess](#) gets notified.

8.21.2 Member Function Documentation

8.21.2.1 void ChildProcessMgr::deregisterChild (AbstractChildProcess * cp)

forget about a child

called by the [ChildProcess](#) as it destructs

Parameters

<i>cp</i>	the child process
-----------	-------------------

8.21.2.2 int VICI::cfi::ChildProcessMgr::numberOfChildren () const [inline]

get the number of child processes

Returns

the number of child processes

8.21.2.3 int ChildProcessMgr::numberOfLiveChildren () const

get the number of running children

Returns

the number of children still running

8.21.2.4 void ChildProcessMgr::registerChild (AbstractChildProcess * cp)

tell the manager about a child

called by the [ChildProcess](#) as it creates the process

Parameters

<i>cp</i>	the child process
-----------	-------------------

The documentation for this class was generated from the following files:

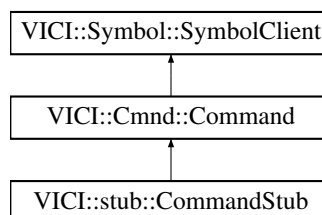
- [proc.h](#)
- [proc.cpp](#)

8.22 VICI::Cmnd::Command Class Reference

The facade for the command library.

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

Inheritance diagram for VICI::Cmnd::Command:



Public Member Functions

- virtual `~Command ()`
virtual destructor
- virtual void `show ()=0`
display the command entry window
- virtual void `setCommand (NodeId nid, const ArgList &args)=0`
set the initial value of the command
- virtual void `selection (Symbol::Symbol *sym)=0`
notification of new current symbol
- virtual void `symbolAttr (Symbol::SymbolAttributes &)=0`
unused by this library
- virtual void `textAttr (Symbol::TextAttributes &)=0`
unused by this library

8.22.1 Detailed Description

The facade for the command library.

The implementation of this class will display a window in which the user will select a command and set its options and arguments.

8.22.2 Member Function Documentation

8.22.2.1 virtual void VICI::Cmnd::Command::selection (Symbol::Symbol * sym) [pure virtual]

notification of new current symbol

Parameters

<i>sym</i>	the symbol that the user just selected
------------	--

Implements [VICI::Symbol::SymbolClient](#).

Implemented in [VICI::stub::CommandStub](#).

8.22.2.2 virtual void VICI::Cmnd::Command::setCommand (NodeId nid, const ArgList & args) [pure virtual]

set the initial value of the command

Parameters

<i>nid</i>	The identifier for the current node
<i>args</i>	The command (args[0]) and options.

Implemented in [VICI::stub::CommandStub](#).

The documentation for this class was generated from the following file:

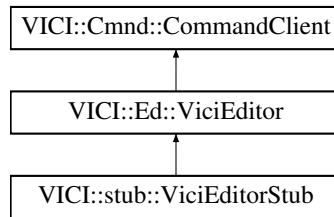
- [vici.h](#)

8.23 VICI::Cmnd::CommandClient Class Reference

An interface that is notified of a command selection.

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

Inheritance diagram for VICI::Cmnd::CommandClient:



Public Member Functions

- virtual `~CommandClient()`
virtual destructor
- virtual void `optionsAndParameters(NodeId nid, const ArgList &args)=0`
this gets called when the user presses OK
- virtual void `cmdnError(csr msg)=0`
this gets called if there is an error

8.23.1 Detailed Description

An interface that is notified of a command selection.

This interface is implemented by objects that need to be notified when the user selects a command and its options.

8.23.2 Member Function Documentation

8.23.2.1 virtual void VICI::Cmnd::CommandClient::cmdnError (csr msg) [pure virtual]

this gets called if there is an error

Parameters

<i>msg</i>	the text of the error message
------------	-------------------------------

Implemented in [VICI::stub::ViciEditorStub](#).

8.23.2.2 virtual void VICI::Cmnd::CommandClient::optionsAndParameters (NodeId nid, const ArgList & args) [pure virtual]

this gets called when the user presses OK

Parameters

<i>args</i>	the selected command and its options are returned in this
<i>nid</i>	The identifier for the current node

Implemented in [VICI::stub::ViciEditorStub](#).

The documentation for this class was generated from the following file:

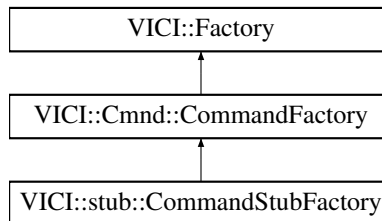
- [vici.h](#)

8.24 VICI::Cmnd::CommandFactory Class Reference

An abstract factory for making an instance of [Command](#).

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

Inheritance diagram for VICI::Cmnd::CommandFactory:



Public Member Functions

- virtual [~CommandFactory](#) ()
virtual destructor
- virtual [Command](#) * [makeCommand](#) ([Window](#) *, [CommandClient](#) *, [Canvas::Canvas](#) *)=0
create an instance of the [Command](#) class

8.24.1 Detailed Description

An abstract factory for making an instance of [Command](#).

The implementation will create either the production version, or a stub, or a test version of the [Command](#) class.

The documentation for this class was generated from the following file:

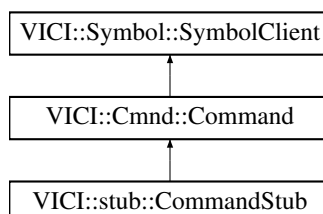
- [vici.h](#)

8.25 VICI::stub::CommandStub Class Reference

A stub version of the [Command](#) module.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::CommandStub:



Public Member Functions

- [CommandStub](#) ([Window](#) *w, [Cmnd::CommandClient](#) *c, [Canvas::Canvas](#) *)
Constructor.
- virtual void [show](#) ()

display the command entry window

- virtual void [setCommand](#) ([NodeId](#) nid, const [ArgList](#) &args)

set the initial value of the command

- virtual void [selection](#) ([Symbol::Symbol](#) *)

notification of new current symbol

- virtual void [symbolAttr](#) ([Symbol::SymbolAttributes](#) &)

unused by this library

- virtual void [textAttr](#) ([Symbol::TextAttributes](#) &)

unused by this library

8.25.1 Detailed Description

A stub version of the Command module.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 [CommandStub::CommandStub](#) ([Window](#) * w, [Cmnd::CommandClient](#) * c, [Canvas::Canvas](#) *)

Constructor.

Parameters

w	The window to display into.
c	The module that gets the results.

8.25.3 Member Function Documentation

8.25.3.1 void [CommandStub::selection](#) ([Symbol::Symbol](#) * sym) [virtual]

notification of new current symbol

Parameters

sym	the symbol that the user just selected
-----	--

Implements [VICI::Cmnd::Command](#).

8.25.3.2 void [CommandStub::setCommand](#) ([NodeId](#) nid, const [ArgList](#) & args) [virtual]

set the initial value of the command

Parameters

nid	The identifier for the current node
args	The command (args[0]) and options.

Implements [VICI::Cmnd::Command](#).

The documentation for this class was generated from the following files:

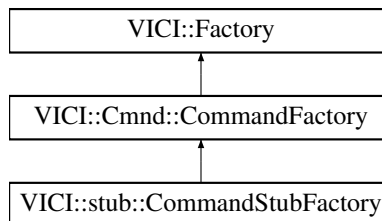
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.26 VICI::stub::CommandStubFactory Class Reference

A factory for creating stub instances of Command objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::CommandStubFactory:



Public Member Functions

- virtual `Cmnd::Command *` `makeCommand` (`Window *`, `Cmnd::CommandClient *`, `Canvas::Canvas *`)
create an instance of the Command class

8.26.1 Detailed Description

A factory for creating stub instances of Command objects.

The documentation for this class was generated from the following files:

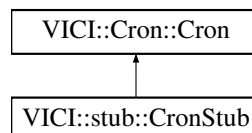
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.27 VICI::Cron::Cron Class Reference

Allow scripts to be scheduled for later running.

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

Inheritance diagram for VICI::Cron::Cron:



Public Member Functions

- virtual `~Cron` ()
virtual destructor
- virtual void `setCurrentFile` (`csr filename`)=0
Set the file to schedule.
- virtual void `show` ()=0
Display the window.

8.27.1 Detailed Description

Allow scripts to be scheduled for later running.

The facade for the cron library. An implementation will allow the user to schedule the running of scripts which can execute without human interaction.

8.27.2 Member Function Documentation

8.27.2.1 virtual void VICI::Cron::Cron::setCurrentFile (*csr filename*) [pure virtual]

Set the file to schedule.

Parameters

<i>filename</i>	the script to schedule.
-----------------	-------------------------

Implemented in [VICI::stub::CronStub](#).

The documentation for this class was generated from the following file:

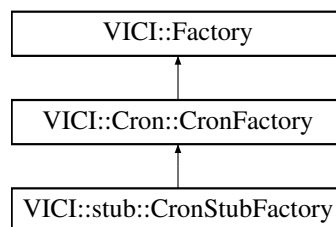
- [vici.h](#)

8.28 VICI::Cron::CronFactory Class Reference

An abstract factory for making an instance of [Cron](#).

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

Inheritance diagram for VICI::Cron::CronFactory:



Public Member Functions

- virtual [~CronFactory](#) ()
virtual destructor
- virtual [Cron](#) * [makeCron](#) ([Window](#) *w)=0
create an instance of the [Cron](#) class

8.28.1 Detailed Description

An abstract factory for making an instance of [Cron](#).

The implementation will create either the production version, or a stub, or a test version of the [Cron](#) class.

8.28.2 Member Function Documentation

8.28.2.1 `virtual Cron* VICI::Cron::CronFactory::makeCron (Window * w)` [pure virtual]

create an instance of the [Cron](#) class

Parameters

<i>w</i>	the window to display into.
----------	-----------------------------

Returns

an instance of the [Cron](#) object

Implemented in [VICI::stub::CronStubFactory](#).

The documentation for this class was generated from the following file:

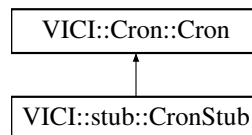
- [vici.h](#)

8.29 VICI::stub::CronStub Class Reference

A stub version of the [Cron](#) module.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::CronStub:



Public Member Functions

- [CronStub](#) ([Window](#) *w)
Constructor.
- virtual void [setCurrentFile](#) ([csr](#) filename)
Set the file to schedule.
- virtual void [show](#) ()
Display the window.

8.29.1 Detailed Description

A stub version of the [Cron](#) module.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 CronStub::CronStub (Window * w)

Constructor.

Parameters

<i>w</i>	The window to display into.
----------	-----------------------------

8.29.3 Member Function Documentation

8.29.3.1 void CronStub::setCurrentFile (*csr filename*) [virtual]

Set the file to schedule.

Parameters

<i>filename</i>	the script to schedule.
-----------------	-------------------------

Implements [VICI::Cron::Cron](#).

The documentation for this class was generated from the following files:

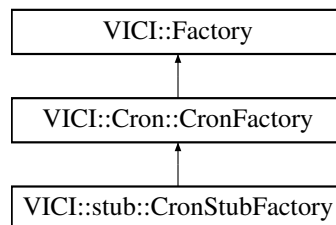
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.30 VICI::stub::CronStubFactory Class Reference

A factory for creating stub instances of [Cron](#) objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::CronStubFactory:



Public Member Functions

- virtual [Cron::Cron](#) * [makeCron](#) ([Window](#) *)
create an instance of the [Cron](#) class

8.30.1 Detailed Description

A factory for creating stub instances of [Cron](#) objects.

8.30.2 Member Function Documentation

8.30.2.1 [Cron::Cron](#) * [CronStubFactory::makeCron](#) ([Window](#) * *w*) [virtual]

create an instance of the [Cron](#) class

Parameters

<i>w</i>	the window to display into.
----------	-----------------------------

Returns

an instance of the [Cron](#) object

Implements [VICI::Cron::CronFactory](#).

The documentation for this class was generated from the following files:

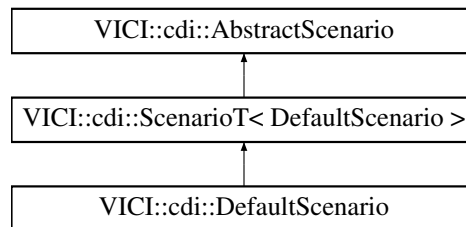
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.31 VICI::cdi::DefaultScenario Class Reference

Provide a default scenario object.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::DefaultScenario:



Public Member Functions

- [DefaultScenario](#) (*csr* name)
Constructor.
- [DefaultScenario](#) ()
Constructor.
- [~DefaultScenario](#) ()
Destructor.

Additional Inherited Members

8.31.1 Detailed Description

Provide a default scenario object.

The documentation for this class was generated from the following file:

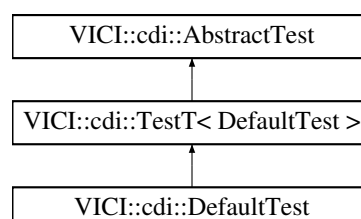
- [testmgr.h](#)

8.32 VICI::cdi::DefaultTest Class Reference

Provide a default version of the [AbstractTest](#) object.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::DefaultTest:



Public Member Functions

- [DefaultTest \(\)](#)
Default constructor that does nothing.
- [~DefaultTest \(\)](#)
Destructor.

Additional Inherited Members

8.32.1 Detailed Description

Provide a default version of the [AbstractTest](#) object.

The documentation for this class was generated from the following file:

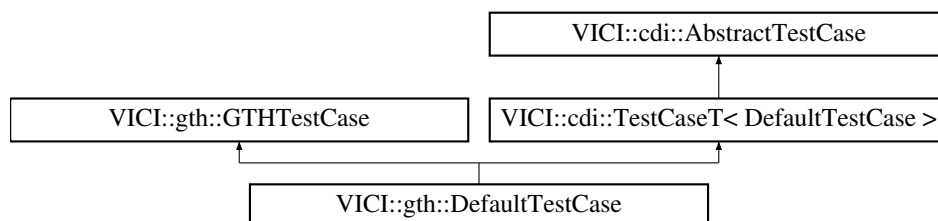
- [testmgr.h](#)

8.33 VICI::gth::DefaultTestCase Class Reference

Provides a test case to use for actions that don't have an explicit test case.

```
#include <vici/libgth.h>
```

Inheritance diagram for VICI::gth::DefaultTestCase:



Public Member Functions

- [DefaultTestCase \(csr nm\)](#)
Constructor.
- [~DefaultTestCase \(\)](#)
Destructor.

Protected Member Functions

- virtual bool [runTest \(\)](#)
Run the tests for the test case.

Additional Inherited Members

8.33.1 Detailed Description

Provides a test case to use for actions that don't have an explicit test case.

8.33.2 Member Function Documentation

8.33.2.1 `virtual bool VICI::gth::DefaultTestCase::runTest () [inline],[protected],[virtual]`

Run the tests for the test case.

Derived classes implement this to perform the required tests.

Returns

true if all tests passed.

Implements [VICI::cdi::AbstractTestCase](#).

The documentation for this class was generated from the following file:

- [libgth.h](#)

8.34 VICI::cfi::Discoverable Class Reference

A mixin class that makes its owner discoverable.

```
#include <vici/discover.h>
```

Protected Attributes

- [DiscoverSharedPointer discover](#)

A shared pointer that references back to the discoverable object.

8.34.1 Detailed Description

A mixin class that makes its owner discoverable.

Adds a shared pointer to a [DiscoverPointer](#) to an object so that it can be discovered.

The documentation for this class was generated from the following file:

- [discover.h](#)

8.35 VICI::cfi::DiscoverPointer Struct Reference

Holds a pointer to a discoverable object.

```
#include <vici/discover.h>
```

Public Member Functions

- [DiscoverPointer \(\)](#)

Default constructor.

- [DiscoverPointer \(void *a\)](#)

Constructor.

Public Attributes

- void * [addr](#)

Pointer to a discoverable object.

8.35.1 Detailed Description

Holds a pointer to a discoverable object.

A discoverable object holds a shared pointer to this object. The DiscoveryManager holds a weak pointer to this object so that it can tell when the owner has been deleted.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 VICI::cfi::DiscoverPointer::DiscoverPointer (void * a) [inline]

Constructor.

Parameters

<i>a</i>	The address of the discoverable object.
----------	---

The documentation for this struct was generated from the following file:

- [discover.h](#)

8.36 VICI::cfi::DiscoveryMgr Class Reference

Manager for discoverable objects.

```
#include <vici/discover.h>
```

Public Member Functions

- void [save](#) (const char *prettyName, [DiscoverSharedPointer](#) p)

Save a discoverable object.

- void [fetch](#) ([csr](#) name, std::vector< void * > &results)

Get a list of objects with a class name.

Static Public Member Functions

- static [DiscoveryMgr](#) & [instance](#) ()

Return a reference to the [DiscoveryMgr](#).

8.36.1 Detailed Description

Manager for discoverable objects.

The manager keeps a list of discoverable objects and is responsible for supplying a list of objects given a class name.

8.36.2 Member Function Documentation

8.36.2.1 void DiscoveryMgr::fetch (*csr name*, std::vector< void * > & *results*)

Get a list of objects with a class name.

Parameters

<i>name</i>	The class name to search for.
<i>results</i>	A vector which is filled in with pointers to the objects.

8.36.2.2 void DiscoveryMgr::save (const char * *prettyName*, DiscoverSharedPointer *p*)

Save a discoverable object.

Parameters

<i>prettyName</i>	The name of the constructor as provided by PRETTY_FUNCTION
<i>p</i>	A shared pointer to the discoverable object.

The documentation for this class was generated from the following files:

- [discover.h](#)
- [discover.cpp](#)

8.37 VICI::stub::Dispatcher Class Reference

Send events to registered objects.

```
#include <vici/libifstubs.h>
```

Public Member Functions

- void [registerEvent](#) (csr useCase, csr event, [Event](#) *fn)
Register a method to be called.
- void [sendEvent](#) (csr useCase, csr event)
Make a registered call.
- void [clearTrace](#) ()
Clear the trace data.
- void [enableTracing](#) (bool x)
Enable tracing.
- void [trace](#) (csr x)
Record a trace message.
- std::vector< std::string > & [getTrace](#) ()
Get the current trace data.

Static Public Member Functions

- static [Dispatcher](#) & [instance](#) ()
Get an instance of the singleton object.

8.37.1 Detailed Description

Send events to registered objects.

The [Dispatcher](#) enables us to make method calls on objects using strings to identify them, thus separating the caller from any dependency on the callee.

The [Dispatcher](#) also collects trace information so that the test harness can verify the correct sequence of method calls.

8.37.2 Member Function Documentation

8.37.2.1 void VICI::stub::Dispatcher::enableTracing (bool x) [inline]

Enable tracing.

Parameters

x	Set to true to enable tracing.
---	--------------------------------

8.37.2.2 std::vector< std::string >& VICI::stub::Dispatcher::getTrace () [inline]

Get the current trace data.

Returns

a reference to the trace data.

8.37.2.3 Dispatcher & Dispatcher::instance () [static]

Get an instance of the singleton object.

Returns

A reference to the [Dispatcher](#)

8.37.2.4 void Dispatcher::registerEvent (csr useCase, csr event, Event * fn)

Register a method to be called.

Parameters

<i>useCase</i>	The name of the use case.
<i>event</i>	The name of the event within the use case.
<i>fn</i>	The function to call.

8.37.2.5 void Dispatcher::sendEvent (csr useCase, csr event)

Make a registered call.

Parameters

<i>useCase</i>	The name of the use case.
<i>event</i>	the name of the event within the use case.

8.37.2.6 void VICI::stub::Dispatcher::trace (csr x) [inline]

Record a trace message.

Parameters

x	The message to record.
---	------------------------

The documentation for this class was generated from the following files:

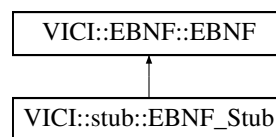
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.38 VICI::EBNF::EBNF Class Reference

A parser for [EBNF](#).

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

Inheritance diagram for VICI::EBNF::EBNF:



Public Member Functions

- virtual `~EBNF ()`
virtual destructor
- virtual bool `validate (csr s)=0`
confirm that an EBNF is valid
- virtual void `getError (int &line, int &column, std::string &text)=0`
get location and details of parsing error
- virtual `ParseTree * parse (csr s)=0`
parse the EBNF

8.38.1 Detailed Description

A parser for [EBNF](#).

The definition of the facade for libebnf.

The implementation will be responsible for parsing some [EBNF](#) text to create a parse tree for use by libsyntax and libcommand.

8.38.2 Member Function Documentation

8.38.2.1 virtual void VICI::EBNF::EBNF::getError (int & line, int & column, std::string & text) [pure virtual]

get location and details of parsing error

Parameters

<i>line</i>	the linenumber of the error returned in this
<i>column</i>	the position on the line of the error returned in this
<i>text</i>	the details of the error

Implemented in [VICI::stub::EBNF_Stub](#).

8.38.2.2 virtual ParseTree* VICI::EBNF::EBNF::parse(*csr s*) [pure virtual]

parse the [EBNF](#)

Parameters

<code>s</code>	the text to parse.
----------------	--------------------

Implemented in [VICI::stub::EBNF_Stub](#).

8.38.2.3 virtual bool VICI::EBNF::EBNF::validate (`csr s`) [pure virtual]

confirm that an [EBNF](#) is valid

Parameters

<code>s</code>	the text to validate
----------------	----------------------

Implemented in [VICI::stub::EBNF_Stub](#).

The documentation for this class was generated from the following file:

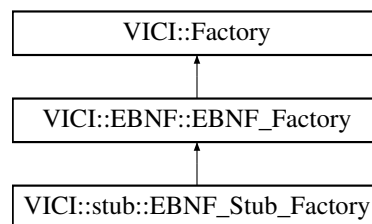
- [vici.h](#)

8.39 VICI::EBNF::EBNF_Factory Class Reference

An abstract factory for making an instance of [EBNF](#).

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

Inheritance diagram for VICI::EBNF::EBNF_Factory:



Public Member Functions

- virtual `~EBNF_Factory` ()
virtual destructor
- virtual `EBNF * makeEBNF` ()=0
the factory for EBNF objects

8.39.1 Detailed Description

An abstract factory for making an instance of [EBNF](#).

An abstract factory for making an instance of [EBNF](#). The implementation will create either the production version, or a stub, or a test version of the [EBNF](#) class.

8.39.2 Member Function Documentation

8.39.2.1 virtual EBNF* VICI::EBNF::EBNF_Factory::makeEBNF () [pure virtual]

the factory for [EBNF](#) objects

Returns

an instance of [EBNF](#) according to the type of the factory.

Implemented in [VICI::stub::EBNF_Stub_Factory](#).

The documentation for this class was generated from the following file:

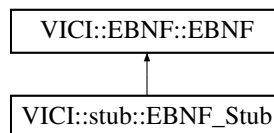
- [vici.h](#)

8.40 VICI::stub::EBNF_Stub Class Reference

A stub version of the [EBNF](#) module interface.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::EBNF_Stub:

**Public Member Functions**

- virtual bool [validate](#) (*csr s*)
confirm that an EBNF is valid
- virtual void [getError](#) (int &, int &, std::string &)
get location and details of parsing error
- virtual [VICI::EBNF::ParseTree](#) * [parse](#) (*csr s*)
parse the EBNF

8.40.1 Detailed Description

A stub version of the [EBNF](#) module interface.

8.40.2 Member Function Documentation

8.40.2.1 void [EBNF_Stub::getError](#) (int & *line*, int & *column*, std::string & *text*) [virtual]

get location and details of parsing error

Parameters

<i>line</i>	the linenumber of the error returned in this
<i>column</i>	the position on the line of the error returned in this
<i>text</i>	the details of the error

Implements [VICI::EBNF::EBNF](#).

8.40.2.2 [EBNF::ParseTree](#) * [EBNF_Stub::parse](#) (*csr s*) [virtual]

parse the [EBNF](#)

Parameters

<code>s</code>	the text to parse.
----------------	--------------------

Implements [VICI::EBNF::EBNF](#).

8.40.2.3 `bool EBNF_Stub::validate (const s) [virtual]`

confirm that an [EBNF](#) is valid

Parameters

<code>s</code>	the text to validate
----------------	----------------------

Implements [VICI::EBNF::EBNF](#).

The documentation for this class was generated from the following files:

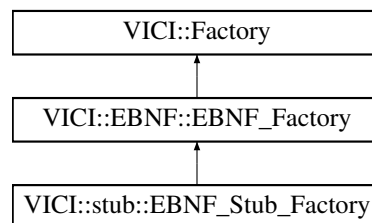
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.41 VICI::stub::EBNF_Stub_Factory Class Reference

A factory for creating instances of the [EBNF_Stub](#).

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::EBNF_Stub_Factory:



Public Member Functions

- virtual [EBNF::EBNF](#) * `makeEBNF` ()
the factory for [EBNF](#) objects

8.41.1 Detailed Description

A factory for creating instances of the [EBNF_Stub](#).

8.41.2 Member Function Documentation

8.41.2.1 `EBNF::EBNF` * `EBNF_Stub_Factory::makeEBNF` () [virtual]

the factory for [EBNF](#) objects

Returns

an instance of [EBNF](#) according to the type of the factory.

Implements [VICI::EBNF::EBNF_Factory](#).

The documentation for this class was generated from the following files:

- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.42 VICI::EbnfNode Class Reference

A node of the [EBNF](#) parse tree.

```
#include <vici/parseTree.h>
```

Public Types

- enum [NodeType](#) {
[Undefined](#), [Grammar](#), [Production](#), [Terminal](#),
[Quotation](#), [Name](#), [Repetition](#), [Option](#),
[Sequence](#), [Choice](#) }

the type of node, which corresponds to the non-terminals of the [EBNF](#) for [EBNF](#).

Public Member Functions

- [EbnfNode](#) ()
Constructor.
- [~EbnfNode](#) ()
Destructor.
- `std::string` [typeOfNode](#) () const
get the type of the node
- `void` [typeOfNode](#) (const `std::string` &t)
set the type of the node

Public Attributes

- [EbnfNode](#) * [parent](#)
pointer to parent node
- [EbnfNode](#) * [prev](#)
pointer to previous sibling node
- [EbnfNode](#) * [next](#)
pointer to next sibling node
- [EbnfNode](#) * [firstChild](#)
pointer to first child node
- [EbnfNode](#) * [lastChild](#)
pointer to last child node
- [NodeType](#) [nodeType](#)
defines what [EBNF](#) non-terminal the node represents
- `std::string` [text](#)
the terminal leaves of the [EBNF](#)

8.42.1 Detailed Description

A node of the [EBNF](#) parse tree.

A node of the [EbnfTree](#) used to pass the parsed form of an [EBNF](#) definition between modules.

8.42.2 Member Enumeration Documentation

8.42.2.1 enum VICI::EbnfNode::NodeType

the type of node, which corresponds to the non-terminals of the [EBNF](#) for [EBNF](#).

Enumerator

Undefined just until we load something sensible.

Grammar the root of the tree - only one of these.

Production one for each production. Can have any number of child nodes. firstChild is the Name of the Production.

Terminal one or two children of type Quotation. firstChild == lastChild if only one child.

Quotation has no children

Name has no children

Repetition can have any number of children

Option can have any number of children

Sequence can have any number of children

Choice has two children

8.42.3 Member Function Documentation

8.42.3.1 string EbnfNode::typeOfNode () const

get the type of the node

Returns

the type of the node as a string

8.42.3.2 void EbnfNode::typeOfNode (const std::string & t)

set the type of the node

Parameters

<i>t</i>	the type of the node as a string
----------	----------------------------------

The documentation for this class was generated from the following files:

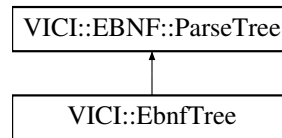
- [parseTree.h](#)
- [parseTree.cpp](#)

8.43 VICI::EbnfTree Class Reference

An implementation of the ParseTree type.

```
#include <vici/parseTree.h>
```

Inheritance diagram for VICI::EbnfTree:



Public Member Functions

- [EbnfTree](#) ()
constructor
- virtual [~EbnfTree](#) ()
destructor
- void [exportXml](#) (const [Path](#) &p)
Save the tree as an XML file.
- void [importXml](#) (const [Path](#) &p)
Load the tree from an XML file.

Public Attributes

- [EbnfNode](#) * [root](#)
root node of the tree - type is Grammar

8.43.1 Detailed Description

An implementation of the ParseTree type.

An implementation of the ParseTree type used to pass the parsed [EBNF](#) between modules.

8.43.2 Member Function Documentation

8.43.2.1 void EbnfTree::exportXml (const Path & p)

Save the tree as an XML file.

Parameters

<i>p</i>	The path to save the file.
----------	----------------------------

8.43.2.2 void EbnfTree::importXml (const Path & p)

Load the tree from an XML file.

Parameters

<i>p</i>	The path to read the file from.
----------	---------------------------------

The documentation for this class was generated from the following files:

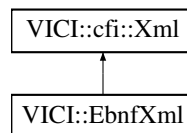
- [parseTree.h](#)
- [parseTree.cpp](#)

8.44 VICI::EbnfXml Class Reference

A specialization of the Xml class for the parse tree.

```
#include <vici/parseTree.h>
```

Inheritance diagram for VICI::EbnfXml:



Public Member Functions

- [EbnfXml](#) ()
constructor
- void [exportTree](#) ([EbnfTree](#) *tree, const [Path](#) &path)
export the tree to parse-tree.xml
- void [importTree](#) ([EbnfTree](#) *tree, const [Path](#) &path)
import a tree from parse-tree.xml

Additional Inherited Members

8.44.1 Detailed Description

A specialization of the Xml class for the parse tree.

A specialization of the Xml class used to save and restore a parse tree. This is mostly used during testing of the syntax library so that a variety of test cases can be created without needing the [EBNF](#) library.

8.44.2 Member Function Documentation

8.44.2.1 void EbnfXml::exportTree (EbnfTree * tree, const Path & path)

export the tree to parse-tree.xml

Parameters

<i>tree</i>	The tree to export
<i>path</i>	The path to write file to.

8.44.2.2 void EbnfXml::importTree (EbnfTree * *tree*, const Path & *path*)

import a tree from parse-tree.xml

Parameters

<i>tree</i>	The tree to import into
<i>path</i>	The path to load the file from.

The documentation for this class was generated from the following files:

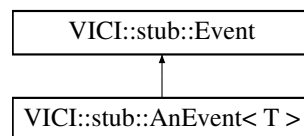
- [parseTree.h](#)
- [parseTree.cpp](#)

8.45 VICI::stub::Event Class Reference

Define an Abstract base type for [Dispatcher](#) events.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::Event:



Public Member Functions

- virtual void [execute](#) ()=0

Execute the associated function pointer.

8.45.1 Detailed Description

Define an Abstract base type for [Dispatcher](#) events.

The documentation for this class was generated from the following file:

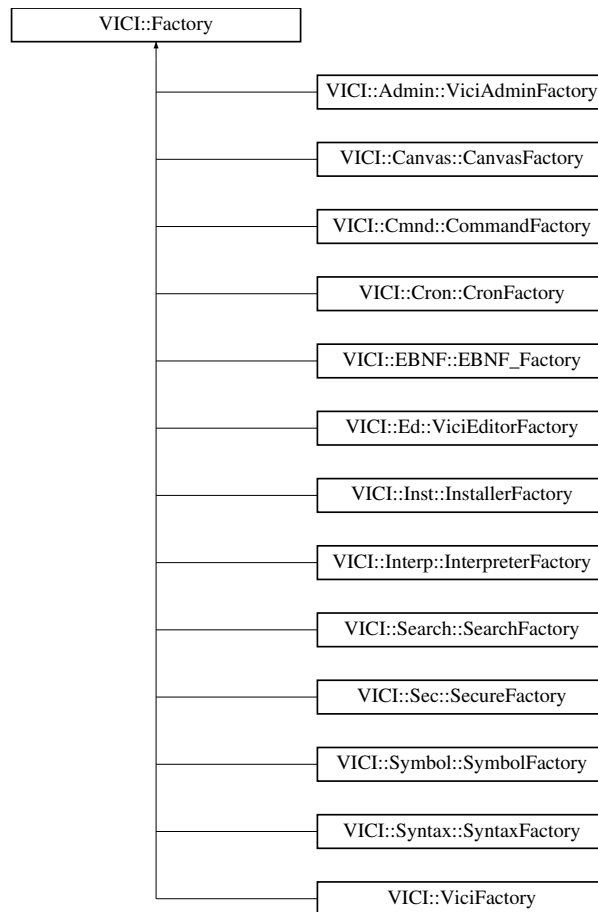
- [libifstubs.h](#)

8.46 VICI::Factory Class Reference

An abstract type for factories.

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

Inheritance diagram for VICI::Factory:



Public Member Functions

- virtual `~Factory ()`
Destructor.

8.46.1 Detailed Description

An abstract type for factories.

The documentation for this class was generated from the following file:

- [vici.h](#)

8.47 VICI::FactoryFactory Class Reference

Responsible for creating and supplying factories for the main modules.

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

Public Member Functions

- `template<class T >`
void `getFactory (Module nm, std::shared_ptr< T > &f)`
Get a factory of the specified type.

- void [registerFactory](#) (Module *m*, FactoryPtr *f*)
Register a factory.

Static Public Member Functions

- static [FactoryFactory](#) & [instance](#) ()
Get a reference to the singleton object.

8.47.1 Detailed Description

Responsible for creating and supplying factories for the main modules.

8.47.2 Member Function Documentation

8.47.2.1 `template<class T > void VICI::FactoryFactory::getFactory (Module nm, std::shared_ptr<T> & f)` [inline]

Get a factory of the specified type.

Parameters

<i>nm</i>	The type of the required factory
<i>f</i>	A shared pointer for the factory.

8.47.2.2 `void FactoryFactory::registerFactory (Module m, FactoryPtr f)`

Register a factory.

This is used by test harnesses to supply alternate test version of modules.

Parameters

<i>m</i>	Specify the type of the factory.
<i>f</i>	The factory to return for the type.

The documentation for this class was generated from the following files:

- [vici.h](#)
- [factory.cpp](#)

8.48 VICI::cfi::FD Class Reference

Wrap file descriptors in a class to ensure closed.

```
#include <sos/fdstream.h>
```

Public Member Functions

- [FD](#) (const std::string &name, int mode, bool lock)
constructor which opens the file
- [FD](#) (int x)
constructor for existing file descriptor
- [~FD](#) ()
destructor

- [operator int \(\)](#)
type conversion

Static Public Member Functions

- static void [report](#) (std::ostream &s, int fd)
Report the details or status of a file descriptor.

8.48.1 Detailed Description

Wrap file descriptors in a class to ensure closed.

The [FD](#) class is used to wrap a file descriptor so that it will be automatically closed when it goes out of scope. This ensures that file descriptors are closed when an exception is thrown.

8.48.2 Constructor & Destructor Documentation

8.48.2.1 FD::FD (const std::string & name, int mode, bool lock)

constructor which opens the file

Parameters

<i>name</i>	the name of file to open
<i>mode</i>	the open mode
<i>lock</i>	if true the file is also locked

8.48.2.2 VICI::cfi::FD::FD (int x) [inline]

constructor for existing file descriptor

Parameters

<i>x</i>	the existing file descriptor
----------	------------------------------

8.48.3 Member Function Documentation

8.48.3.1 void FD::report (std::ostream & s, int fd) [static]

Report the details or status of a file descriptor.

Parameters

<i>s</i>	the stream to write to
<i>fd</i>	the file descriptor to examine.

The documentation for this class was generated from the following files:

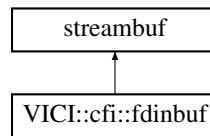
- [fdstream.h](#)
- [fd.cpp](#)

8.49 VICI::cfi::fdinbuf Class Reference

An input stream buffer.

```
#include <sos/fdstream.h>
```

Inheritance diagram for VICI::cfi::fdinbuf:



Public Member Functions

- `fdinbuf` (int fd, bool isCloseNeeded, `fdistream` &owner)
constructor
- void `close` ()
Close the input file descriptor.
- virtual `~fdinbuf` ()
destructor.

Protected Member Functions

- virtual int `underflow` ()
read characters from the buffer.

Protected Attributes

- `fdistream` & `mOwner`
Our owning input stream.
- int `mFd`
The input file descriptor.
- bool `mIsCloseNeeded`
Whether the file descriptor needs to be closed on destruction.
- char `mBuffer` [BUFFER_SIZE+PUSHBACK_SIZE]

Static Protected Attributes

- static const int `BUFFER_SIZE` = 8192
Size of buffer.
- static const int `PUSHBACK_SIZE` = 4
maximum number of push back characters supported

8.49.1 Detailed Description

An input stream buffer.

An input stream buffer which reads from a file descriptor. This provides input buffering.

8.49.2 Constructor & Destructor Documentation

8.49.2.1 `fdinbuf::fdinbuf (int fd, bool isCloseNeeded, fdistream & owner)`

constructor

Constructor which accepts the file descriptor to read from.

Parameters

<i>fd</i>	the file descriptor to read from
<i>isCloseNeeded</i>	a flag indicating whether the file descriptor is closed on destruction
<i>owner</i>	our owning input stream

8.49.3 Member Data Documentation

8.49.3.1 char VICI::cfi::fdinbuf::mBuffer[BUFFER_SIZE+PUSHBACK_SIZE] [protected]

Input buffer. This holds the data read and a bit more for a push-back buffer.

The documentation for this class was generated from the following files:

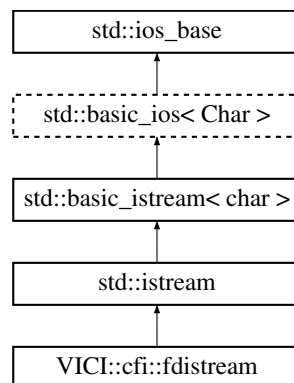
- [fdstream.h](#)
- [fdistream.cpp](#)

8.50 VICI::cfi::fdistream Class Reference

A file descriptor input stream.

```
#include <sos/fdstream.h>
```

Inheritance diagram for VICI::cfi::fdistream:



Public Member Functions

- [fdistream](#) (int fd, bool isCloseNeeded=false)
constructor
- void [close](#) ()
close the file descriptor

Static Public Attributes

- static bool [CLOSE_NEEDED](#) = true
Flag to indicate file descriptor should be closed.
- static bool [CLOSE_NOT_NEEDED](#) = false
Flag to indicate file descriptor should not be closed.

Protected Attributes

- [fdinbuf mBuf](#)
Our input file descriptor stream buffer.

Friends

- class **fdinbuf**

8.50.1 Detailed Description

A file descriptor input stream.

8.50.2 Constructor & Destructor Documentation

8.50.2.1 `fdistream::fdistream (int fd, bool isCloseNeeded = false)`

constructor

Constructor which accepts the file descriptor to read from

Parameters

<i>fd</i>	the file descriptor to read from
<i>isCloseNeeded</i>	a flag indicating whether the file descriptor is closed on destruction.

The documentation for this class was generated from the following files:

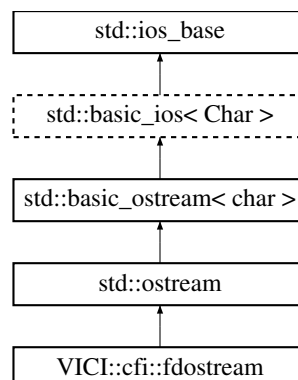
- [fdstream.h](#)
- [fdistream.cpp](#)

8.51 VICI::cfi::fdostream Class Reference

A file descriptor output stream.

```
#include <sos/fdostream.h>
```

Inheritance diagram for VICI::cfi::fdostream:



Public Member Functions

- [fdostream](#) (int fd, bool isCloseNeeded=false)

constructor

- void [close](#) ()

close the file descriptor

Static Public Attributes

- static bool [CLOSE_NEEDED](#) = true
Flag to indicate file descriptor should be closed.
- static bool [CLOSE_NOT_NEEDED](#) = false
Flag to indicate file descriptor should not be closed.

Protected Attributes

- [fdoutbuf mBuf](#)
Our output file descriptor stream buffer.

Friends

- class [fdoutbuf](#)

8.51.1 Detailed Description

A file descriptor output stream.

8.51.2 Constructor & Destructor Documentation

8.51.2.1 fdostream::fdostream (int *fd*, bool *isCloseNeeded* = false)

constructor

Constructor which accepts the file descriptor to write to and a flag indicating whether the file descriptor is closed on destruction.

Parameters

<i>fd</i>	the file descriptor to write to
<i>isCloseNeeded</i>	if true close on destruction

The documentation for this class was generated from the following files:

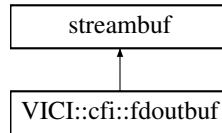
- [fdstream.h](#)
- [fdostream.cpp](#)

8.52 VICI::cfi::fdoutbuf Class Reference

An output stream buffer.

```
#include <sos/fdstream.h>
```

Inheritance diagram for VICI::cfi::fdoutbuf:



Public Member Functions

- `fdoutbuf` (int fd, bool isCloseNeeded, `fdostream` &owner)
constructor
- void `close` ()
Close the output file descriptor.
- virtual `~fdoutbuf` ()
destructor.

Protected Member Functions

- int `flushBuffer` ()
Flush the characters in the buffer.
- virtual int `overflow` (int c)
Write the indicated character and all previous characters.
- virtual int `sync` ()
Flush the data in the buffer.

Protected Attributes

- `fdostream` & `mOwner`
Our owning output stream.
- int `mFd`
The output file descriptor.
- bool `mIsCloseNeeded`
Whether the file descriptor needs to be closed on destruction.
- char `mBuffer` [BUFFER_SIZE]
Output buffer.

Static Protected Attributes

- static const int `BUFFER_SIZE` = 8192
Output buffer size.

8.52.1 Detailed Description

An output stream buffer.

An output stream buffer which writes to a file descriptor. This provides output buffering.

8.52.2 Constructor & Destructor Documentation

8.52.2.1 `fdoutbuf::fdoutbuf (int fd, bool isCloseNeeded, fdostream & owner)`

constructor

Parameters

<i>fd</i>	the file descriptor to write to
<i>isCloseNeeded</i>	a flag indicating whether the file descriptor is closed on destruction
<i>owner</i>	our owning output stream

8.52.3 Member Function Documentation

8.52.3.1 void fdobuf::close ()

Close the output file descriptor.

This will be done whether or not the close needed flag is set.

8.52.3.2 int fdobuf::overflow (int c) [protected],[virtual]

Write the indicated character and all previous characters.

Parameters

<i>c</i>	the current character.
----------	------------------------

The documentation for this class was generated from the following files:

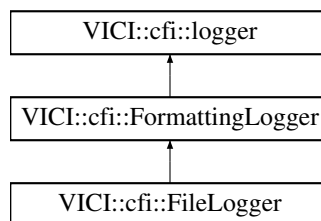
- [fdstream.h](#)
- [fdostream.cpp](#)

8.53 VICI::cfi::FileLogger Class Reference

Class for logging to a file.

```
#include <vici/log.h>
```

Inheritance diagram for VICI::cfi::FileLogger:



Public Member Functions

- [FileLogger](#) (const [Path](#) &fname)
Constructor.
- int [log](#) ([Severity](#), *csr* line)
log to the unnamed log
- int [log](#) (*csr* logName, [Severity](#), *csr* line)
log to the named log

Protected Attributes

- `std::ofstream f`
The logging stream.

Additional Inherited Members

8.53.1 Detailed Description

Class for logging to a file.

Writes log messages to a file.

The log messages include the output from the [FormattingLogger](#).

8.53.2 Constructor & Destructor Documentation

8.53.2.1 FileLogger::FileLogger (const Path & *fname*)

Constructor.

Parameters

<i>fname</i>	the name of the log file.
--------------	---------------------------

8.53.3 Member Function Documentation

8.53.3.1 int FileLogger::log (Severity *sev*, *csr line*) [virtual]

log to the unnamed log

Parameters

<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

8.53.3.2 int FileLogger::log (*csr logName*, Severity *sev*, *csr line*) [virtual]

log to the named log

Parameters

<i>logName</i>	the name of the log stream to write to
<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

The documentation for this class was generated from the following files:

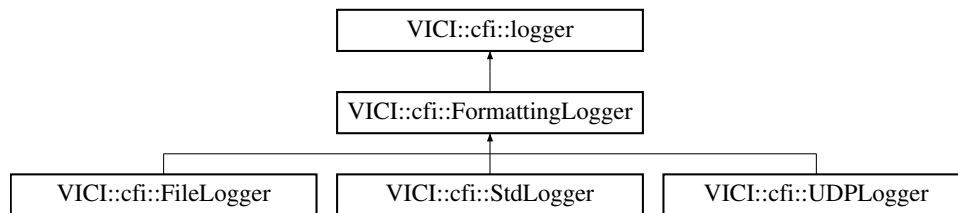
- [log.h](#)
- [log.cpp](#)

8.54 VICI::cfi::FormattingLogger Class Reference

Class for producing a formatted log message.

```
#include <vici/log.h>
```

Inheritance diagram for VICI::cfi::FormattingLogger:



Protected Member Functions

- `std::string timeStamp ()`
generates a string containing the current time
- `csr hostname ()`
generates a string containing the hostname.
- `csr process ()`
generates a string containing the process id
- `csr severity (Severity)`
generates a string containing the severity level
- `int format (std::ostream &s, Severity sev, csr msg)`
output the message to the stream

Additional Inherited Members

8.54.1 Detailed Description

Class for producing a formatted log message.

Provides a common formatting function for the different loggers. Writes log messages with a time stamp, host name, and process id in addition to the severity and the message.

8.54.2 Member Function Documentation

8.54.2.1 `int FormattingLogger::format (std::ostream & s, Severity sev, csr msg)` [protected]

output the message to the stream

Parameters

<code>s</code>	the log stream to write to
<code>sev</code>	the severity level to use
<code>msg</code>	the message to write to the log

Returns

the number of characters written to the stream

8.54.2.2 string FormattingLogger::timeStamp () [protected]

generates a string containing the current time

Returns

current time as a string with format of H:M:S Y-m-d

The documentation for this class was generated from the following files:

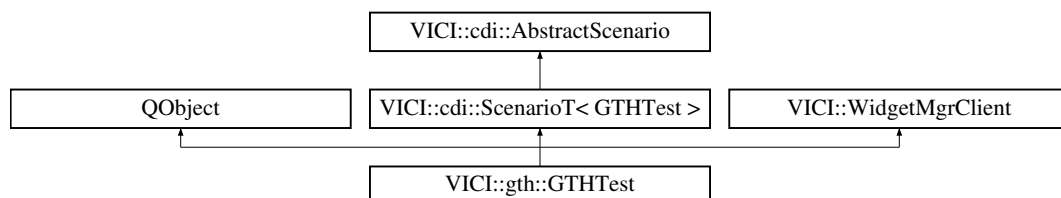
- [log.h](#)
- [log.cpp](#)

8.55 VICI::gth::GHTTest Class Reference

An AbstractTest derived class for testing GUI programs.

```
#include <vici/libgth.h>
```

Inheritance diagram for VICI::gth::GHTTest:

**Signals**

- void [testingCompleted](#) ()
This signal is emitted when the main window tests have completed.

Public Member Functions

- void [jumpThread](#) (csr tname, csr widgetName, csr command, const std::vector< std::string > ¶ms)
Method for thread jumping.
- [GHTTest](#) (csr name)
Costructor.
- [~GHTTest](#) ()
Destructor.
- csr [getName](#) ()
Get the name of the scenario.

- virtual bool [willRunTests](#) ()
Indicate that the scenario will take over from the Tester.
- virtual void [runTests](#) (csr scenarioName, cdi::ScenarioResults *)
Run an event loop and wait until the testing completes.
- void [endMainWindow](#) ()
Called when testing for the main window completes.
- void [runActions](#) (csr testCase, ConDes)
Called by the constructor and destructor of a test case.
- void [assignVariables](#) (const TestAction &, csr result)
Applies a regular expression to the result to get values for variables.
- std::string [getJump](#) (const TestAction &, csr result)
Applies regular expressions to the result to find a jump label.
- csr [getVar](#) (csr varName) const
Get the value of a variable.
- void [substVars](#) (const ParamList &, ParamList &)
Substitute variables for their values in the parameters.
- virtual void [windowHasRegistered](#) (GTHWindowWidget *)
Called to notify of a new window registering itself.
- virtual void [windowHasDeregistered](#) (GTHWindowWidget *)
Called to notify that a window has deregistered itself.
- void [regWidget](#) (void *w, VICI::gth::WidgetType t, csr n)
Register a widget so that it can be found by name.
- std::string [prefix](#) ()
get thread and time

Public Attributes

- cdi::ScenarioResults * [results](#)
Reference to the results.

Additional Inherited Members

8.55.1 Detailed Description

An AbstractTest derived class for testing GUI programs.

This object is constructed at the start of a test run and destroyed when the test is finished.

The documentation for this class was generated from the following files:

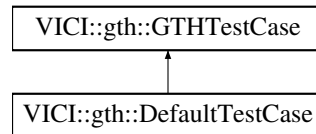
- [libgth.h](#)
- [libgth.cpp](#)

8.56 VICI::gth::GHTTestCase Class Reference

Interface for test cases that do gui testing.

```
#include <vici/libgth.h>
```

Inheritance diagram for VICI::gth::GHTTestCase:



Protected Member Functions

- `csr getVar (csr)`
Get the value of a variable from `GTHTest`.
- `GTHTestCase (csr name)`
Constructor.
- `~GTHTestCase ()`
Destructor.

Protected Attributes

- `GTHTest * myScenario`
The owning scenario.
- `const std::string & name`
The name of the test case, as installed.

8.56.1 Detailed Description

Interface for test cases that do gui testing.

The documentation for this class was generated from the following files:

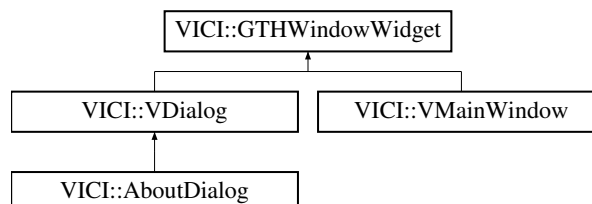
- `libgth.h`
- `libgth.cpp`

8.57 VICI::GTHWindowWidget Class Reference

A mix-in class with the functions required for the gui test harness.

```
#include <vici/libgui.h>
```

Inheritance diagram for `VICI::GTHWindowWidget`:



Public Member Functions

- virtual void `RegnFn (std::function< void(void *, gth::WidgetType, csr)> reg)=0`
This may be called by the test harness to get pointers to the individual widgets.

Protected Member Functions

- void [registerWindow](#) ()
This is called by showEvent on first show.

8.57.1 Detailed Description

A mix-in class with the functions required for the gui test harness.

The documentation for this class was generated from the following files:

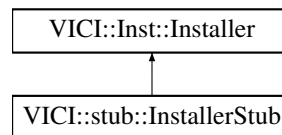
- [libgui.h](#)
- [libgui.cpp](#)

8.58 VICI::Inst::Installer Class Reference

Install a script into the user's desktop.

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

Inheritance diagram for VICI::Inst::Installer:



Public Member Functions

- virtual [~Installer](#) ()
virtual destructor
- virtual void [setCurrentFile](#) (csr filename)=0
Set the script to install.
- virtual void [show](#) ()=0
Display the window.

8.58.1 Detailed Description

Install a script into the user's desktop.

The facade for the installer library.

The implementation will allow the user to install the script into their desktop menu system.

8.58.2 Member Function Documentation

8.58.2.1 virtual void VICI::Inst::Installer::setCurrentFile (csr filename) [pure virtual]

Set the script to install.

Parameters

<i>filename</i>	the script to install.
-----------------	------------------------

Implemented in [VICI::stub::InstallerStub](#).

The documentation for this class was generated from the following file:

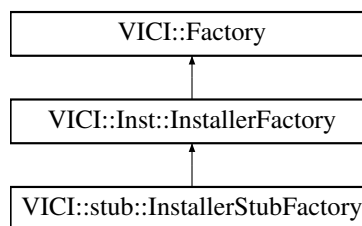
- [vici.h](#)

8.59 VICI::Inst::InstallerFactory Class Reference

An abstract factory for making an instance of [Installer](#).

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

Inheritance diagram for VICI::Inst::InstallerFactory:



Public Member Functions

- virtual [~InstallerFactory](#) ()
virtual destructor
- virtual [Installer](#) * [makeInstaller](#) ([Window](#) *w)=0
create an instance of the [Installer](#) class

8.59.1 Detailed Description

An abstract factory for making an instance of [Installer](#).

The implementation will create either the production version, or a stub, or a test version of the [Installer](#) class.

8.59.2 Member Function Documentation

8.59.2.1 virtual [Installer](#)* [VICI::Inst::InstallerFactory::makeInstaller](#) ([Window](#) * w) [pure virtual]

create an instance of the [Installer](#) class

Parameters

<i>w</i>	the window to display into
----------	----------------------------

Returns

an instance of the [Installer](#).

Implemented in [VICI::stub::InstallerStubFactory](#).

The documentation for this class was generated from the following file:

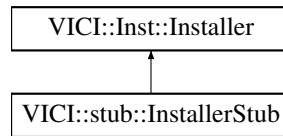
- [vici.h](#)

8.60 VICI::stub::InstallerStub Class Reference

A stub version of the Installer module.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::InstallerStub:



Public Member Functions

- [InstallerStub \(Window *w\)](#)
Constructor.
- virtual void [setCurrentFile \(csr filename\)](#)
Set the script to install.
- virtual void [show \(\)](#)
Display the window.

8.60.1 Detailed Description

A stub version of the Installer module.

8.60.2 Constructor & Destructor Documentation

8.60.2.1 InstallerStub::InstallerStub (Window * w)

Constructor.

Parameters

<i>w</i>	The window to display into.
----------	-----------------------------

8.60.3 Member Function Documentation

8.60.3.1 void InstallerStub::setCurrentFile (csr filename) [virtual]

Set the script to install.

Parameters

<i>filename</i>	the script to install.
-----------------	------------------------

Implements [VICI::Inst::Installer](#).

The documentation for this class was generated from the following files:

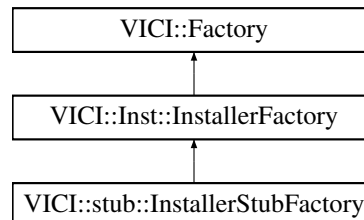
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.61 VICI::stub::InstallerStubFactory Class Reference

A factory for creating stub instances of Installer objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::InstallerStubFactory:



Public Member Functions

- virtual [Inst::Installer](#) * [makeInstaller](#) ([Window](#) *)
create an instance of the Installer class

8.61.1 Detailed Description

A factory for creating stub instances of Installer objects.

8.61.2 Member Function Documentation

8.61.2.1 [Inst::Installer](#) * [InstallerStubFactory::makeInstaller](#) ([Window](#) * *w*) [virtual]

create an instance of the Installer class

Parameters

<i>w</i>	the window to display into
----------	----------------------------

Returns

an instance of the Installer.

Implements [VICI::Inst::InstallerFactory](#).

The documentation for this class was generated from the following files:

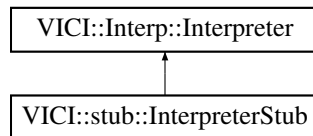
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.62 VICI::Interp::Interpreter Class Reference

The API for interpreter library.

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

Inheritance diagram for VICI::Interp::Interpreter:



Public Member Functions

- virtual `~Interpreter ()`
virtual destructor
- virtual void `addClient (InterpreterClient *)=0`
add a client for notifications
- virtual void `setScript (csr filename)=0`
the XML VICI script to execute
- virtual void `setArgs (VICI::ArgList args)=0`
set the program options
- virtual void `debugMode (bool mode)=0`
enable debug mode
- virtual int `openDisplay (Nodeld node)=0`
open a Display object for reading
- virtual void `dataAck (Nodeld node)=0`
call this after responding to InterpreterClient::dataReady() to wait for the next set of data.
- virtual void `setValue (csr varName, csr value)=0`
this is called when a variable has its value changed by the user
- virtual void `setPosn (ThreadId tid, Nodeld node)=0`
call this to set the point where execution resumes
- virtual void `setBreak (Nodeld node, bool set)=0`
call this to mark a node as a break point
- virtual void `setInterval (double secs)=0`
set the time interval between executing commands
- virtual void `saveSnapshot (csr filename)=0`
save a snapshot file
- virtual void `loadSnapshot (csr filename)=0`
restore a snapshot file
- virtual void `run ()=0`
start running the script
- virtual void `run (csr functionName)=0`
start running the script at a specified function
- virtual void `pause ()=0`
pause the script
- virtual void `step (ThreadId tid)=0`
- virtual void `resume ()=0`
resume a paused script
- virtual void `kill ()=0`
stop the script
- virtual int `result ()=0`
get the exit code

Static Public Attributes

- static const [NodeId OutDisplay](#) = -1
the node id of the default output display
- static const [NodeId ErrDisplay](#) = -2
the node id of the default error display

8.62.1 Detailed Description

The API for interpreter library.

This is the facade for the interpreter library, libvici.

It provides an interface that lets the user control the operation of the interpreter.

8.62.2 Member Function Documentation

8.62.2.1 virtual void VICI::Interp::Interpreter::dataAck ([NodeId node](#)) [pure virtual]

call this after responding to [InterpreterClient::dataReady\(\)](#) to wait for the next set of data.

Parameters

<i>node</i>	the node id of the display that is being acknowledged.
-------------	--

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.2 virtual void VICI::Interp::Interpreter::debugMode ([bool mode](#)) [pure virtual]

enable debug mode

turning this on causes events to be generated at each step of the processing.

Parameters

<i>mode</i>	set this true to enable debugging mode
-------------	--

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.3 virtual void VICI::Interp::Interpreter::loadSnapshot ([csr filename](#)) [pure virtual]

restore a snapshot file

Parameters

<i>filename</i>	the name of the file to load the state from
-----------------	---

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.4 virtual int VICI::Interp::Interpreter::openDisplay ([NodeId node](#)) [pure virtual]

open a Display object for reading

the [InterpreterClient::dataReady\(\)](#) will be called when the file descriptor has data to be read.

Parameters

<i>node</i>	the display object to open
-------------	----------------------------

Returns

file descriptor of the display object

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.5 virtual void VICI::Interp::Interpreter::run (*csr functionName*) [pure virtual]

start running the script at a specified function

Parameters

<i>functionName</i>	the function to start
---------------------	-----------------------

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.6 virtual void VICI::Interp::Interpreter::saveSnapshot (*csr filename*) [pure virtual]

save a snapshot file

Parameters

<i>filename</i>	the name of the file to save the interpreter state to
-----------------	---

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.7 virtual void VICI::Interp::Interpreter::setBreak (*Nodeld node, bool set*) [pure virtual]

call this to mark a node as a break point

Parameters

<i>node</i>	The node to break execution on when reached
<i>set</i>	True to set the breakpoint and false to clear it.

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.8 virtual void VICI::Interp::Interpreter::setInterval (*double secs*) [pure virtual]

set the time interval between executing commands

Parameters

<i>secs</i>	the interval to wait between executing commands
-------------	---

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.9 virtual void VICI::Interp::Interpreter::setPosn (*ThreadId tid, Nodeld node*) [pure virtual]

call this to set the point where execution resumes

Parameters

<i>tid</i>	the execution thread to be repositioned
<i>node</i>	the place to resume execution

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.10 virtual void VICI::Interp::Interpreter::setValue (csr varName, csr value) [pure virtual]

this is called when a variable has its value changed by the user

Parameters

<i>varName</i>	the variable being changed
<i>value</i>	the new value to assign to the variable

Implemented in [VICI::stub::InterpreterStub](#).

8.62.2.11 virtual void VICI::Interp::Interpreter::step (ThreadId tid) [pure virtual]

Parameters

<i>tid</i>	the thread to execute one command from
------------	--

Implemented in [VICI::stub::InterpreterStub](#).

The documentation for this class was generated from the following file:

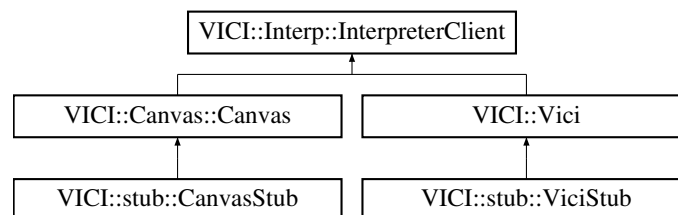
- [vici.h](#)

8.63 VICI::Interp::InterpreterClient Class Reference

The interface that must be implemented by clients of the interpreter.

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

Inheritance diagram for VICI::Interp::InterpreterClient:



Public Member Functions

- virtual [~InterpreterClient](#) ()
virtual destructor
- virtual void [setValue](#) (csr varName, csr value)=0
this gets called when a script variable has a change of value
- virtual void [setFile](#) (int state, csr filename)=0
this gets called when a monitored file changes state.
- virtual void [setCursor](#) (ThreadId tid, NodeId node)=0
this gets called as the shell steps through the script

- virtual void `breakReached (ThreadId tid, Nodeld node)=0`
this gets called when a break point is reached
- virtual void `dataReady (Nodeld node)=0`
this gets called when data is available on a display
- virtual void `reportError (Severity sev, csr msg)=0`
This gets called if an error is detected.
- virtual void `done ()=0`
this gets called when the script completes

8.63.1 Detailed Description

The interface that must be implemented by clients of the interpreter.

The clients of the interpreter are responsible for showing the activity during debugging and for displaying the output streams. By implementing this interface and registering with the interpreter they will notified accordingly.

8.63.2 Member Function Documentation

8.63.2.1 virtual void VICI::Interp::InterpreterClient::breakReached (ThreadId *tid*, Nodeld *node*) [pure virtual]

this gets called when a break point is reached

Parameters

<i>tid</i>	the thread which reached the breakpoint
<i>node</i>	the location of the breakpoint

Implemented in [VICI::stub::ViciStub](#), and [VICI::stub::CanvasStub](#).

8.63.2.2 virtual void VICI::Interp::InterpreterClient::dataReady (Nodeld *node*) [pure virtual]

this gets called when data is available on a display

Parameters

<i>node</i>	the node of the display object in the flowchart
-------------	---

Implemented in [VICI::stub::ViciStub](#), and [VICI::stub::CanvasStub](#).

8.63.2.3 virtual void VICI::Interp::InterpreterClient::reportError (Severity *sev*, csr *msg*) [pure virtual]

This gets called if an error is detected.

Parameters

<i>msg</i>	The text of the error message.
<i>sev</i>	The severity of the error

Implemented in [VICI::stub::ViciStub](#), and [VICI::stub::CanvasStub](#).

8.63.2.4 virtual void VICI::Interp::InterpreterClient::setCursor (ThreadId *tid*, Nodeld *node*) [pure virtual]

this gets called as the shell steps through the script

Parameters

<i>tid</i>	the thread which changed to executing this node
<i>node</i>	the node that is currently being executed

Implemented in [VICI::stub::ViciStub](#), and [VICI::stub::CanvasStub](#).

8.63.2.5 virtual void VICI::Interp::InterpreterClient::setFile (int *state*, *csr filename*) [pure virtual]

this gets called when a monitored file changes state.

Parameters

<i>state</i>	this needs to be defined
<i>filename</i>	the name of the file that changed

Implemented in [VICI::stub::ViciStub](#), and [VICI::stub::CanvasStub](#).

8.63.2.6 virtual void VICI::Interp::InterpreterClient::setValue (*csr varName*, *csr value*) [pure virtual]

this gets called when a script variable has a change of value

Parameters

<i>varName</i>	the name of the variable that changed value
<i>value</i>	the new value of the variable

Implemented in [VICI::stub::ViciStub](#), and [VICI::stub::CanvasStub](#).

The documentation for this class was generated from the following file:

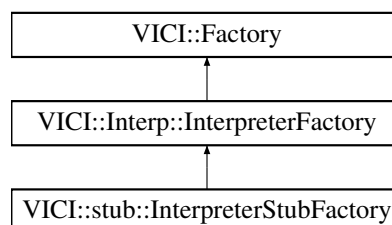
- [vici.h](#)

8.64 VICI::Interp::InterpreterFactory Class Reference

An abstract factory for making an instance of [Interpreter](#).

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

Inheritance diagram for VICI::Interp::InterpreterFactory:



Public Member Functions

- virtual [~InterpreterFactory](#) ()
virtual destructor
- virtual [Interpreter](#) * [makeInterpreter](#) ([Sec::Secure](#) *sec)=0
create an instance of the [Interpreter](#) class

8.64.1 Detailed Description

An abstract factory for making an instance of [Interpreter](#).

The implementation will create either the production version, or a stub, or a test version of the [Interpreter](#) class.

8.64.2 Member Function Documentation

8.64.2.1 `virtual Interpreter* VICI::Interp::InterpreterFactory::makeInterpreter (Sec::Secure * sec)` [pure virtual]

create an instance of the [Interpreter](#) class

Parameters

<code>sec</code>	pointer to the security class.
------------------	--------------------------------

Implemented in [VICI::stub::InterpreterStubFactory](#).

The documentation for this class was generated from the following file:

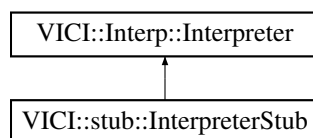
- [vici.h](#)

8.65 VICI::stub::InterpreterStub Class Reference

A stub version of the Interpreter module.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::InterpreterStub:



Public Member Functions

- [InterpreterStub](#) ([Sec::Secure](#) *s)
Constructor.
- virtual void [addClient](#) ([Interp::InterpreterClient](#) *sc)
add a client for notifications
- virtual void [debugMode](#) (bool)
enable debug mode
- virtual void [setScript](#) (csr filename)
the XML VICI script to execute
- virtual void [setArgs](#) ([ArgList](#) args)
set the program options
- virtual void [setValue](#) (csr varName, csr value)
this is called when a variable has its value changed by the user
- virtual void [setPosn](#) ([ThreadId](#), [Nodeld](#))
call this to set the point where execution resumes
- virtual void [setInterval](#) (double secs)
set the time interval between executing commands

- virtual void [setBreak](#) ([NodeId](#), bool)
 - call this to mark a node as a break point*
- virtual int [openDisplay](#) ([NodeId](#))
 - open a Display object for reading*
- virtual void [dataAck](#) ([NodeId](#))
 - call this after responding to InterpreterClient::dataReady() to wait for the next set of data.*
- virtual void [step](#) ([ThreadId](#))
- virtual void [resume](#) ()
 - resume a paused script*
- virtual void [run](#) ()
 - start running the script*
- virtual void [run](#) ([csr](#) funcName)
 - start running the script at a specified function*
- virtual void [pause](#) ()
 - pause the script*
- virtual void [kill](#) ()
 - stop the script*
- virtual int [result](#) ()
 - get the exit code*
- virtual void [saveSnapshot](#) ([csr](#) filename)
 - save a snapshot file*
- virtual void [loadSnapshot](#) ([csr](#) filename)
 - restore a snapshot file*

Additional Inherited Members

8.65.1 Detailed Description

A stub version of the Interpreter module.

8.65.2 Constructor & Destructor Documentation

8.65.2.1 InterpreterStub::InterpreterStub ([Sec::Secure](#) * [s](#))

Constructor.

Parameters

s	The security module.
-------------------	----------------------

8.65.3 Member Function Documentation

8.65.3.1 void InterpreterStub::dataAck ([NodeId](#) [node](#)) [virtual]

call this after responding to InterpreterClient::dataReady() to wait for the next set of data.

Parameters

node	the node id of the display that is being acknowledged.
----------------------	--

Implements [VICI::Interp::Interpreter](#).

8.65.3.2 void InterpreterStub::debugMode (bool *mode*) [virtual]

enable debug mode

turning this on causes events to be generated at each step of the processing.

Parameters

<i>mode</i>	set this true to enable debugging mode
-------------	--

Implements [VICI::Interp::Interpreter](#).

8.65.3.3 void InterpreterStub::loadSnapshot (*csr filename*) [virtual]

restore a snapshot file

Parameters

<i>filename</i>	the name of the file to load the state from
-----------------	---

Implements [VICI::Interp::Interpreter](#).

8.65.3.4 int InterpreterStub::openDisplay (*NodeId node*) [virtual]

open a Display object for reading

the InterpreterClient::dataReady() will be called when the file descriptor has data to be read.

Parameters

<i>node</i>	the display object to open
-------------	----------------------------

Returns

file descriptor of the display object

Implements [VICI::Interp::Interpreter](#).

8.65.3.5 void InterpreterStub::run (*csr functionName*) [virtual]

start running the script at a specified function

Parameters

<i>functionName</i>	the function to start
---------------------	-----------------------

Implements [VICI::Interp::Interpreter](#).

8.65.3.6 void InterpreterStub::saveSnapshot (*csr filename*) [virtual]

save a snapshot file

Parameters

<i>filename</i>	the name of the file to save the interpreter state to
-----------------	---

Implements [VICI::Interp::Interpreter](#).

8.65.3.7 void InterpreterStub::setBreak (*NodeId node, bool set*) [virtual]

call this to mark a node as a break point

Parameters

<i>node</i>	The node to break execution on when reached
<i>set</i>	True to set the breakpoint and false to clear it.

Implements [VICI::Interp::Interpreter](#).

8.65.3.8 void InterpreterStub::setInterval (double secs) [virtual]

set the time interval between executing commands

Parameters

<i>secs</i>	the interval to wait between executing commands
-------------	---

Implements [VICI::Interp::Interpreter](#).

8.65.3.9 void InterpreterStub::setPosn (ThreadId tid, NodeId node) [virtual]

call this to set the point where execution resumes

Parameters

<i>tid</i>	the execution thread to be repositioned
<i>node</i>	the place to resume execution

Implements [VICI::Interp::Interpreter](#).

8.65.3.10 void InterpreterStub::setValue (csr varName, csr value) [virtual]

this is called when a variable has its value changed by the user

Parameters

<i>varName</i>	the variable being changed
<i>value</i>	the new value to assign to the variable

Implements [VICI::Interp::Interpreter](#).

8.65.3.11 void InterpreterStub::step (ThreadId tid) [virtual]

Parameters

<i>tid</i>	the thread to execute one command from
------------	--

Implements [VICI::Interp::Interpreter](#).

The documentation for this class was generated from the following files:

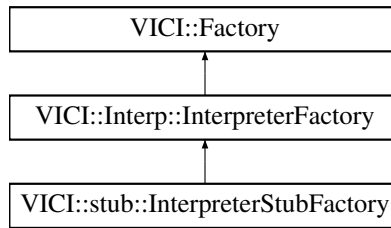
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.66 VICI::stub::InterpreterStubFactory Class Reference

A factory for creating stub instances of Interpreter objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::InterpreterStubFactory:



Public Member Functions

- virtual [Interp::Interpreter](#) * [makeInterpreter](#) ([Sec::Secure](#) *)
create an instance of the Interpreter class

8.66.1 Detailed Description

A factory for creating stub instances of Interpreter objects.

8.66.2 Member Function Documentation

8.66.2.1 [Interp::Interpreter](#) * [InterpreterStubFactory::makeInterpreter](#) ([Sec::Secure](#) * *sec*) [virtual]

create an instance of the Interpreter class

Parameters

<i>sec</i>	pointer to the security class.
------------	--------------------------------

Implements [VICI::Interp::InterpreterFactory](#).

The documentation for this class was generated from the following files:

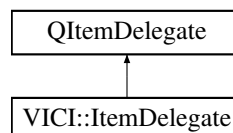
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.67 VICI::ItemDelegate Class Reference

An item delegate that uses QLineEdit for editing list and table entries.

```
#include <vici/libgui.h>
```

Inheritance diagram for VICI::ItemDelegate:



Signals

- void [validationError](#) (const QModelIndex &) const
This signal can be emitted if there is a validation error.

Public Member Functions

- [ItemDelegate](#) (QObject *parent=0)
Constructor.
- QWidget * [createEditor](#) (QWidget *parent, const QStyleOptionViewItem &option, const QModelIndex &index) const
Create a QLineEdit for editing the item.
- void [setEditorData](#) (QWidget *editor, const QModelIndex &index) const
Copy the data from the model to the editor.
- void [setModelData](#) (QWidget *editor, QAbstractItemModel *model, const QModelIndex &index) const
Copy the data from the editor to the model.
- void [updateEditorGeometry](#) (QWidget *editor, const QStyleOptionViewItem &option, const QModelIndex &index) const
Notification that the shape of the owner has changed.
- bool [hasEditor](#) () const
Check if delegate is in edit mode.

8.67.1 Detailed Description

An item delegate that uses QLineEdit for editing list and table entries.

The QListWidget and QTableWidget have faulty editing capability. This item can replace the default one to provide a much nicer user experience when editing the items in lists and tables.

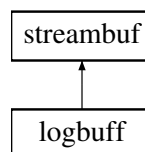
The documentation for this class was generated from the following files:

- [libgui.h](#)
- [libgui.cpp](#)

8.68 logbuff Class Reference

A streambuf class specialized for logging.

Inheritance diagram for logbuff:



Public Member Functions

- [logbuff](#) ()
default constructor
- [logbuff](#) (csr)
constructor for named log
- virtual [~logbuff](#) ()
destructor
- void [severity](#) (Severity s)
set severity

Protected Member Functions

- void `init ()`
initialize using details from the XINI config
- int `flushBuffer ()`
- virtual int `overflow (int c)`
- virtual int `sync ()`

Protected Attributes

- char `buffer [bufferSize]`
message collected into here
- `Severity sev`
the current severity level
- `logger * mLogger`
thing for writing the message
- string `name`
name to use in the XINI configuration

Static Protected Attributes

- static const int `bufferSize = 128`
limit line size in log files.

8.68.1 Detailed Description

A streambuf class specialized for logging.

This is a private implementation class.

8.68.2 Member Function Documentation

8.68.2.1 `int logbuff::flushBuffer ()` [protected]

we have a line for the log - we now delegate this to the log function that has been selected by the user.

8.68.2.2 `int logbuff::overflow (int c)` [protected],[virtual]

this gets called when the buffer fills. Since we are writing to a log file, this happens whenever the line length exceeds the buffer size.

8.68.2.3 `int logbuff::sync ()` [protected],[virtual]

this gets called when 'endl' or 'flush' is called on the stream.

The documentation for this class was generated from the following file:

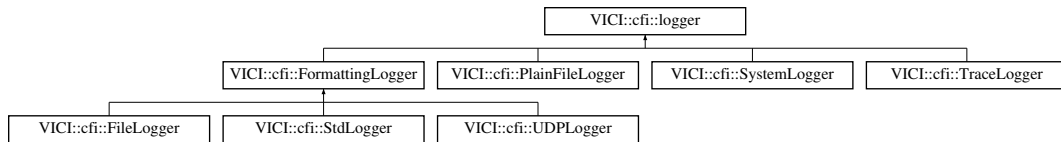
- `log.cpp`

8.69 VICI::cfi::logger Class Reference

An abstract base class used by the log stream to write logs.

```
#include <vici/log.h>
```

Inheritance diagram for VICI::cfi::logger:



Public Member Functions

- virtual `~logger ()`
Destructor.
- virtual int `log (Severity sev, csr line)=0`
log to the unnamed log
- virtual int `log (csr logName, Severity sev, csr line)=0`
log to the named log

Protected Member Functions

- `logger ()`
Constructor.

Protected Attributes

- `Semaphore sem4`
Mutual exclusion lock.

8.69.1 Detailed Description

An abstract base class used by the log stream to write logs.

This is the API that libraries must implement to be used as a logging stream.

8.69.2 Member Function Documentation

8.69.2.1 virtual int VICI::cfi::logger::log (Severity sev, csr line) [pure virtual]

log to the unnamed log

Parameters

<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implemented in [VICI::cfi::SystemLogger](#), [VICI::cfi::TraceLogger](#), [VICI::cfi::UDPLogger](#), [VICI::cfi::PlainFileLogger](#), [VICI::cfi::FileLogger](#), and [VICI::cfi::StdLogger](#).

8.69.2.2 `virtual int VICI::cfi::logger::log (csr logName, Severity sev, csr line)` [pure virtual]

log to the named log

Parameters

<i>logName</i>	the name of the log stream to write to
<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implemented in [VICI::cfi::SystemLogger](#), [VICI::cfi::TraceLogger](#), [VICI::cfi::UDPLogger](#), [VICI::cfi::PlainFileLogger](#), [VICI::cfi::FileLogger](#), and [VICI::cfi::StdLogger](#).

The documentation for this class was generated from the following file:

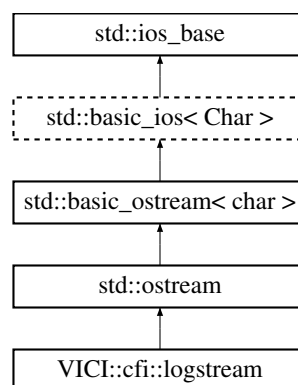
- [log.h](#)

8.70 VICI::cfi::logstream Class Reference

A stream class specialized for logging.

```
#include <vici/log.h>
```

Inheritance diagram for VICI::cfi::logstream:

**Public Member Functions**

- virtual [~logstream \(\)](#)
destructor
- void [severity \(Severity sev\)](#)
Set the severity for the current message.

Static Public Member Functions

- static `logstream & instance ()`
Get an instance of the unnamed stream.
- static `logstream & instance (csr name)`
Get an instance of a named stream.

Protected Member Functions

- `logstream (std::streambuf *buf)`
Static pointer to only instance of the anonymous object.

8.70.1 Detailed Description

A stream class specialized for logging.

There is an unnamed stream that is the default, and named streams for special purposes. The streams pick up their characteristics from [XINI](#) configuration file:

- `//LOG` the logging section of the configuration file
- `//LOG/[name]` section for a named log
- `//LOG/type` is one of `cout`, `cerr`, `clog`, `file`, `trace` or `syslog`.
- `//LOG/file` specifies the file name for file or trace logs
- `//LOG/ident` specifies the identifier for syslog logs

8.70.2 Constructor & Destructor Documentation

8.70.2.1 `logstream::logstream (std::streambuf * buf)` `[protected]`

Static pointer to only instance of the anonymous object.

Single instances of named streams. Private constructor.

Parameters

<code>buf</code>	pointer to a standard stream buffer
------------------	-------------------------------------

8.70.3 Member Function Documentation

8.70.3.1 `logstream & logstream::instance ()` `[static]`

Get an instance of the unnamed stream.

Returns

the instance of the anonymous log stream

8.70.3.2 `logstream & logstream::instance (csr name)` `[static]`

Get an instance of a named stream.

Parameters

<i>name</i>	name of the log stream used in the config file
-------------	--

Returns

the instance of the named log stream

8.70.3.3 void logstream::severity (Severity sev)

Set the severity for the current message.

Parameters

<i>sev</i>	new severity to use for the following messages
------------	--

The documentation for this class was generated from the following files:

- [log.h](#)
- [log.cpp](#)

8.71 VICI::gth::LuaScript Class Reference

Provides a wrapper for a lua_State object.

```
#include <vici/libgth.h>
```

Public Member Functions

- [LuaScript](#) (csr chunk)
Constructor.
- [~LuaScript](#) ()
Destructor.
- `std::string call` (csr cmnd, const [ParamList](#) ¶ms)
Run a function in the Lua chunk.

Static Public Member Functions

- static int [luaVersion](#) ()
Allow test scripts to check if Lua is installed (>0) and the version (51, 53)

8.71.1 Detailed Description

Provides a wrapper for a lua_State object.

The documentation for this class was generated from the following files:

- [libgth.h](#)
- [libgth.cpp](#)

8.72 VICI::Metrics Class Reference

A class to encapsulate measurements.

```
#include <vici/libgui.h>
```

Static Public Member Functions

- static int [ppm](#) (int m=1)
return the number of pixels for something m ems wide.

8.72.1 Detailed Description

A class to encapsulate measurements.

This class provides a method that should be used when setting the size of widgets. It provides the size of the letter 'm' in pixels and should therefore provide sizes that are somewhat independent of the physical resolution of the screen.

The documentation for this class was generated from the following files:

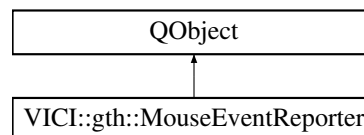
- [libgui.h](#)
- [libgui.cpp](#)

8.73 VICI::gth::MouseEventReporter Class Reference

Provides a means of getting scene coordinates in a QGraphicsView.

```
#include <vici/libgth.h>
```

Inheritance diagram for VICI::gth::MouseEventReporter:



Public Member Functions

- [MouseEventReporter](#) (QGraphicsView *v)
Constructor.

Protected Member Functions

- bool [eventFilter](#) (QObject *obj, QEvent *event)
Report position of mouse click in a graphics view.

8.73.1 Detailed Description

Provides a means of getting scene coordinates in a QGraphicsView.

The documentation for this class was generated from the following file:

- [libgth.h](#)

8.74 VICI::cdi::NullTrace Class Reference

A class for dummy trace objects.

```
#include <vici/trace.h>
```

Public Member Functions

- [NullTrace](#) ()
Constructor.
- `template<class T >`
[NullTrace](#) & `operator<<` (T x)
Noop version of stream function.

8.74.1 Detailed Description

A class for dummy trace objects.

A dummy object is required so that stream syntax can be used without causing a problem if tracing is turned off.

This is used by the [TRACE\(level\)](#) macro.

The documentation for this class was generated from the following file:

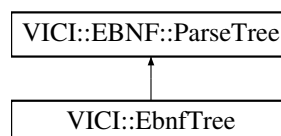
- [trace.h](#)

8.75 VICI::EBNF::ParseTree Class Reference

A type for the parsed form of [EBNF](#).

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

Inheritance diagram for VICI::EBNF::ParseTree:



Public Member Functions

- `virtual` `~ParseTree` ()
virtual destructor

8.75.1 Detailed Description

A type for the parsed form of [EBNF](#).

A type for the parsed form of [EBNF](#) used to pass the results of parsing the command's options [EBNF](#) to the syntax library without all the details leaking into all other modules.

The documentation for this class was generated from the following file:

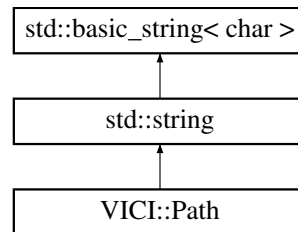
- [vici.h](#)

8.76 VICI::Path Class Reference

Manipulate path strings.

```
#include <vici/stringy.h>
```

Inheritance diagram for VICI::Path:



Public Member Functions

- [Path](#) ()
Constructor.
- [Path](#) (csr path)
Constructor.
- [Path & append](#) (csr name)
Append aname to a path.
- [Path & appendExt](#) (csr ext)
Append an extension to this.
- [Path & up](#) ()
shorten this by a level
- bool [absolute](#) () const
- std::string [base](#) () const
- std::string [dir](#) () const
- std::string [ext](#) () const
- std::string [name](#) () const
- std::string [noExt](#) () const
- void [split](#) (std::vector< std::string > &dirs) const
break into separate dirs
- bool [operator==](#) (const [Path](#) &x) const
test for equality
- bool [isChildOf](#) (const [Path](#) &p) const
test for equality
- [Path](#) & [operator=](#) (const char *)
assignment
- [Path](#) & [operator=](#) (const std::string &)
assignment
- [operator const char *](#) () const
type conversion
- bool [defined](#) () const
type conversion

Static Public Member Functions

- static bool `check` (const std::string &path)
checks for valid string
- static void `setSpacePolicy` (bool allowSpaces=false)
set policy for spaces in file names

8.76.1 Detailed Description

Manipulate path strings.

8.76.2 Constructor & Destructor Documentation

8.76.2.1 Path::Path (csr path) [explicit]

Constructor.

normalizes out any surplus /'s

Parameters

<i>path</i>	path to a file or directory
-------------	-----------------------------

8.76.3 Member Function Documentation

8.76.3.1 bool Path::absolute () const

Returns

true if its an absolute path

8.76.3.2 Path & Path::append (csr name)

Append aname to a path.

Parameters

<i>name</i>	append a / and name to this
-------------	-----------------------------

Returns

reference to new value of this

8.76.3.3 Path & Path::appendExt (csr ext)

Append an extension to this.

Parameters

<i>ext</i>	the extension to append.
------------	--------------------------

Returns

reference to new value of this

8.76.3.4 `string Path::base () const`**Returns**

everything after last /

8.76.3.5 `bool Path::defined () const`

type conversion

true if it has a value

8.76.3.6 `string Path::dir () const`**Returns**

everything up to last /

8.76.3.7 `string Path::ext () const`**Returns**

extension part of file name

8.76.3.8 `bool Path::isChildOf (const Path & p) const`

test for equality

test for equality true if this is a subdir of p

8.76.3.9 `string Path::name () const`**Returns**

base without the extension

8.76.3.10 `string Path::noExt () const`**Returns**

everything but extension

8.76.3.11 `void Path::split (std::vector< std::string > & dirs) const`

break into separate dirs

Parameters

<i>dirs</i>	The directory parts of the file name returned.
-------------	--

8.76.3.12 Path & Path::up ()

shorten this by a level

Returns

reference to new value of this

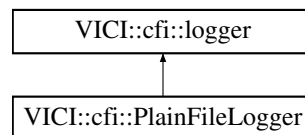
The documentation for this class was generated from the following files:

- [stringy.h](#)
- [stringy.cpp](#)

8.77 VICI::cfi::PlainFileLogger Class Reference

```
#include <vici/log.h>
```

Inheritance diagram for VICI::cfi::PlainFileLogger:



Public Member Functions

- [PlainFileLogger](#) (const [Path](#) &fname)
Constructor.
- int [log](#) ([Severity](#), [csr](#) line)
log to the unnamed log
- int [log](#) ([csr](#) logName, [Severity](#), [csr](#) line)
log to the named log

Protected Attributes

- std::ofstream [f](#)
The logging stream.

Additional Inherited Members

8.77.1 Detailed Description

Writes log messages to a file without adding any formatting.

8.77.2 Constructor & Destructor Documentation

8.77.2.1 PlainFileLogger::PlainFileLogger (const Path & fname)

Constructor.

Parameters

<i>fname</i>	the name of the log file.
--------------	---------------------------

8.77.3 Member Function Documentation

8.77.3.1 int PlainFileLogger::log (Severity sev, csr line) [virtual]

log to the unnamed log

Parameters

<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

8.77.3.2 int PlainFileLogger::log (csr logName, Severity sev, csr line) [virtual]

log to the named log

Parameters

<i>logName</i>	the name of the log stream to write to
<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

The documentation for this class was generated from the following files:

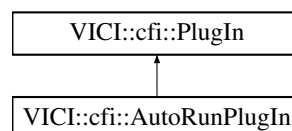
- [log.h](#)
- [log.cpp](#)

8.78 VICI::cfi::PlugIn Class Reference

Base class for objects loaded from dynamically loaded libraries.

```
#include <vici/plugin.h>
```

Inheritance diagram for VICI::cfi::PlugIn:



Public Member Functions

- virtual `~PlugIn ()`
Destructor.

Friends

- class **PlugInMgr**

8.78.1 Detailed Description

Base class for objects loaded from dynamically loaded libraries.

The documentation for this class was generated from the following files:

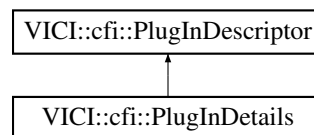
- [plugin.h](#)
- [plugin.cpp](#)

8.79 VICI::cfi::PlugInDescriptor Struct Reference

Descriptor for plug-ins that can be used by the application.

```
#include <plugin.h>
```

Inheritance diagram for VICI::cfi::PlugInDescriptor:



Public Attributes

- `std::string pluginFamily`
The type of the plug-in.
- `std::string name`
The name as in the xini config.
- `std::string description`
Some description for users.
- `std::string version`
Should match the defined VICI_PLUGIN_VERSION.
- `VICI::Path library`
The shared library's path.
- `bool onDemand`
True if the library is loaded on demand.
- `bool autoRun`
True if its run automatically on start.

8.79.1 Detailed Description

Descriptor for plug-ins that can be used by the application.

The documentation for this struct was generated from the following file:

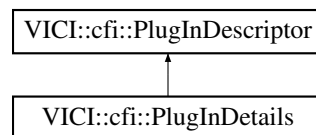
- [plugin.h](#)

8.80 VICI::cfi::PlugInDetails Struct Reference

Details of plug-ins as provided by the loaded library.

```
#include <plugin.h>
```

Inheritance diagram for VICI::cfi::PlugInDetails:



Public Member Functions

- [PlugInDetails \(\)](#)
Constructor to initialise variables.

Public Attributes

- [PlugInFactory * factory](#)
Construct an instance of the plug-in.
- bool [loadable](#)
False if the library cannot be loaded.

8.80.1 Detailed Description

Details of plug-ins as provided by the loaded library.

The documentation for this struct was generated from the following files:

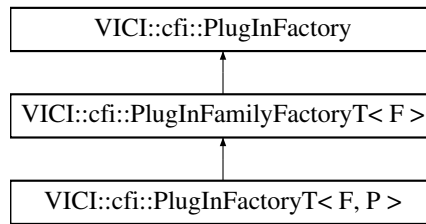
- [plugin.h](#)
- [plugin.cpp](#)

8.81 VICI::cfi::PlugInFactory Class Reference

Base class for factories that create plug-in objects.

```
#include <vici/plugin.h>
```

Inheritance diagram for VICI::cfi::PlugInFactory:



Public Member Functions

- virtual [~PlugInFactory](#) ()
Destructor.
- virtual [PlugIn * make](#) ()=0
Construct a plug-in object.

8.81.1 Detailed Description

Base class for factories that create plug-in objects.

8.81.2 Member Function Documentation

8.81.2.1 virtual [PlugIn*](#) [VICI::cfi::PlugInFactory::make](#) () [pure virtual]

Construct a plug-in object.

Returns

Pointer to new plug-in object

Implemented in [VICI::cfi::PlugInFactoryT< F, P >](#), and [VICI::cfi::PlugInFamilyFactoryT< F >](#).

The documentation for this class was generated from the following file:

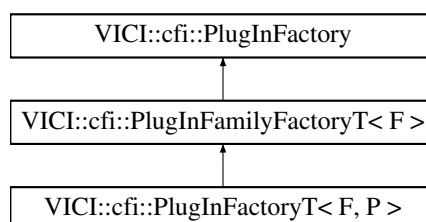
- [plugin.h](#)

8.82 VICI::cfi::PlugInFactoryT< F, P > Class Template Reference

Template class for factories.

```
#include <vici/plugin.h>
```

Inheritance diagram for [VICI::cfi::PlugInFactoryT< F, P >](#):



Public Member Functions

- virtual [~PlugInFactoryT](#) ()
Destructor.
- virtual F * [make](#) ()
Construct a plug-in object.

8.82.1 Detailed Description

```
template<class F, class P>class VICI::cfi::PlugInFactoryT< F, P >
```

Template class for factories.

[PlugIn](#) P must be derived from [PlugInFamily](#) F

8.82.2 Member Function Documentation

8.82.2.1 `template<class F, class P > virtual F* VICI::cfi::PlugInFactoryT< F, P >::make () [inline], [virtual]`

Construct a plug-in object.

Returns

Pointer to new plug-in object

Implements [VICI::cfi::PlugInFamilyFactoryT< F >](#).

The documentation for this class was generated from the following file:

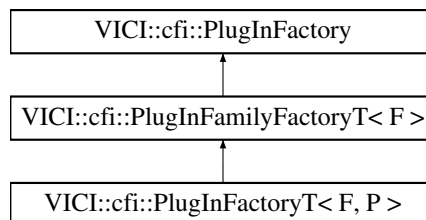
- [plugin.h](#)

8.83 VICI::cfi::PlugInFamilyFactoryT< F > Class Template Reference

Template base class for families of plug-in factories.

```
#include <vici/plugin.h>
```

Inheritance diagram for [VICI::cfi::PlugInFamilyFactoryT< F >](#):



Public Member Functions

- virtual F * [make](#) ()=0
Construct a plug-in object.

8.83.1 Detailed Description

```
template<class F>class VICI::cfi::PlugInFamilyFactoryT< F >
```

Template base class for families of plug-in factories.

Each family of plug-in objects will have its own API that is specified by the application and implemented by the plug-in library. This template allows the [PlugInMgr](#) to return an object of the correct type without having knowledge of that type built in.

8.83.2 Member Function Documentation

8.83.2.1 `template<class F> virtual F* VICI::cfi::PlugInFamilyFactoryT< F >::make () [pure virtual]`

Construct a plug-in object.

Returns

Pointer to new plug-in object

Implements [VICI::cfi::PlugInFactory](#).

Implemented in [VICI::cfi::PlugInFactoryT< F, P >](#).

The documentation for this class was generated from the following file:

- [plugin.h](#)

8.84 VICI::cfi::PlugInLib Class Reference

Manages an instance of a dynamically loaded shared library.

```
#include <vici/plugin.h>
```

Public Types

- enum [Mode](#) { [AutoRun](#), [OnDemand](#), [StayResident](#) }
Control lifetime of the library.

Public Member Functions

- [PlugInLib](#) (const [Path](#) &libPath)
Constructor.
- [~PlugInLib](#) ()
Destructor.
- void [setMode](#) ([Mode](#) m)
Set the lifetime for a plug-in.
- [Mode](#) [getMode](#) ()
Get the mode for the entire library.
- bool [loaded](#) ()
Check if successfully loaded the library.
- void [startUsage](#) ()
Indicate that the library is being used.
- void [endUsage](#) ()

- *Advise that a plug-in is no longer using the library.*
- bool `inUse ()`
Test if the library is in use.
- const `VICI::Path & getPath ()`
Get the path to the library.

8.84.1 Detailed Description

Manages an instance of a dynamically loaded shared library.

This object is responsible for loading and unloading the library. It is also responsible for calling the initialisation and finalisation code within the library using the two extern "C" functions.

Note that a library may have more than one plug-in class defined, and these may even have different lifetimes.

8.84.2 Member Enumeration Documentation

8.84.2.1 enum VICI::cfi::PlugInLib::Mode

Control lifetime of the library.

Enumerator

- AutoRun*** The library can be removed after initialisation.
- OnDemand*** The library can be removed once its unused.
- StayResident*** The library is only removed on program termination.

8.84.3 Constructor & Destructor Documentation

8.84.3.1 PlugInLib::PlugInLib (const Path & libPath)

Constructor.

Parameters

<i>libPath</i>	The path to the shared library, from xini config
----------------	--

8.84.4 Member Function Documentation

8.84.4.1 Mode VICI::cfi::PlugInLib::getMode () [inline]

Get the mode for the entire library.

Returns

The lifetime mode.

8.84.4.2 const VICI::Path& VICI::cfi::PlugInLib::getPath () [inline]

Get the path to the library.

Returns

the library path

8.84.4.3 `bool VICI::cfi::PlugInLib::inUse () [inline]`

Test if the library is in use.

Returns

True if a plug-in is still in active use.

8.84.4.4 `bool PlugInLib::loaded ()`

Check if successfully loaded the library.

Returns

true if loaded OK

8.84.4.5 `void PlugInLib::setMode (Mode m)`

Set the lifetime for a plug-in.

This value is combined with the existing values to get a lifetime value for the overall library

Parameters

<i>m</i>	The mode for the plug-in
----------	--------------------------

The documentation for this class was generated from the following files:

- [plugin.h](#)
- [plugin.cpp](#)

8.85 VICI::cfi::PlugInMgr Class Reference

Manage the handling of plug-in shared libraries.

```
#include <vici/plugin.h>
```

Public Member Functions

- void [load](#) (*csr prog*)
Load the plug-ins listed in the xini config.
- void [regn](#) (const [PlugInDetails](#) &details)
Allow a plug-in to register itself.
- void [getDetails](#) (std::vector< [PlugInDescriptor](#) > &plugins)
Get list of plug ins.
- void [release](#) ([PlugInLib](#) *)
Release a library.
- void [shutdown](#) ()
Unload all the libraries prior to exiting the program.
- template<class [PlugInFamily](#) >
std::unique_ptr< [PlugInFamily](#) > [getPlugin](#) (*csr name*)
Get a plug-in.

Static Public Member Functions

- static [PlugInMgr](#) & [instance](#) ()
Get a reference to the singleton object.

8.85.1 Detailed Description

Manage the handling of plug-in shared libraries.

8.85.2 Member Function Documentation

8.85.2.1 void PlugInMgr::load (*csr prog*)

Load the plug-ins listed in the xini config.

Parameters

<i>prog</i>	The name used to identify the set of plugins for this program.
-------------	--

The documentation for this class was generated from the following files:

- [plugin.h](#)
- [plugin.cpp](#)

8.86 VICI::cfi::ProcessOwner Class Reference

Interface that allows the owner of a child process to be notified.

```
#include <vici/proc.h>
```

Public Member Functions

- virtual [~ProcessOwner](#) ()
Destructor.
- virtual void [processTerminated](#) ([AbstractChildProcess](#) *cp)=0
Notification that process terminated.

8.86.1 Detailed Description

Interface that allows the owner of a child process to be notified.

8.86.2 Member Function Documentation

8.86.2.1 virtual void VICI::cfi::ProcessOwner::processTerminated ([AbstractChildProcess](#) * *cp*) [pure virtual]

Notification that process terminated.

Parameters

<i>cp</i>	The child process that ended.
-----------	-------------------------------

The documentation for this class was generated from the following file:

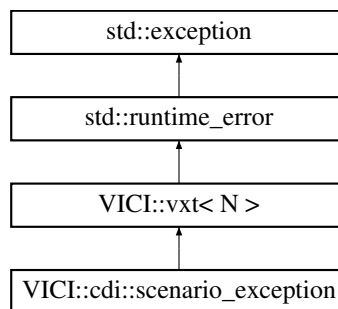
- [proc.h](#)

8.87 VICI::cdi::scenario_exception Class Reference

throw this to abandon an entire scenario

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::scenario_exception:



Public Member Functions

- [scenario_exception](#) ()
Constructor.

Additional Inherited Members

8.87.1 Detailed Description

throw this to abandon an entire scenario

The documentation for this class was generated from the following file:

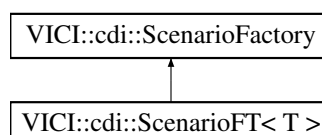
- [testmgr.h](#)

8.88 VICI::cdi::ScenarioFactory Class Reference

Define a type for scenario factories.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::ScenarioFactory:



Public Member Functions

- [ScenarioFactory](#) ()
Constructor.
- virtual [~ScenarioFactory](#) ()
Destructor.
- virtual [AbstractScenario](#) * [make](#) (csr name)=0
Create an scenario object.

8.88.1 Detailed Description

Define a type for scenario factories.

The documentation for this class was generated from the following file:

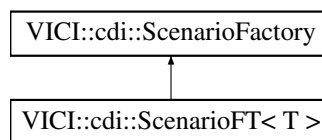
- [testmgr.h](#)

8.89 VICI::cdi::ScenarioFT< T > Class Template Reference

Responsible for creating a scenario of some type.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::ScenarioFT< T >:



Public Member Functions

- [AbstractScenario](#) * [make](#) (csr name)
Create a scenario object.

8.89.1 Detailed Description

```
template<class T>class VICI::cdi::ScenarioFT< T >
```

Responsible for creating a scenario of some type.

8.89.2 Member Function Documentation

8.89.2.1 `template<class T > AbstractScenario* VICI::cdi::ScenarioFT< T >::make (csr name) [inline], [virtual]`

Create a scenario object.

Returns

Pointer to a new scenario object.

Implements [VICI::cdi::ScenarioFactory](#).

The documentation for this class was generated from the following file:

- [testmgr.h](#)

8.90 VICI::cdi::ScenarioResults Struct Reference

Responsible for storing the results of testing for a scenario.

```
#include <vici/testmgr.h>
```

Public Member Functions

- [ScenarioResults](#) ()
Constructor.

Public Attributes

- int [mNumTests](#)
total number of tests
- int [mNumErrors](#)
number of errors reported
- bool [failure](#)
true if scenario tests failed

8.90.1 Detailed Description

Responsible for storing the results of testing for a scenario.

The documentation for this struct was generated from the following file:

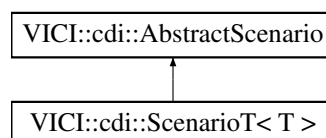
- [testmgr.h](#)

8.91 VICI::cdi::ScenarioT< T > Class Template Reference

Responsible for installing a factory to create scenarios of the required type.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::ScenarioT< T >:



Public Member Functions

- [ScenarioT](#) ([csr name](#))
Constructor.

Static Public Member Functions

- static void [install](#) ([csr nm](#))
Install a scenario factor into the [Tester](#).

Additional Inherited Members

8.91.1 Detailed Description

```
template<class T>class VICI::cdi::ScenarioT< T >
```

Responsible for installing a factory to create scenarios of the required type.

8.91.2 Constructor & Destructor Documentation

8.91.2.1 `template<class T> VICI::cdi::ScenarioT< T >::ScenarioT (csr name) [inline],[explicit]`

Constructor.

Parameters

<i>name</i>	The name of the scenario.
-------------	---------------------------

8.91.3 Member Function Documentation

8.91.3.1 `template<class T> static void VICI::cdi::ScenarioT< T >::install (csr nm) [inline],[static]`

Install a scenario factor into the [Tester](#).

Parameters

<i>nm</i>	The name of the scenario
-----------	--------------------------

The documentation for this class was generated from the following file:

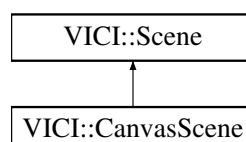
- [testmgr.h](#)

8.92 VICI::Scene Class Reference

A wrapper for QGraphicsScene class.

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

Inheritance diagram for VICI::Scene:



Public Member Functions

- virtual [~Scene](#) ()
virtual destructor

8.92.1 Detailed Description

A wrapper for QGraphicsScene class.

A wrapper for QGraphicsScene class so that both [Symbol](#) and [Canvas](#) can refer to the same scene objects.

The documentation for this class was generated from the following file:

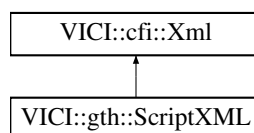
- [vici.h](#)

8.93 VICI::gth::ScriptXML Class Reference

Provides an interface to the script XML file.

```
#include <vici/libgth.h>
```

Inheritance diagram for VICI::gth::ScriptXML:



Public Member Functions

- [ScriptXML](#) (const [VICI::Path](#) &, bool [create](#))
Constructor.
- [~ScriptXML](#) ()
Destructor.
- void [getWindows](#) ([ParamList](#) &)
Get a list of windows from the XML script.
- void [getTestsForWindow](#) ([csr](#) wnm, [ListOfLists](#) &)
Get a list of test cases for a window.
- void [getTestActions](#) ([csr](#) tnm, [ConDes](#), std::vector< [TestAction](#) > &)
Get the test actions for a test case constructor or destructor.
- void [getScripts](#) (std::map< std::string, std::string > &)
Get the script for a window.
- void [addExecFilePath](#) ([csr](#) fileName)
Insert the executable's path into the script.
- void [addTest](#) ([csr](#) testName, bool unreferenced)
Add the name of test cases that are [not] referenced in the script.
- void [addWindow](#) ([csr](#) windowName, bool unreferenced)
Add the name of windows that are [not] referenced in the script.
- void [addWidget](#) ([csr](#) widgetName, [csr](#) widgetType)
Update the list of widgets and their types.
- void [addWidgetType](#) ([csr](#) widgetType, const [ListOfLists](#) &)
Update description of commands for a widget type.

Additional Inherited Members

8.93.1 Detailed Description

Provides an interface to the script XML file.

The documentation for this class was generated from the following files:

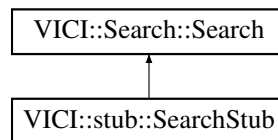
- [libgth.h](#)
- [script.cpp](#)

8.94 VICI::Search::Search Class Reference

Allow the user to search for, and organise commands.

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

Inheritance diagram for VICI::Search::Search:



Public Member Functions

- virtual [~Search](#) ()
virtual destructor
- virtual void [show](#) ()=0
make the search window visible
- virtual void [setNode](#) ([NodeId](#))=0
set the identity of the current node

8.94.1 Detailed Description

Allow the user to search for, and organise commands.

The facade for the [Search](#) library.

The implementation will present a window in which the user can search for commands, and in which commands can be organised.

The documentation for this class was generated from the following file:

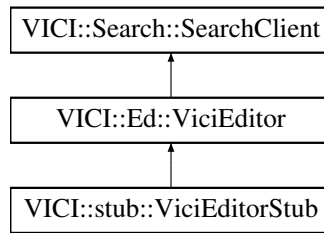
- [vici.h](#)

8.95 VICI::Search::SearchClient Class Reference

The interface that must implemented for clients of [Search](#).

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

Inheritance diagram for VICI::Search::SearchClient:



Public Member Functions

- virtual `~SearchClient()`
virtual destructor
- virtual void `selectedCommand(Nodeld n, csr c)=0`
this function is called when the user selects a command.

8.95.1 Detailed Description

The interface that must implemented for clients of [Search](#).

The client of [Search](#) will implement this interface so that it can be informed when the user selects a command.

8.95.2 Member Function Documentation

8.95.2.1 virtual void `VICI::Search::SearchClient::selectedCommand(Nodeld n, csr c)` [pure virtual]

this function is called when the user selects a command.

Parameters

<code>c</code>	the selected command
<code>n</code>	the identifier for the node

Implemented in [VICI::stub::ViciEditorStub](#).

The documentation for this class was generated from the following file:

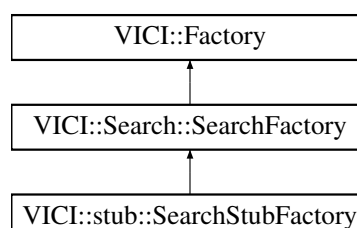
- [vici.h](#)

8.96 VICI::Search::SearchFactory Class Reference

An abstract factory for making an instance of [Search](#).

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

Inheritance diagram for `VICI::Search::SearchFactory`:



Public Member Functions

- virtual `~SearchFactory ()`
virtual destructor
- virtual `Search * makeSearch (Window *w, SearchClient *sc)=0`
create an instance of the Search class

8.96.1 Detailed Description

An abstract factory for making an instance of `Search`.

The implementation will create either the production version, or a stub, or a test version of the `Search` class.

8.96.2 Member Function Documentation

8.96.2.1 `virtual Search* VICI::Search::SearchFactory::makeSearch (Window * w, SearchClient * sc)` [pure virtual]

create an instance of the `Search` class

Parameters

<code>w</code>	the window to display into
<code>sc</code>	the client to report the results to

Implemented in `VICI::stub::SearchStubFactory`.

The documentation for this class was generated from the following file:

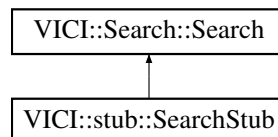
- [vici.h](#)

8.97 VICI::stub::SearchStub Class Reference

A stub version of the `Search` module.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for `VICI::stub::SearchStub`:



Public Member Functions

- `SearchStub (Window *w, VICI::Search::SearchClient *c)`
Constructor.
- virtual void `show ()`
make the search window visible
- virtual void `setNode (NodeId)`
set the identity of the current node

8.97.1 Detailed Description

A stub version of the [Search](#) module.

8.97.2 Constructor & Destructor Documentation

8.97.2.1 SearchStub::SearchStub (Window * w, VICI::Search::SearchClient * c)

Constructor.

Parameters

<i>w</i>	The window to display into.
<i>c</i>	The module that get the search results.

The documentation for this class was generated from the following files:

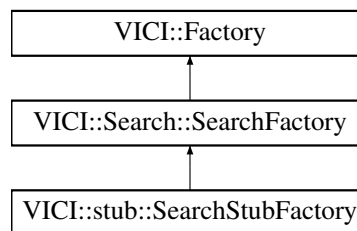
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.98 VICI::stub::SearchStubFactory Class Reference

A factory for creating stub instances of [Search](#) objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::SearchStubFactory:



Public Member Functions

- virtual [Search::Search](#) * [makeSearch](#) (Window *, [Search::SearchClient](#) *)
create an instance of the [Search](#) class

8.98.1 Detailed Description

A factory for creating stub instances of [Search](#) objects.

8.98.2 Member Function Documentation

8.98.2.1 Search::Search * SearchStubFactory::makeSearch (Window * w, Search::SearchClient * sc) [virtual]

create an instance of the [Search](#) class

Parameters

<code>w</code>	the window to display into
<code>sc</code>	the client to report the results to

Implements [VICI::Search::SearchFactory](#).

The documentation for this class was generated from the following files:

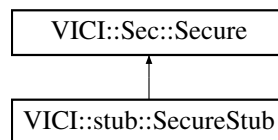
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.99 VICI::Sec::Secure Class Reference

Provide security for the scripts.

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

Inheritance diagram for VICI::Sec::Secure:



Public Member Functions

- virtual `~Secure()`
virtual destructor
- virtual bool `verifySignature(csrfilename)=0`
check to ensure we are allowed to open this file
- virtual void `addSignature(csrfilename)=0`
sign this file with the user's key

8.99.1 Detailed Description

Provide security for the scripts.

The facade for the security library.

The aim is to prevent users from running random scripts however this may be difficult to achieve. It may also conflict with SourceForge's terms and conditions. Hence this module will not be implemented for a while.

8.99.2 Member Function Documentation

8.99.2.1 virtual void VICI::Sec::Secure::addSignature (*csrfilename*) [pure virtual]

sign this file with the user's key

Parameters

<i>filename</i>	add a signature to the script.
-----------------	--------------------------------

Implemented in [VICI::stub::SecureStub](#).

8.99.2.2 virtual bool VICI::Sec::Secure::verifySignature (csr filename) [pure virtual]

check to ensure we are allowed to open this file

Parameters

<i>filename</i>	the script to verify
-----------------	----------------------

Returns

true if the signature is valid

Implemented in [VICI::stub::SecureStub](#).

The documentation for this class was generated from the following file:

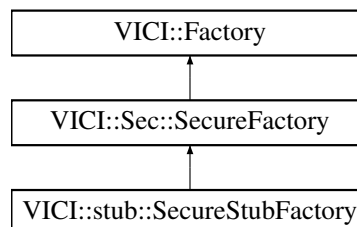
- [vici.h](#)

8.100 VICI::Sec::SecureFactory Class Reference

An abstract factory for making an instance of [Secure](#).

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

Inheritance diagram for VICI::Sec::SecureFactory:



Public Member Functions

- virtual [~SecureFactory](#) ()
virtual destructor
- virtual [Secure](#) * [makeSecure](#) ()=0
create an instance of the [Secure](#) class

8.100.1 Detailed Description

An abstract factory for making an instance of [Secure](#).

The implementation will create either the production version, or a stub, or a test version of the [Secure](#) class.

8.100.2 Member Function Documentation

8.100.2.1 virtual Secure* VICI::Sec::SecureFactory::makeSecure () [pure virtual]

create an instance of the [Secure](#) class

return an instance of the [Secure](#) object.

Implemented in [VICI::stub::SecureStubFactory](#).

The documentation for this class was generated from the following file:

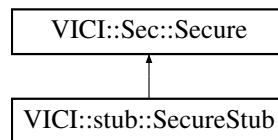
- [vici.h](#)

8.101 VICI::stub::SecureStub Class Reference

A stub version of the Secure module.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::SecureStub:



Public Member Functions

- virtual bool [verifySignature](#) (csr filename)
check to ensure we are allowed to open this file
- virtual void [addSignature](#) (csr filename)
sign this file with the user's key

8.101.1 Detailed Description

A stub version of the Secure module.

8.101.2 Member Function Documentation

8.101.2.1 void SecureStub::addSignature (csr filename) [virtual]

sign this file with the user's key

Parameters

<i>filename</i>	add a signature to the script.
-----------------	--------------------------------

Implements [VICI::Sec::Secure](#).

8.101.2.2 bool SecureStub::verifySignature (csr filename) [virtual]

check to ensure we are allowed to open this file

Parameters

<i>filename</i>	the script to verify
-----------------	----------------------

Returns

true if the signature is valid

Implements [VICI::Sec::Secure](#).

The documentation for this class was generated from the following files:

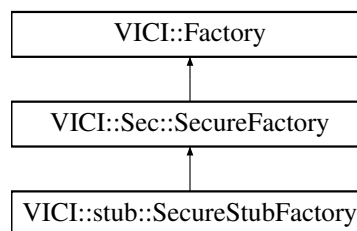
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.102 VICI::stub::SecureStubFactory Class Reference

A factory for creating stub instances of Security objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::SecureStubFactory:



Public Member Functions

- virtual [Sec::Secure](#) * [makeSecure](#) ()
create an instance of the Secure class

8.102.1 Detailed Description

A factory for creating stub instances of Security objects.

8.102.2 Member Function Documentation

8.102.2.1 [Sec::Secure](#) * [SecureStubFactory::makeSecure](#) () [virtual]

create an instance of the Secure class

return an instance of the Secure object.

Implements [VICI::Sec::SecureFactory](#).

The documentation for this class was generated from the following files:

- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.103 VICI::cfi::Semaphore Class Reference

A semaphore for managing exclusive access to resources across multiple processes.

```
#include <vici/ipc.h>
```

Public Member Functions

- [Semaphore](#) (char id)
Constructor.
- [~Semaphore](#) ()
Destructor.
- void [acquire](#) ()
wait until available
- void [release](#) ()
release exclusive lock

8.103.1 Detailed Description

A semaphore for managing exclusive access to resources across multiple processes.

Provides the ability to limit access to a resource to one process at a time.

The semaphore uses a second semaphore to manage its lifetime. The first process to create it creates the system wide semaphore, and the last to delete it removes the system wide semaphore.

8.103.2 Constructor & Destructor Documentation

8.103.2.1 Semaphore::Semaphore (char id)

Constructor.

Parameters

<i>id</i>	an identifier for the semaphore.
-----------	----------------------------------

The documentation for this class was generated from the following files:

- [ipc.h](#)
- [ipc.cpp](#)

8.104 VICI::cfi::SemaphoreLock Class Reference

Mutual exclusion lock.

```
#include <vici/ipc.h>
```

Public Member Functions

- [SemaphoreLock](#) ([Semaphore](#) &s)
Constructor which acquires a lock on the supplied semaphore.
- [~SemaphoreLock](#) ()
Destructor which releases the lock.

8.104.1 Detailed Description

Mutual exclusion lock.

Wraps the [Semaphore](#) so that it will be automatically released even if an exception is thrown.

8.104.2 Constructor & Destructor Documentation

8.104.2.1 SemaphoreLock::SemaphoreLock (Semaphore & s) [explicit]

Constructor which acquires a lock on the supplied semaphore.

Parameters

<code>s</code>	The semaphore being managed
----------------	-----------------------------

The documentation for this class was generated from the following files:

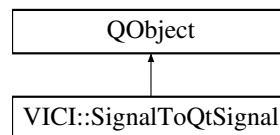
- [ipc.h](#)
- [ipc.cpp](#)

8.105 VICI::SignalToQtSignal Class Reference

A class to convert operating system signals into Qt signals.

```
#include <vici/libgui.h>
```

Inheritance diagram for VICI::SignalToQtSignal:



Signals

- void [notifySignal](#) ()
This signal is emitted when an operating signal arrives.

Public Member Functions

- void [emitSignal](#) ()
Called by signal handler when a signal arrives.

8.105.1 Detailed Description

A class to convert operating system signals into Qt signals.

This class provides a mechanism to safely handle operating system signals and generate a Qt signal. It uses a pipe to pass a byte to a thread. The Qt signal should be connected via a queued connection.

The documentation for this class was generated from the following files:

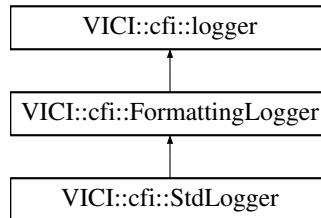
- [libgui.h](#)
- [libgui.cpp](#)

8.106 VICI::cfi::StdLogger Class Reference

Class for logging to the std output streams.

```
#include <vici/log.h>
```

Inheritance diagram for VICI::cfi::StdLogger:



Public Types

- enum [Stream](#) { **COUT**, **CERR**, **CLOG** }
Enumeration of the standard streams.

Public Member Functions

- [StdLogger](#) ([Stream](#) cc)
Constructor.
- int [log](#) ([Severity](#), [csr](#) line)
log to the unnamed log
- int [log](#) ([csr](#) logName, [Severity](#), [csr](#) line)
log to the named log

Additional Inherited Members

8.106.1 Detailed Description

Class for logging to the std output streams.

8.106.2 Constructor & Destructor Documentation

8.106.2.1 `VICI::cfi::StdLogger::StdLogger (Stream cc)` `[inline]`, `[explicit]`

Constructor.

Parameters

cc	enum to specify the stream to write to
----	--

8.106.3 Member Function Documentation

8.106.3.1 `int StdLogger::log (Severity sev, csr line)` `[virtual]`

log to the unnamed log

Parameters

<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

8.106.3.2 `int StdLogger::log (csr logName, Severity sev, csr line) [virtual]`

log to the named log

Parameters

<i>logName</i>	the name of the log stream to write to
<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

The documentation for this class was generated from the following files:

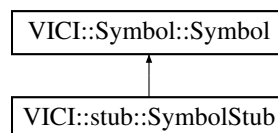
- [log.h](#)
- [log.cpp](#)

8.107 VICI::Symbol::Symbol Class Reference

Represents something that is drawn on the canvas.

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

Inheritance diagram for VICI::Symbol::Symbol:



Public Member Functions

- virtual `~Symbol ()`
virtual destructor
- virtual `Symbol * clone (SymbolOwner *owner)=0`
construct a copy of the symbol
- virtual void `setAttributes (SymbolAttributes &att)=0`
change the attributes of the symbol
- virtual bool `isCommand ()=0`

- check if the symbol represents a command object*

 - virtual void `draw` (`Scene *scene`, double `x`, double `y`, double `scale`)=0

display the object
- virtual void `attachCommand` (const `ArgList` &`args`)=0

Associate the symbol with a command.
- virtual void `setNodeid` (`Nodeid` `nid`)=0

Set the node id for the object.
- virtual `Style` `getStyle` ()=0

get the style for the symbol
- virtual void `setHighlight` (bool `x`)=0

show the symbol as selected

8.107.1 Detailed Description

Represents something that is drawn on the canvas.

The implementation will paint itself on the provided scene.

8.107.2 Member Function Documentation

8.107.2.1 virtual void VICI::Symbol::Symbol::attachCommand (const `ArgList` &`args`) [pure virtual]

Associate the symbol with a command.

The first entry becomes the label for the object. The entire argument list is assigned to the tool tip for the symbol.

Parameters

<code>args</code>	the command and its options
-------------------	-----------------------------

Implemented in [VICI::stub::SymbolStub](#).

8.107.2.2 virtual `Symbol*` VICI::Symbol::Symbol::clone (`SymbolOwner` *`owner`) [pure virtual]

construct a copy of the symbol

Parameters

<code>owner</code>	the object that owns the symbol
--------------------	---------------------------------

Implemented in [VICI::stub::SymbolStub](#).

8.107.2.3 virtual void VICI::Symbol::Symbol::draw (`Scene` *`scene`, double `x`, double `y`, double `scale`) [pure virtual]

display the object

Parameters

<code>scene</code>	the canvas on which to display the object.
<code>x</code>	the x coordinate on the scene
<code>y</code>	the y coordinate on the scene
<code>scale</code>	the scaling factor to use

Implemented in [VICI::stub::SymbolStub](#).

8.107.2.4 virtual Style VICI::Symbol::Symbol::getStyle () [pure virtual]

get the style for the symbol

Returns

the style of the symbol.

Implemented in [VICI::stub::SymbolStub](#).

8.107.2.5 virtual bool VICI::Symbol::Symbol::isCommand () [pure virtual]

check if the symbol represents a command object

Returns

true if the object is a command or choice

Implemented in [VICI::stub::SymbolStub](#).

8.107.2.6 virtual void VICI::Symbol::Symbol::setAttributes (SymbolAttributes & att) [pure virtual]

change the attributes of the symbol

This will cause the symbol to be redisplayed with the new attributes.

Parameters

<i>att</i>	the new attributes.
------------	---------------------

Implemented in [VICI::stub::SymbolStub](#).

8.107.2.7 virtual void VICI::Symbol::Symbol::setHighlight (bool x) [pure virtual]

show the symbol as selected

Parameters

<i>x</i>	True to show highlighted.
----------	---------------------------

Implemented in [VICI::stub::SymbolStub](#).

8.107.2.8 virtual void VICI::Symbol::Symbol::setNodeid (Nodeid nid) [pure virtual]

Set the node id for the object.

Parameters

<i>nid</i>	The node id to show on the symbol.
------------	------------------------------------

Implemented in [VICI::stub::SymbolStub](#).

The documentation for this class was generated from the following file:

- [vici.h](#)

8.108 VICI::Symbol::SymbolAttributes Class Reference

Holds the attributes of a symbol.

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


Public Member Functions

- [SymbolAttributes](#) ()
constructor

Public Attributes

- [Colour](#) `lineColour`
the colour for lines
- [Colour](#) `fillColour`
the colour to fill the symbol
- `std::string` `fillPattern`
the pattern to fill the symbol
- `std::string` `linePattern`
the pattern for the lines

8.108.1 Detailed Description

Holds the attributes of a symbol.

Holds the fill colour and pattern, and the line colour and pattern for a symbol. In a future release the use will be able to modify these if they have vision issues.

The documentation for this class was generated from the following file:

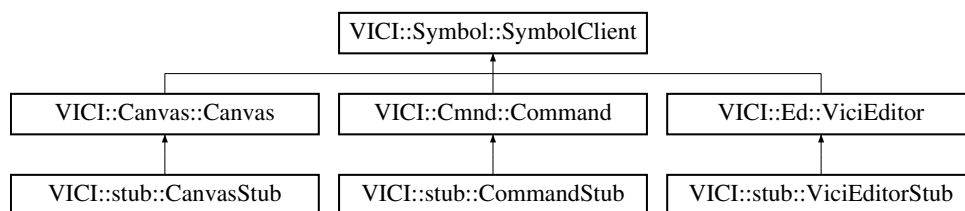
- [vici.h](#)

8.109 VICI::Symbol::SymbolClient Class Reference

An interface that is notified of changes to symbol manager.

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

Inheritance diagram for VICI::Symbol::SymbolClient:



Public Member Functions

- `virtual` `~SymbolClient` ()
virtual destructor
- `virtual void` `selection` (`Symbol *sym`)=0
notify that a symbol has been selected.
- `virtual void` `symbolAttr` (`SymbolAttributes &att`)=0
notify that the default symbol attributes have changed
- `virtual void` `textAttr` (`TextAttributes &att`)=0
notify that the default text attributes have changed

8.109.1 Detailed Description

An interface that is notified of changes to symbol manager.

This interface is implemented by objects that need to be notified when the user changes the default symbol of attributes.

8.109.2 Member Function Documentation

8.109.2.1 `virtual void VICI::Symbol::SymbolClient::selection (Symbol * sym) [pure virtual]`

notify that a symbol has been selected.

Parameters

<i>sym</i>	the symbol that was selected.
------------	-------------------------------

Todo do we need to notify of deselection ?

Implemented in [VICI::Cmnd::Command](#), [VICI::Canvas::Canvas](#), [VICI::stub::ViciEditorStub](#), [VICI::stub::CanvasStub](#), and [VICI::stub::CommandStub](#).

8.109.2.2 `virtual void VICI::Symbol::SymbolClient::symbolAttr (SymbolAttributes & att) [pure virtual]`

notify that the default symbol attributes have changed

Parameters

<i>att</i>	the new default attributes
------------	----------------------------

Implemented in [VICI::Cmnd::Command](#), [VICI::Canvas::Canvas](#), [VICI::stub::ViciEditorStub](#), [VICI::stub::CanvasStub](#), and [VICI::stub::CommandStub](#).

8.109.2.3 `virtual void VICI::Symbol::SymbolClient::textAttr (TextAttributes & att) [pure virtual]`

notify that the default text attributes have changed

Parameters

<i>att</i>	the new default text attributes.
------------	----------------------------------

Implemented in [VICI::Cmnd::Command](#), [VICI::Canvas::Canvas](#), [VICI::stub::ViciEditorStub](#), [VICI::stub::CanvasStub](#), and [VICI::stub::CommandStub](#).

The documentation for this class was generated from the following file:

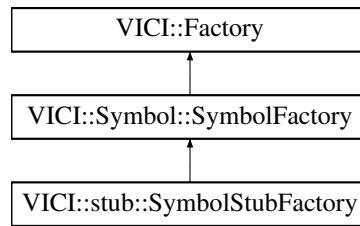
- [vici.h](#)

8.110 VICI::Symbol::SymbolFactory Class Reference

An abstract factory for making an instance of [SymbolMgr](#).

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

Inheritance diagram for VICI::Symbol::SymbolFactory:



Public Member Functions

- virtual `~SymbolFactory()`
virtual destructor
- virtual `SymbolMgr * makeSymbolMgr(Window *w)=0`
create an instance of the `SymbolMgr` class

8.110.1 Detailed Description

An abstract factory for making an instance of `SymbolMgr`.

The implementation will create either the production version, or a stub, or a test version of the `SymbolMgr` class.

8.110.2 Member Function Documentation

8.110.2.1 `virtual SymbolMgr* VICI::Symbol::SymbolFactory::makeSymbolMgr(Window * w) [pure virtual]`

create an instance of the `SymbolMgr` class

Parameters

<code>w</code>	the window to display into.
----------------	-----------------------------

Implemented in `VICI::stub::SymbolStubFactory`.

The documentation for this class was generated from the following file:

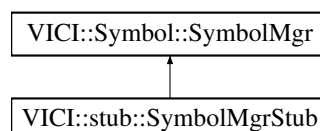
- [vici.h](#)

8.111 VICI::Symbol::SymbolMgr Class Reference

The facade for the symbol library.

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

Inheritance diagram for `VICI::Symbol::SymbolMgr`:



Public Member Functions

- virtual `~SymbolMgr()`

- virtual destructor*
- virtual void `addClient (SymbolClient *client)=0`
add a client for the symbol manager
- virtual `SymbolAttributes getDefaultAttr (Style s)=0`
get the attributes for a symbol of the given style
- virtual `TextAttributes getDefaultTextAttr ()=0`
get the current default text attributes
- virtual `Symbol * getSymbol (Style style, SymbolOwner *owner)=0`
get a new symbol of the specified style

8.111.1 Detailed Description

The facade for the symbol library.

The implementation of this class will provide access to the symbol user interface.

8.111.2 Member Function Documentation

8.111.2.1 virtual void `VICI::Symbol::SymbolMgr::addClient (SymbolClient * client)` [pure virtual]

add a client for the symbol manager

Parameters

<i>client</i>	the object to notify of user actions.
---------------	---------------------------------------

Implemented in [VICI::stub::SymbolMgrStub](#).

8.111.2.2 virtual `SymbolAttributes VICI::Symbol::SymbolMgr::getDefaultAttr (Style s)` [pure virtual]

get the attributes for a symbol of the given style

Parameters

<i>s</i>	the style of the symbol to get the attributes for.
----------	--

Returns

the attributes

Implemented in [VICI::stub::SymbolMgrStub](#).

8.111.2.3 virtual `TextAttributes VICI::Symbol::SymbolMgr::getDefaultTextAttr ()` [pure virtual]

get the current default text attributes

Returns

the attributes

Implemented in [VICI::stub::SymbolMgrStub](#).

8.111.2.4 virtual `Symbol* VICI::Symbol::SymbolMgr::getSymbol (Style style, SymbolOwner * owner)` [pure virtual]

get a new symbol of the specified style

Parameters

<i>style</i>	the style of the requested symbol
<i>owner</i>	the owner for the returned symbol

Returns

the new symbol

Implemented in [VICI::stub::SymbolMgrStub](#).

The documentation for this class was generated from the following file:

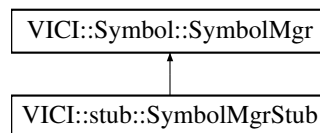
- [vici.h](#)

8.112 VICI::stub::SymbolMgrStub Class Reference

A stub version of the [Symbol](#) Manager.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::SymbolMgrStub:



Public Member Functions

- [SymbolMgrStub](#) ([Window](#) *w)
Costructor.
- virtual void [addClient](#) ([Symbol::SymbolClient](#) *)
add a client for the symbol manager
- virtual [Symbol::SymbolAttributes](#) [getDefaultAttr](#) ([Symbol::Style](#))
get the attributes for a symbol of the given style
- virtual [Symbol::TextAttributes](#) [getDefaultTextAttr](#) ()
get the current default text attributes
- virtual [Symbol::Symbol](#) * [getSymbol](#) ([Symbol::Style](#), [Symbol::SymbolOwner](#) *)
get a new symbol of the specified style

8.112.1 Detailed Description

A stub version of the [Symbol](#) Manager.

8.112.2 Constructor & Destructor Documentation

8.112.2.1 SymbolMgrStub::SymbolMgrStub (Window * w)

Costructor.

Parameters

<i>w</i>	The window to display into.
----------	-----------------------------

8.112.3 Member Function Documentation**8.112.3.1 void SymbolMgrStub::addClient (Symbol::SymbolClient * *client*) [virtual]**

add a client for the symbol manager

Parameters

<i>client</i>	the object to notify of user actions.
---------------	---------------------------------------

Implements [VICI::Symbol::SymbolMgr](#).

8.112.3.2 Symbol::SymbolAttributes SymbolMgrStub::getDefaultAttr (Symbol::Style *s*) [virtual]

get the attributes for a symbol of the given style

Parameters

<i>s</i>	the style of the symbol to get the attributes for.
----------	--

Returns

the attributes

Implements [VICI::Symbol::SymbolMgr](#).

8.112.3.3 Symbol::TextAttributes SymbolMgrStub::getDefaultTextAttr () [virtual]

get the current default text attributes

Returns

the attributes

Implements [VICI::Symbol::SymbolMgr](#).

8.112.3.4 Symbol::Symbol * SymbolMgrStub::getSymbol (Symbol::Style *style*, Symbol::SymbolOwner * *owner*) [virtual]

get a new symbol of the specified style

Parameters

<i>style</i>	the style of the requested symbol
<i>owner</i>	the owner for the returned symbol

Returns

the new symbol

Implements [VICI::Symbol::SymbolMgr](#).

The documentation for this class was generated from the following files:

- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.113 VICI::Symbol::SymbolOwner Class Reference

Represents something that is notified about events occurring on a symbol.

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

Public Member Functions

- virtual `~SymbolOwner()`
virtual destructor
- virtual void `selected()`=0
the object has become selected
- virtual void `opened()`=0
- virtual void `dragged(double x, double y)`=0
the object was moved

8.113.1 Detailed Description

Represents something that is notified about events occurring on a symbol.

The object that owns the symbol gets called via this interface when the user interacts with the symbol.

8.113.2 Member Function Documentation

8.113.2.1 `virtual void VICI::Symbol::SymbolOwner::dragged(double x, double y) [pure virtual]`

the object was moved

Parameters

<code>x</code>	the new x coordinate of the symbol
<code>y</code>	the new y coordinate of the symbol

8.113.2.2 `virtual void VICI::Symbol::SymbolOwner::opened() [pure virtual]`

the object was opened

The documentation for this class was generated from the following file:

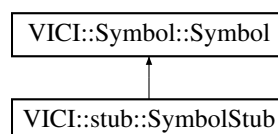
- [vici.h](#)

8.114 VICI::stub::SymbolStub Class Reference

A stub version of the [Symbol](#) class.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::SymbolStub:



Public Member Functions

- [SymbolStub](#) (bool command)
Constructor.
- virtual [Symbol](#) * [clone](#) ([VICI::Symbol::SymbolOwner](#) *)
construct a copy of the symbol
- virtual bool [isCommand](#) ()
check if the symbol represents a command object
- virtual void [setAttributes](#) ([VICI::Symbol::SymbolAttributes](#) &)
change the attributes of the symbol
- virtual void [draw](#) ([Scene](#) *, double x, double y, double scale)
display the object
- virtual void [attachCommand](#) (const [ArgList](#) &)
Associate the symbol with a command.
- virtual void [setNodeid](#) ([Nodeid](#))
Set the node id for the object.
- virtual [VICI::Symbol::Style](#) [getStyle](#) ()
get the style for the symbol
- virtual void [setHighlight](#) (bool)
show the symbol as selected

8.114.1 Detailed Description

A stub version of the [Symbol](#) class.

8.114.2 Constructor & Destructor Documentation

8.114.2.1 [VICI::stub::SymbolStub::SymbolStub](#) (bool *command*) `[inline]`

Constructor.

Parameters

<i>command</i>	True if the symbol represents a command.
----------------	--

8.114.3 Member Function Documentation

8.114.3.1 virtual void [VICI::stub::SymbolStub::attachCommand](#) (const [ArgList](#) & *args*) `[inline]`, `[virtual]`

Associate the symbol with a command.

The first entry becomes the label for the object. The entire argument list is assigned to the tool tip for the symbol.

Parameters

<i>args</i>	the command and its options
-------------	-----------------------------

Implements [VICI::Symbol::Symbol](#).

8.114.3.2 virtual [Symbol](#)* [VICI::stub::SymbolStub::clone](#) ([VICI::Symbol::SymbolOwner](#) * *owner*) `[inline]`,
`[virtual]`

construct a copy of the symbol

Parameters

<i>owner</i>	the object that owns the symbol
--------------	---------------------------------

Implements [VICI::Symbol::Symbol](#).

8.114.3.3 `virtual void VICI::stub::SymbolStub::draw (Scene * scene, double x, double y, double scale) [inline], [virtual]`

display the object

Parameters

<i>scene</i>	the canvas on which to display the object.
<i>x</i>	the x coordinate on the scene
<i>y</i>	the y coordinate on the scene
<i>scale</i>	the scaling factor to use

Implements [VICI::Symbol::Symbol](#).

8.114.3.4 `virtual VICI::Symbol::Style VICI::stub::SymbolStub::getStyle () [inline], [virtual]`

get the style for the symbol

Returns

the style of the symbol.

Implements [VICI::Symbol::Symbol](#).

8.114.3.5 `virtual bool VICI::stub::SymbolStub::isCommand () [inline], [virtual]`

check if the symbol represents a command object

Returns

true if the object is a command or choice

Implements [VICI::Symbol::Symbol](#).

8.114.3.6 `virtual void VICI::stub::SymbolStub::setAttributes (VICI::Symbol::SymbolAttributes & att) [inline], [virtual]`

change the attributes of the symbol

This will cause the symbol to be redisplayed with the new attributes.

Parameters

<i>att</i>	the new attributes.
------------	---------------------

Implements [VICI::Symbol::Symbol](#).

8.114.3.7 `virtual void VICI::stub::SymbolStub::setHighlight (bool x) [inline], [virtual]`

show the symbol as selected

Parameters

x	True to show highlighted.
---	---------------------------

Implements [VICI::Symbol::Symbol](#).

8.114.3.8 virtual void VICI::stub::SymbolStub::setNodeId (NodeId nid) [inline],[virtual]

Set the node id for the object.

Parameters

nid	The node id to show on the symbol.
-----	------------------------------------

Implements [VICI::Symbol::Symbol](#).

The documentation for this class was generated from the following file:

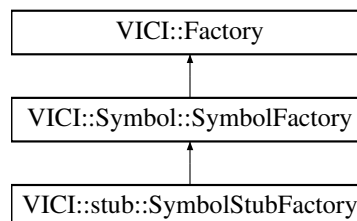
- [libifstubs.h](#)

8.115 VICI::stub::SymbolStubFactory Class Reference

A factory for creating stub instances of [Symbol](#) Manager objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::SymbolStubFactory:



Public Member Functions

- virtual [Symbol::SymbolMgr](#) * makeSymbolMgr (Window *)
create an instance of the SymbolMgr class

8.115.1 Detailed Description

A factory for creating stub instances of [Symbol](#) Manager objects.

8.115.2 Member Function Documentation

8.115.2.1 [Symbol::SymbolMgr](#) * SymbolStubFactory::makeSymbolMgr (Window * w) [virtual]

create an instance of the SymbolMgr class

Parameters

<i>w</i>	the window to display into.
----------	-----------------------------

Implements [VICI::Symbol::SymbolFactory](#).

The documentation for this class was generated from the following files:

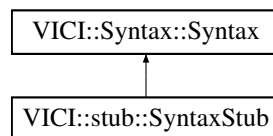
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.116 VICI::Syntax::Syntax Class Reference

Display a syntax chart of command options.

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

Inheritance diagram for VICI::Syntax::Syntax:



Public Member Functions

- virtual [~Syntax](#) ()
virtual destructor
- virtual void [show](#) (*csr s*)=0
display the syntax chart for the supplied ebnf

8.116.1 Detailed Description

Display a syntax chart of command options.

The definition of the facade for libsyntax.

The implementation will display a syntax chart of command options, allowing the user to navigate between non-terminal nodes of the [EBNF](#).

8.116.2 Member Function Documentation

8.116.2.1 virtual void [VICI::Syntax::Syntax::show](#) (*csr s*) [*pure virtual*]

display the syntax chart for the supplied ebnf

Parameters

<i>s</i>	the EBNF to build the chart for.
----------	--

Implemented in [VICI::stub::SyntaxStub](#).

The documentation for this class was generated from the following file:

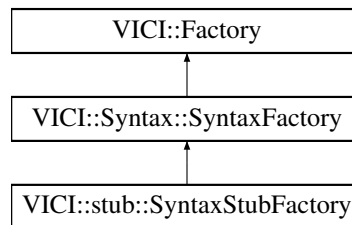
- [vici.h](#)

8.117 VICI::Syntax::SyntaxFactory Class Reference

An abstract factory for making an instance of [Syntax](#).

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

Inheritance diagram for VICI::Syntax::SyntaxFactory:



Public Member Functions

- virtual `~SyntaxFactory ()`
virtual destructor
- virtual `Syntax * makeSyntax (Window *w, EBNF::EBNF *ebnf)=0`
Create an instance of [Syntax](#).

8.117.1 Detailed Description

An abstract factory for making an instance of [Syntax](#).

An abstract factory for making an instance of [Syntax](#). The implementation will create either the production version, or a stub, or a test version of the [Syntax](#) class.

8.117.2 Member Function Documentation

8.117.2.1 `virtual Syntax* VICI::Syntax::SyntaxFactory::makeSyntax (Window * w, EBNF::EBNF * ebnf)` [pure virtual]

Create an instance of [Syntax](#).

Parameters

<i>w</i>	the window to display the chart in.
<i>ebnf</i>	pointer to the EBNF object for parsing the option parameters.

Returns

an instance of [Syntax](#)

Implemented in [VICI::stub::SyntaxStubFactory](#).

The documentation for this class was generated from the following file:

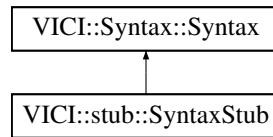
- [vici.h](#)

8.118 VICI::stub::SyntaxStub Class Reference

A stub version of the [Syntax](#) module interface.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::SyntaxStub:



Public Member Functions

- [SyntaxStub](#) ([Window *w](#), [EBNF::EBNF *e](#))
Costructor.
- virtual void [show](#) ([csr s](#))
display the syntax chart for the supplied ebnf

Protected Attributes

- [Window * win](#)
The window to display on.
- [EBNF::EBNF * ebnf](#)
The [EBNF](#) module.

8.118.1 Detailed Description

A stub version of the [Syntax](#) module interface.

8.118.2 Constructor & Destructor Documentation

8.118.2.1 SyntaxStub::SyntaxStub ([Window * w](#), [EBNF::EBNF * e](#))

Costructor.

Parameters

w	The window to display on.
e	The EBNF module.

8.118.3 Member Function Documentation

8.118.3.1 void SyntaxStub::show ([csr s](#)) [[virtual](#)]

display the syntax chart for the supplied ebnf

Parameters

s	the EBNF to build the chart for.
-------------------	--

Implements [VICI::Syntax::Syntax](#).

The documentation for this class was generated from the following files:

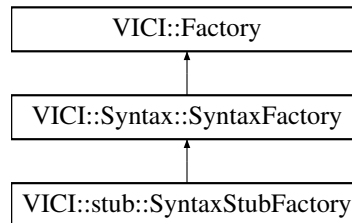
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.119 VICI::stub::SyntaxStubFactory Class Reference

A factory for creating [SyntaxStub](#) objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::SyntaxStubFactory:



Public Member Functions

- virtual [Syntax::Syntax *](#) `makeSyntax (Window *, EBNF::EBNF *)`
Create an instance of [Syntax](#).

8.119.1 Detailed Description

A factory for creating [SyntaxStub](#) objects.

8.119.2 Member Function Documentation

8.119.2.1 `Syntax::Syntax * SyntaxStubFactory::makeSyntax (Window * w, EBNF::EBNF * ebnf)` [virtual]

Create an instance of [Syntax](#).

Parameters

<i>w</i>	the window to display the chart in.
<i>ebnf</i>	pointer to the EBNF object for parsing the option parameters.

Returns

an instance of [Syntax](#)

Implements [VICI::Syntax::SyntaxFactory](#).

The documentation for this class was generated from the following files:

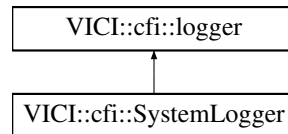
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.120 VICI::cfi::SystemLogger Class Reference

logger that interfaces to the Linux syslog

```
#include <vici/log.h>
```

Inheritance diagram for VICI::cfi::SystemLogger:



Public Member Functions

- [SystemLogger](#) (*csr ident*)
- [log](#) (*Severity, csr line*)
log to the unnamed log
- [log](#) (*csr logName, Severity, csr line*)
log to the named log

Additional Inherited Members

8.120.1 Detailed Description

logger that interfaces to the Linux syslog

8.120.2 Constructor & Destructor Documentation

8.120.2.1 SystemLogger::SystemLogger (*csr ident*)

Parameters

<i>ident</i>	the identifier for the log.
--------------	-----------------------------

8.120.3 Member Function Documentation

8.120.3.1 int SystemLogger::log (*Severity sev, csr line*) [virtual]

log to the unnamed log

Parameters

<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

8.120.3.2 int SystemLogger::log (*csr logName, Severity sev, csr line*) [virtual]

log to the named log

Parameters

<i>logName</i>	the name of the log stream to write to
<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

The documentation for this class was generated from the following files:

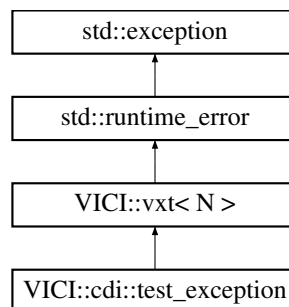
- [log.h](#)
- [log.cpp](#)

8.121 VICI::cdi::test_exception Class Reference

throw this to abandon a particular test case

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::test_exception:



Public Member Functions

- [test_exception \(\)](#)
Constructor.

Additional Inherited Members

8.121.1 Detailed Description

throw this to abandon a particular test case

The documentation for this class was generated from the following file:

- [testmgr.h](#)

8.122 VICI::gth::TestAction Struct Reference

A single action that can be applied to a GUI.

```
#include <vici/libgth.h>
```


Public Attributes

- `std::string label`
Destination label for jumps.
- `int delay`
Time to wait in milliseconds.
- `std::string widgetName`
The name of the widget.
- `std::string command`
The command to run.
- `ParamList params`
The parameters for the command.
- `std::string assignmentRegex`
Regular expression for specifying values for variables.
- `ParamList assignmentNames`
The variables to accept the values.
- `std::vector< std::pair
< std::string, std::string > > jumps`
Pairs of regular expressions and jump labels.

8.122.1 Detailed Description

A single action that can be applied to a GUI.

The documentation for this struct was generated from the following file:

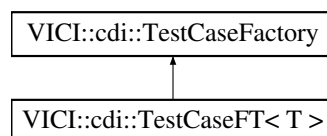
- [libgth.h](#)

8.123 VICI::cdi::TestCaseFactory Class Reference

Define a base type for test case factories.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::TestCaseFactory:



Public Member Functions

- `virtual ~TestCaseFactory ()`
Destructor.
- `virtual AbstractTestCase * make (csr nm)=0`
Create an instance of the test case object.

8.123.1 Detailed Description

Define a base type for test case factories.

8.123.2 Member Function Documentation

8.123.2.1 `virtual AbstractTestCase* VICI::cdi::TestCaseFactory::make (csr nm)` [pure virtual]

Create an instance of the test case object.

Parameters

<i>nm</i>	The name of the test case
-----------	---------------------------

Returns

A pointer to a new instance of the test case.

Implemented in [VICI::cdi::TestCaseFT< T >](#).

The documentation for this class was generated from the following file:

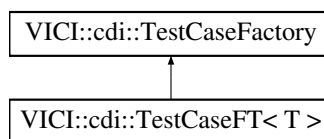
- [testmgr.h](#)

8.124 VICI::cdi::TestCaseFT< T > Class Template Reference

Responsible for creating a test case of some type.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::TestCaseFT< T >:



Public Member Functions

- `AbstractTestCase * make (csr nm)`
Create a test case object.

8.124.1 Detailed Description

```
template<class T>class VICI::cdi::TestCaseFT< T >
```

Responsible for creating a test case of some type.

8.124.2 Member Function Documentation

8.124.2.1 `template<class T > AbstractTestCase* VICI::cdi::TestCaseFT< T >::make (csr nm)` [inline], [virtual]

Create a test case object.

Parameters

<i>nm</i>	The name of the test case
-----------	---------------------------

Returns

A pointer to a new instance of the test case.

Implements [VICI::cdi::TestCaseFactory](#).

The documentation for this class was generated from the following file:

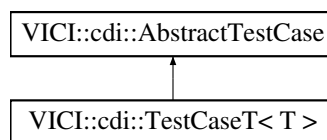
- [testmgr.h](#)

8.125 VICI::cdi::TestCaseT< T > Class Template Reference

Responsible for installing a factory for the test case.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::TestCaseT< T >:



Public Member Functions

- [TestCaseT](#) ([csr nm](#))
Constructor.

Static Public Member Functions

- static void [install](#) ([csr nm](#), [csr scn](#))
Install a factory for the test case.

Additional Inherited Members

8.125.1 Detailed Description

```
template<class T>class VICI::cdi::TestCaseT< T >
```

Responsible for installing a factory for the test case.

8.125.2 Constructor & Destructor Documentation

8.125.2.1 `template<class T> VICI::cdi::TestCaseT< T >::TestCaseT (csr nm) [inline]`

Constructor.

Parameters

<i>nm</i>	The name of the test case.
-----------	----------------------------

8.125.3 Member Function Documentation

8.125.3.1 `template<class T> static void VICI::cdi::TestCaseT< T >::install (csr nm, csr scn) [inline], [static]`

Install a factory for the test case.

Parameters

<i>nm</i>	The name of the test case
<i>scn</i>	The name of enclosing scenario.

The documentation for this class was generated from the following file:

- [testmgr.h](#)

8.126 VICI::cdi::Tester Class Reference

Responsible for managing the testing.

```
#include <vici/testmgr.h>
```

Public Member Functions

- void [configure](#) (csr testName)
specify the name of the test as in the configuration file
- csr [getTestName](#) ()
Get the name of the test.
- virtual [~Tester](#) ()
Destructor.
- cfi::logstream & [log](#) ()
Get a reference to the logging stream used for the testing.
- void [addTestCase](#) (csr name, csr scenario, [TestCaseFactory](#) *tcf)
Add a test case factory.
- void [runTestCase](#) (csr scenarioName, csr testCaseName, [ScenarioResults](#) *)
Run a specific test case.
- void [getTestCases](#) (csr scenarioName, std::set< std::string > &testCases)
Get the test case names for a scenario.
- void [addScenario](#) (csr scenario, [ScenarioFactory](#) *sf)
Add a scenario factory.
- void [setTest](#) ([TestFactory](#) *t)
Set the main test object factory.
- [AbstractTest](#) * [getTest](#) ()
Get the test object.
- [AbstractScenario](#) * [getScenario](#) ()
Get the current scenario object.
- virtual void [runTests](#) ()
Run all the tests.
- bool [summary](#) ()
Print a summary of the testing to the log stream.

Static Public Member Functions

- static [Tester](#) & [instance](#) ()
Get a reference to the [Tester](#).

Protected Member Functions

- [Tester](#) ()
Protected Constructor.
- void [title](#) ()
Print title for the test.
- void [installDefaults](#) ()
Install default test and scenario objects.
- void [testScenario](#) ([csr](#) scenario, [ScenarioFactory](#) *scn)
Do testing of a scenario.

Protected Attributes

- std::string [nameOfTest](#)
Identifier for test in the configuration file.
- std::string [logName](#)
Identifier for the log.
- bool [failed](#)
Flag to hold overall testing result.
- std::map< std::string,
std::map< std::string,
[TestCaseFactory](#) * > > [tests](#)
Factories for test cases indexed by name and scenario name.
- std::map< std::string,
[ScenarioFactory](#) * > [scenarios](#)
Factories for scenarios indexed by scenario name.
- std::map< std::string,
[ScenarioResults](#) * > [results](#)
Results for scenario tests indexed by scenario name.
- [TestFactory](#) * [theTest](#)
Factor for the overall test.
- [AbstractTest](#) * [absTest](#)
Pointer to the abstract test object. Only valid during the test run.
- [AbstractScenario](#) * [absScenario](#)
Pointer to the abstract scenario object. Only valid during the test run.

8.126.1 Detailed Description

Responsible for managing the testing.

A singleton that manages the testing, accumulates the test results and provides a report.

8.126.2 Member Function Documentation

8.126.2.1 void [Tester::addScenario](#) ([csr](#) scenario, [ScenarioFactory](#) * sf)

Add a scenario factory.

Parameters

<i>scenario</i>	The name of the scenario
<i>sf</i>	The factory for the scenario

8.126.2.2 void Tester::addTestCase (*csr name*, *csr scenario*, TestCaseFactory * *tcf*)

Add a test case factory.

Parameters

<i>name</i>	The name of the test case.
<i>scenario</i>	The name of the enclosing scenario.
<i>tcf</i>	The factory for the test case.

8.126.2.3 void Tester::configure (*csr testName*)

specify the name of the test as in the configuration file

Parameters

<i>testName</i>	The name of the test in the config file.
-----------------	--

8.126.2.4 AbstractScenario* VICI::cdi::Tester::getScenario () [inline]

Get the current scenario object.

Returns

The current scenario object

8.126.2.5 AbstractTest* VICI::cdi::Tester::getTest () [inline]

Get the test object.

Returns

The test object

8.126.2.6 csr VICI::cdi::Tester::getTestName () [inline]

Get the name of the test.

Returns

the name of test

8.126.2.7 Tester & Tester::instance () [static]

Get a reference to the [Tester](#).

Returns

A reference to the [Tester](#) singleton object.

8.126.2.8 `logstream & Tester::log ()`

Get a reference to the logging stream used for the testing.

Returns

A reference to the logging stream

8.126.2.9 `void VICI::cdi::Tester::setTest (TestFactory * t) [inline]`

Set the main test object factory.

Parameters

<code>t</code>	The factory for the test object.
----------------	----------------------------------

8.126.2.10 `bool Tester::summary ()`

Print a summary of the testing to the log stream.

Returns

true if the testing found no unexpected errors

8.126.2.11 `void Tester::testScenario (csr scenario, ScenarioFactory * scn) [protected]`

Do testing of a scenario.

Parameters

<code>scenario</code>	The name of the scenario.
<code>scn</code>	The factory for creating the scenario object

The documentation for this class was generated from the following files:

- [testmgr.h](#)
- [testmgr.cpp](#)

8.127 VICI::cdi::TestEvent Class Reference

Represent an event in the object under test.

```
#include <vici/testmgr.h>
```

Public Member Functions

- [TestEvent](#) (const std::string &s)
Constructor.
- const std::string & [id](#) ()
Get the identity of the event.

8.127.1 Detailed Description

Represent an event in the object under test.

These objects are placed into a queue when an event of interest occurs in the object under test. The [AsyncTestCase](#) object will get the object from the queue and perform checks on the object under test.

8.127.2 Constructor & Destructor Documentation

8.127.2.1 `VICI::cdi::TestEvent::TestEvent (const std::string & s) [inline]`

Constructor.

Parameters

<code>s</code>	The identity of the event.
----------------	----------------------------

8.127.3 Member Function Documentation

8.127.3.1 `const std::string& VICI::cdi::TestEvent::id () [inline]`

Get the identity of the event.

Returns

The identifier for the event.

The documentation for this class was generated from the following file:

- [testmgr.h](#)

8.128 VICI::cdi::TestEventQueue Class Reference

Responsible for queuing TestEvents.

```
#include <vici/testmgr.h>
```

Public Member Functions

- [~TestEventQueue \(\)](#)
Destructor.
- void [enqueueEvent \(TestEvent *\)](#)
Place the event on the queue.

Static Public Member Functions

- static [TestEventQueue & instance \(\)](#)
Get reference to the singleton queue object.
- static void [event \(csr id\)](#)
Create a [TestEvent](#), queue it, and wait for it to be processed.

Public Attributes

- `std::mutex` [eventMx](#)
Control access to the queue.
- `std::list< TestEvent * >` [events](#)
The queue.
- `std::set< std::string >` [runningEvents](#)
The set of events being handled.
- `std::mutex` [functionsMx](#)
Control access to list of running events.
- `std::condition_variable` [eventCV](#)
Wait for event to be queued or processed.

Protected Member Functions

- [TestEventQueue](#) ()
Constructor.

8.128.1 Detailed Description

Responsible for queuing TestEvents.

The queue provides a condition variable that allows the test case to wait until an event has been queued.

8.128.2 Member Function Documentation

8.128.2.1 void TestEventQueue::enqueueEvent ([TestEvent](#) * *ev*)

Place the event on the queue.

Parameters

<i>ev</i>	The TestEvent object to place on the queue
-----------	--

8.128.2.2 void TestEventQueue::event (*csr id*) [static]

Create a [TestEvent](#), queue it, and wait for it to be processed.

Parameters

<i>id</i>	The identity for the event
-----------	----------------------------

8.128.2.3 [TestEventQueue](#) & [TestEventQueue::instance](#) () [static]

Get reference to the singleton queue object.

Returns

reference to the [TestEventQueue](#)

The documentation for this class was generated from the following files:

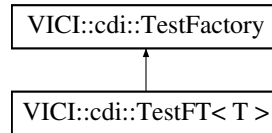
- [testmgr.h](#)
- [testmgr.cpp](#)

8.129 VICI::cdi::TestFactory Class Reference

Responsible for creating an object that manages the resources for the entire test.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::TestFactory:



Public Member Functions

- virtual `~TestFactory ()`
Destructor.
- virtual `AbstractTest * make ()=0`
Create an instance of the test object.

8.129.1 Detailed Description

Responsible for creating an object that manages the resources for the entire test.

8.129.2 Member Function Documentation

8.129.2.1 virtual `AbstractTest* VICI::cdi::TestFactory::make ()` [pure virtual]

Create an instance of the test object.

Returns

Pointer to the test object.

Implemented in `VICI::cdi::TestFT< T >`.

The documentation for this class was generated from the following file:

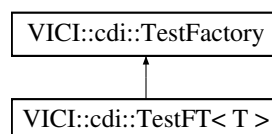
- `testmgr.h`

8.130 VICI::cdi::TestFT< T > Class Template Reference

Responsible for creating a test object of the required type.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::TestFT< T >:



Public Member Functions

- [AbstractTest](#) * [make](#) ()
Construct a Test object.

8.130.1 Detailed Description

```
template<class T>class VICI::cdi::TestFT< T >
```

Responsible for creating a test object of the required type.

8.130.2 Member Function Documentation

8.130.2.1 `template<class T > AbstractTest* VICI::cdi::TestFT< T >::make ()` `[inline]`, `[virtual]`

Construct a Test object.

Returns

pointer to a new object derived from [AbstractTest](#)

Implements [VICI::cdi::TestFactory](#).

The documentation for this class was generated from the following file:

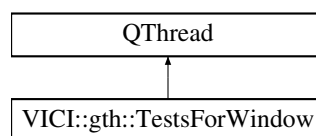
- [testmgr.h](#)

8.131 VICI::gth::TestsForWindow Class Reference

Provides a thread for applying test cases to a window.

```
#include <vici/libgth.h>
```

Inheritance diagram for VICI::gth::TestsForWindow:



Signals

- void [sendAdaptor](#) (const std::string &tname, const std::string &widgetName, const std::string &command, const std::vector< std::string > ¶ms)
A signal containing the adaptor to run in the GUI thread.

Public Member Functions

- [TestsForWindow](#) ([GTHTest](#) *, [csr](#) window, bool isMainWindow)
Constructor.
- [~TestsForWindow](#) ()
Destructor.

- void [addSequence](#) ()
Increment the sequence number.
- void [addTest](#) (csr testCase)
Add a test case to the current sequence.
- void [addAction](#) (csr testCase, [ConDes](#), const [TestAction](#) &)
Add an action to a test case.
- void [addScript](#) (csr chunk)
Add a Lua script.
- [csr name](#) ()
Get the name of the window.
- virtual void [run](#) ()
Begin execution of the thread.
- void [pause](#) ()
Pause the thread.
- void [resume](#) ()
Resume execution of the thread.
- bool [checkPaused](#) ()
Test if paused.
- void [jumpThread](#) (csr tcname, csr widgetName, csr command, const std::vector< std::string > ¶ms)
Run the action in the GUI thread.
- void [jumpNotify](#) ()
Notification that the action has completed.

8.131.1 Detailed Description

Provides a thread for applying test cases to a window.

The documentation for this class was generated from the following files:

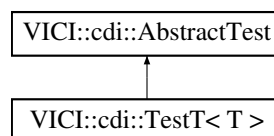
- [libgth.h](#)
- [libgth.cpp](#)

8.132 VICI::cdi::TestT< T > Class Template Reference

Responsible for installing a factory for making test objects.

```
#include <vici/testmgr.h>
```

Inheritance diagram for VICI::cdi::TestT< T >:



Static Public Member Functions

- static void [install](#) ()
Install a factory for a type of [AbstractTest](#) object.

Additional Inherited Members

8.132.1 Detailed Description

```
template<class T>class VICI::cdi::TestT< T >
```

Responsible for installing a factory for making test objects.

The documentation for this class was generated from the following file:

- [testmgr.h](#)

8.133 VICI::Symbol::TextAttributes Class Reference

Hold the attributes of a text comment.

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

Public Member Functions

- [TextAttributes](#) ()
constructor

Public Attributes

- `std::string` [fontName](#)
the name of the font face
- `float` [fontSize](#)
the size of the font

8.133.1 Detailed Description

Hold the attributes of a text comment.

Holds the font size, and name for text objects.

Todo should also include the style - bold, italic, normal

The documentation for this class was generated from the following file:

- [vici.h](#)

8.134 VICI::cdi::Trace Class Reference

This class is used to create trace log entries.

```
#include <vici/trace.h>
```

Public Member Functions

- [Trace](#) (int n)
Constructor.

- [Trace](#) (int *n*, *csr* file)
Constructor.
- `template<class T >`
[Trace](#) & `operator<<` (T *x*)
Stream output operator.
- `~Trace` ()
Destructor.

8.134.1 Detailed Description

This class is used to create trace log entries.

Objects of this class should only be transient. i.e. don't actually name the object, so that when the statement ends the destructor is called immediately.

Typical usage is `TRACE(3) << "my trace message";`

8.134.2 Constructor & Destructor Documentation

8.134.2.1 `Trace::Trace (int n) [explicit]`

Constructor.

Parameters

<i>n</i>	debug level. See Tracer for detailed explanation.
----------	---

8.134.2.2 `Trace::Trace (int n, csr file)`

Constructor.

Parameters

<i>n</i>	debug level. See Tracer for detailed explanation.
<i>file</i>	source file.

The documentation for this class was generated from the following files:

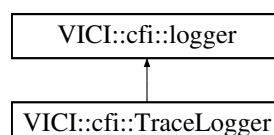
- [trace.h](#)
- [trace.cpp](#)

8.135 VICI::cfi::TraceLogger Class Reference

Class for logging tracing output.

```
#include <vici/log.h>
```

Inheritance diagram for VICI::cfi::TraceLogger:



Public Member Functions

- [TraceLogger](#) (const [Path](#) &fname)
- int [log](#) ([Severity](#), [csr](#) line)
log to the unnamed log
- int [log](#) ([csr](#) logName, [Severity](#), [csr](#) line)
log to the named log

Protected Attributes

- std::ofstream f
The logging stream.

Additional Inherited Members

8.135.1 Detailed Description

Class for logging tracing output.

8.135.2 Constructor & Destructor Documentation

8.135.2.1 TraceLogger::TraceLogger (const [Path](#) & *fname*)

Parameters

<i>fname</i>	the name of the log file.
--------------	---------------------------

8.135.3 Member Function Documentation

8.135.3.1 int TraceLogger::log ([Severity](#) *sev*, [csr](#) *line*) [virtual]

log to the unnamed log

Parameters

<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

8.135.3.2 int TraceLogger::log ([csr](#) *logName*, [Severity](#) *sev*, [csr](#) *line*) [virtual]

log to the named log

Parameters

<i>logName</i>	the name of the log stream to write to
<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

The documentation for this class was generated from the following files:

- [log.h](#)
- [log.cpp](#)

8.136 VICI::cdi::Tracer Class Reference

This class manages the tracing for an application.

```
#include <vici/trace.h>
```

Public Types

- enum [CallEntry](#) { **ENTER** =+1, **EXIT** =-1, **NONE** =0 }
Indicator for entry or exit of a function.

Public Member Functions

- void [log](#) ([csr file](#), int level, [CallEntry](#) call, [csr text](#))
Write a trace line to the log stream.

Static Public Member Functions

- static [Tracer](#) & [instance](#) ()
Get an instance of the singleton object.

8.136.1 Detailed Description

This class manages the tracing for an application.

This class is responsible for managing tracing for an application.

It handles both normal trace messages and function call tracing.

[Trace](#) messages are sent to the log stream defined for tracing in the XINI configuration file. The messages have microsecond resolution timestamps to enable accurate comparison of messages from multiple processes.

Tracing can be controlled on a file by file basis by setting the debug level for the file in the tracing section of the XINI configuration file.

8.136.2 Member Function Documentation

8.136.2.1 void Tracer::log (csr file, int level, CallEntry call, csr text)

Write a trace line to the log stream.

Parameters

<i>file</i>	determines the debugging level for that file from sourceMap
<i>level</i>	the current debug level
<i>call</i>	+1 on entry, -1 on exit, 0 otherwise
<i>text</i>	the log message

The documentation for this class was generated from the following files:

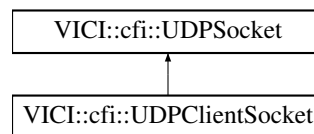
- [trace.h](#)
- [trace.cpp](#)

8.137 VICI::cfi::UDPClientSocket Class Reference

A UDP client socket.

```
#include <sos/udpsocket.h>
```

Inheritance diagram for VICI::cfi::UDPClientSocket:



Public Member Functions

- [UDPClientSocket](#) ([csr](#) host, int portid)
Constructor.

Additional Inherited Members

8.137.1 Detailed Description

A UDP client socket.

A UDP socket which send messages to a specified server.

8.137.2 Constructor & Destructor Documentation

8.137.2.1 UDPClientSocket::UDPClientSocket (*csr* host, int portid)

Constructor.

Parameters

<i>host</i>	the name of the server to send to
<i>portid</i>	the port for the listening server

The documentation for this class was generated from the following files:

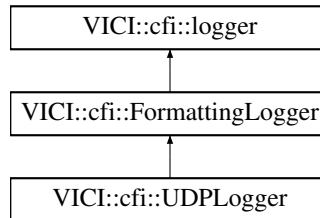
- [udpsocket.h](#)
- [udpsocket.cpp](#)

8.138 VICI::cfi::UDPLoader Class Reference

Class for logging to a logging server.

```
#include <vici/log.h>
```

Inheritance diagram for VICI::cfi::UDPLoader:



Public Member Functions

- [UDPLoader](#) (*csr* hostname, int port)
- int [log](#) (Severity, *csr* line)
log to the unnamed log
- int [log](#) (*csr* logName, Severity, *csr* line)
log to the named log

Protected Attributes

- [UDPClientSocket](#) socket
the udp socket connected to the logging server

Additional Inherited Members

8.138.1 Detailed Description

Class for logging to a logging server.

8.138.2 Constructor & Destructor Documentation

8.138.2.1 UDPLoader::UDPLoader (*csr* hostname, int port)

Parameters

<i>hostname</i>	the identity of the log server in the configuration file
<i>port</i>	the port to send to

8.138.3 Member Function Documentation

8.138.3.1 int UDPLoader::log (Severity sev, *csr* line) [virtual]

log to the unnamed log

Parameters

<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

8.138.3.2 `int UDPLogger::log (csr logName, Severity sev, csr line) [virtual]`

log to the named log

Parameters

<i>logName</i>	the name of the log stream to write to
<i>sev</i>	the severity level for the message
<i>line</i>	the message to place in the log

Returns

the number of characters from the string that were written.

Implements [VICI::cfi::logger](#).

The documentation for this class was generated from the following files:

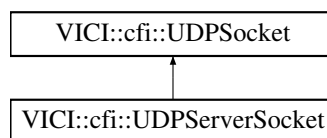
- [log.h](#)
- [log.cpp](#)

8.139 VICI::cfi::UDPServerSocket Class Reference

A UDP Server socket.

```
#include <udpsocket.h>
```

Inheritance diagram for VICI::cfi::UDPServerSocket:



Public Member Functions

- [UDPServerSocket](#) (int portid)
Constructor.
- void [send](#) (csr message, csr host, int port)
send message to specified host and port. Used for replies.

Additional Inherited Members

8.139.1 Detailed Description

A UDP Server socket.

A UDP socket which accepts incoming packets.

8.139.2 Constructor & Destructor Documentation

8.139.2.1 UDPServerSocket::UDPServerSocket (int *portid*)

Constructor.

Parameters

<i>portid</i>	the port to listen on.
---------------	------------------------

8.139.3 Member Function Documentation

8.139.3.1 void UDPServerSocket::send (*csr message*, *csr host*, int *port*)

send message to specified host and port. Used for replies.

Parameters

<i>message</i>	the text to send.
<i>host</i>	the machine to send to.
<i>port</i>	the port to send to.

The documentation for this class was generated from the following files:

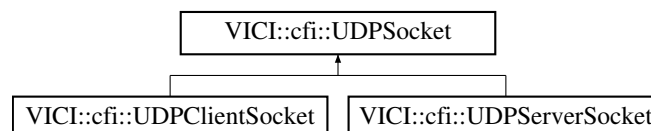
- [udpsocket.h](#)
- [udpsocket.cpp](#)

8.140 VICI::cfi::UDPSocket Class Reference

An abstract base class with common stuff for UDP sockets.

```
#include <sos/udpsocket.h>
```

Inheritance diagram for VICI::cfi::UDPSocket:



Public Member Functions

- virtual [~UDPSocket](#) ()
destructor
- int [fd](#) ()
get the file descriptor for the socket

- void `send` (*csr* message)
 - send a message via the socket's file descriptor*
- void `recv` (std::string &message, std::string &host, int &port)
 - wait for the next incoming message*

Static Public Attributes

- static const int `BUFFER_SIZE` = 16000

Protected Member Functions

- `UDPSocket` ()
 - constructor*

Protected Attributes

- struct sockaddr_in `addr`
 - internal address structure*
- unsigned int `addr_len`
 - size of address structure*
- int `sd`
 - file descriptor*

8.140.1 Detailed Description

An abstract base class with common stuff for UDP sockets.

The common code for client and server udp sockets which manages the file descriptor and the buffer for the messages.

8.140.2 Member Function Documentation

8.140.2.1 void UDPSocket::recv (std::string & message, std::string & host, int & port)

wait for the next incoming message

Parameters

<i>message</i>	a string into which the message is placed
<i>host</i>	the host name of the sending machine
<i>port</i>	the port of the sending machine.

8.140.2.2 void UDPSocket::send (csr message)

send a message via the socket's file descriptor

Parameters

<i>message</i>	the text to send.
----------------	-------------------

8.140.3 Member Data Documentation

8.140.3.1 `const int VICI::cfi::UDPSocket::BUFFER_SIZE = 16000` `[static]`

the max size for udp packets is 64K but we should keep them as small as possible and send several if needs be.

The documentation for this class was generated from the following files:

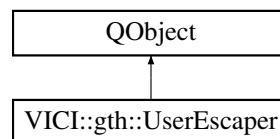
- [udpsocket.h](#)
- [udpsocket.cpp](#)

8.141 VICI::gth::UserEscaper Class Reference

A class for allowing the user to suspend the test.

```
#include <vici/libgth.h>
```

Inheritance diagram for VICI::gth::UserEscaper:



Public Member Functions

- [UserEscaper](#) (bool &flag)
Constructor.

Protected Member Functions

- bool [eventFilter](#) (QObject *obj, QEvent *event)
Called to process events.

8.141.1 Detailed Description

A class for allowing the user to suspend the test.

This class manages a keyboard event for the F5 key so that the user can suspend a GUI test and regain control of the mouse. Useful if you want to make a note about some aspect of the test.

The documentation for this class was generated from the following files:

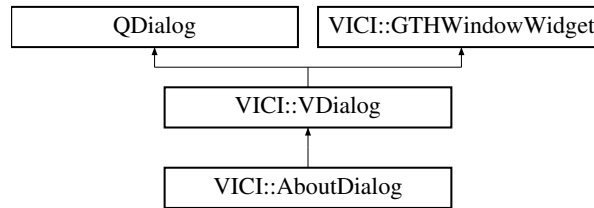
- [libgth.h](#)
- [libgth.cpp](#)

8.142 VICI::VDialog Class Reference

A dialog class that automatically registers itself.

```
#include <vici/libgui.h>
```

Inheritance diagram for VICI::VDialog:



Public Member Functions

- [VDialog](#) (QWidget *parent=0, Qt::WindowFlags f=0)
Constructor.
- [~VDialog](#) ()
Destructor.

Protected Member Functions

- virtual void [showEvent](#) (QShowEvent *event)
Called when the dialog is shown.

8.142.1 Detailed Description

A dialog class that automatically registers itself.

The documentation for this class was generated from the following files:

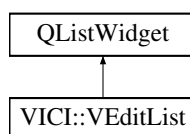
- [libgui.h](#)
- [libgui.cpp](#)

8.143 VICI::VEditList Class Reference

Builds on QListWidget to provide better editing ability.

```
#include <vici/libgui.h>
```

Inheritance diagram for VICI::VEditList:



Public Member Functions

- [VEditList](#) (QWidget *parent=0)
Constructor.
- void [append](#) (const std::vector< std::string > &)
Add list of strings.
- std::vector< std::string > [getVals](#) () const
Get the rows as std::strings.

Protected Slots

- virtual void [addBlank](#) (QListWidgetItem *)
Add a blank row at the end.
- virtual void [changed](#) (QListWidgetItem *it, QListWidgetItem *)
Called if a row has been changed.

8.143.1 Detailed Description

Builds on QListWidget to provide better editing ability.

This class provides a list widget where the entries can be edited or moved using drag-n-drop.

The documentation for this class was generated from the following files:

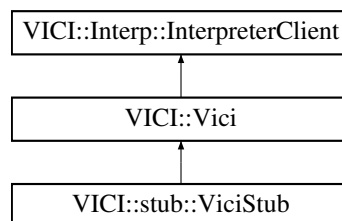
- [libgui.h](#)
- [libgui.cpp](#)

8.144 VICI::Vici Class Reference

An API for the vici runtime gui.

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

Inheritance diagram for VICI::Vici:



Public Member Functions

- virtual [~Vici](#) ()
virtual destructor
- virtual [Interp::Interpreter](#) * [getInterpreter](#) ()=0
Get a reference to the interpreter object.
- virtual void [openFile](#) (const [Path](#) &)=0
Set the file to execute.
- virtual void [autoStart](#) ()=0
Start the script as soon as the GUI is ready.

8.144.1 Detailed Description

An API for the vici runtime gui.

The facade for the runtime program.

The implementation will run a script and provide a user interface for interacting with the script.

The documentation for this class was generated from the following file:

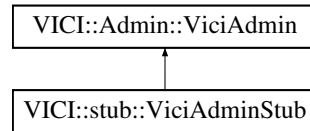
- [vici.h](#)

8.145 VICI::Admin::ViciAdmin Class Reference

Application for preparing commands.

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

Inheritance diagram for VICI::Admin::ViciAdmin:



Public Member Functions

- virtual `~ViciAdmin ()`
virtual destructor
- virtual `VICI::Window * getMainWindow ()=0`
Get a pointer to the app main window.

8.145.1 Detailed Description

Application for preparing commands.

The facade for the vici-adm application.

The [VICI](#) application for preparing commands so that they are easier for novice users to find and use.

The documentation for this class was generated from the following file:

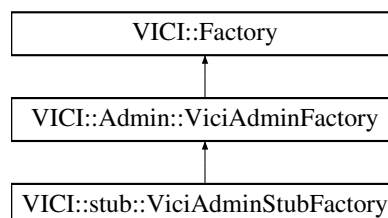
- [vici.h](#)

8.146 VICI::Admin::ViciAdminFactory Class Reference

An abstract factory for making an instance of [ViciAdmin](#).

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

Inheritance diagram for VICI::Admin::ViciAdminFactory:



Public Member Functions

- virtual `~ViciAdminFactory ()`
virtual destructor
- virtual `ViciAdmin * makeViciAdmin ()=0`
Create an instance of the [ViciAdmin](#) class.

8.146.1 Detailed Description

An abstract factory for making an instance of [ViciAdmin](#).

The implementation will create either the production version, or a stub, or a test version of the [ViciAdmin](#) class.

8.146.2 Member Function Documentation

8.146.2.1 virtual ViciAdmin* VICI::Admin::ViciAdminFactory::makeViciAdmin () [pure virtual]

Create an instance of the [ViciAdmin](#) class.

Returns

an instance of the [ViciAdmin](#) class

Implemented in [VICI::stub::ViciAdminStubFactory](#).

The documentation for this class was generated from the following file:

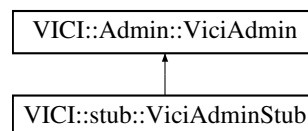
- [vici.h](#)

8.147 VICI::stub::ViciAdminStub Class Reference

A stub version of the ViciAdmin module interface.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::ViciAdminStub:



Public Member Functions

- virtual [VICI::Window](#) * [getMainWindow](#) ()
Get a pointer to the app main window.

8.147.1 Detailed Description

A stub version of the ViciAdmin module interface.

The documentation for this class was generated from the following files:

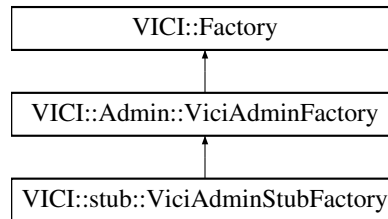
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.148 VICI::stub::ViciAdminStubFactory Class Reference

A factory for creating stub instances of ViciAdmin.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::ViciAdminStubFactory:



Public Member Functions

- virtual [Admin::ViciAdmin](#) * [makeViciAdmin](#) ()
Create an instance of the ViciAdmin class.

8.148.1 Detailed Description

A factory for creating stub instances of ViciAdmin.

8.148.2 Member Function Documentation

8.148.2.1 Admin::ViciAdmin * ViciAdminStubFactory::makeViciAdmin () [virtual]

Create an instance of the ViciAdmin class.

Returns

an instance of the ViciAdmin class

Implements [VICI::Admin::ViciAdminFactory](#).

The documentation for this class was generated from the following files:

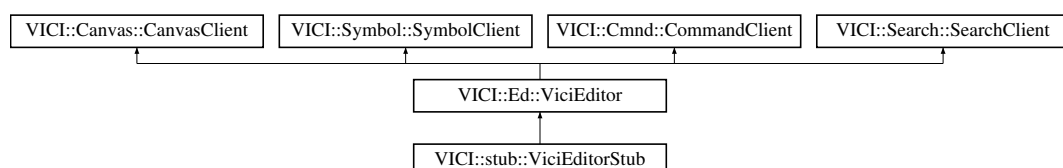
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.149 VICI::Ed::ViciEditor Class Reference

The flowchart editor.

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

Inheritance diagram for VICI::Ed::ViciEditor:



Public Member Functions

- virtual `~ViciEditor ()`
virtual destructor
- virtual `VICI::Window * getMainWindow ()=0`
Get a pointer to the main window.
- virtual void `openFile (const VICI::Path &)=0`
Tell the editor which file to start with.

8.149.1 Detailed Description

The flowchart editor.

The facade for the editor classes.

The implementation will allow the user to enter and edit a flowchart, test it and install it into their desktop.

The documentation for this class was generated from the following file:

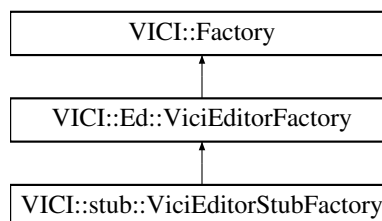
- [vici.h](#)

8.150 VICI::Ed::ViciEditorFactory Class Reference

An abstract factory for making an instance of [ViciEditor](#).

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

Inheritance diagram for VICI::Ed::ViciEditorFactory:



Public Member Functions

- virtual `~ViciEditorFactory ()`
virtual destructor
- virtual `ViciEditor * makeViciEditor ()=0`
create an instance of the ViciEditor class

8.150.1 Detailed Description

An abstract factory for making an instance of [ViciEditor](#).

The implementation will create either the production version, or a stub, or a test version of the [ViciEditor](#) class.

The documentation for this class was generated from the following file:

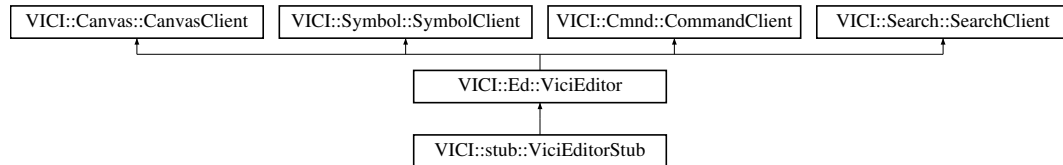
- [vici.h](#)

8.151 VICI::stub::ViciEditorStub Class Reference

A stub version of the [Vici](#) Editor module.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::ViciEditorStub:



Public Member Functions

- virtual [VICI::Window](#) * [getMainWindow](#) ()
Get a pointer to the main window.
- virtual void [openFile](#) (const [VICI::Path](#) &)
Tell the editor which file to start with.
- virtual void [newSymbol](#) ([Symbol::Symbol](#) *)
called when a symbol is placed on the canvas.
- virtual void [newChart](#) ()
called when a new diagram has been started
- virtual void [changedChart](#) ()
called when the chart is first changed
- virtual void [searchAction](#) ([NodeId](#))
notification that the user wants the search tab
- virtual void [commandAction](#) ([NodeId](#), const [ArgList](#) &)
notification that the command tab is to be displayed
- virtual void [breakAction](#) ([NodeId](#), bool)
notification that a breakpoint has been set or cleared
- virtual void [selection](#) ([Symbol::Symbol](#) *)
notify that a symbol has been selected.
- virtual void [symbolAttr](#) ([Symbol::SymbolAttributes](#) &)
notify that the default symbol attributes have changed
- virtual void [textAttr](#) ([Symbol::TextAttributes](#) &)
notify that the default text attributes have changed
- virtual void [optionsAndParameters](#) ([NodeId](#), const [ArgList](#) &)
this gets called when the user presses OK
- virtual void [cmndError](#) (csr)
this gets called if there is an error
- virtual void [selectedCommand](#) ([NodeId](#), csr)
this function is called when the user selects a command.

8.151.1 Detailed Description

A stub version of the [Vici](#) Editor module.

8.151.2 Member Function Documentation

8.151.2.1 `void ViciEditorStub::cmdnError (csr msg)` [virtual]

this gets called if there is an error

Parameters

<i>msg</i>	the text of the error message
------------	-------------------------------

Implements [VICI::Cmnd::CommandClient](#).

8.151.2.2 void ViciEditorStub::newSymbol (Symbol::Symbol * sym) [virtual]

called when a symbol is placed on the canvas.

Parameters

<i>sym</i>	the symbol that was placed on the canvas.
------------	---

Implements [VICI::Canvas::CanvasClient](#).

8.151.2.3 void ViciEditorStub::optionsAndParameters (NodId n, const ArgList & args) [virtual]

this gets called when the user presses OK

Parameters

<i>args</i>	the selected command and its options are returned in this
<i>nid</i>	The identifier for the current node

Implements [VICI::Cmnd::CommandClient](#).

8.151.2.4 void ViciEditorStub::selectedCommand (NodId n, csr c) [virtual]

this function is called when the user selects a command.

Parameters

<i>c</i>	the selected command
<i>n</i>	the identifier for the node

Implements [VICI::Search::SearchClient](#).

8.151.2.5 void ViciEditorStub::selection (Symbol::Symbol * sym) [virtual]

notify that a symbol has been selected.

Parameters

<i>sym</i>	the symbol that was selected.
------------	-------------------------------

Todo do we need to notify of deselection ?

Implements [VICI::Symbol::SymbolClient](#).

8.151.2.6 void ViciEditorStub::symbolAttr (Symbol::SymbolAttributes & att) [virtual]

notify that the default symbol attributes have changed

Parameters

<i>att</i>	the new default attributes
------------	----------------------------

Implements [VICI::Symbol::SymbolClient](#).

8.151.2.7 void `ViciEditorStub::textAttr (Symbol::TextAttributes & att)` [virtual]

notify that the default text attributes have changed

Parameters

<i>att</i>	the new default text attributes.
------------	----------------------------------

Implements [VICI::Symbol::SymbolClient](#).

The documentation for this class was generated from the following files:

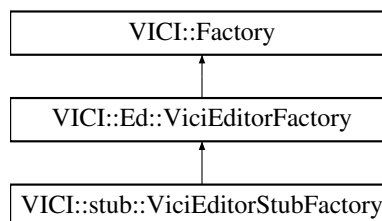
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.152 VICI::stub::ViciEditorStubFactory Class Reference

A factory for creating stub instances of [Vici](#) Editor objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for `VICI::stub::ViciEditorStubFactory`:



Public Member Functions

- virtual `Ed::ViciEditor * makeViciEditor ()`
create an instance of the ViciEditor class

8.152.1 Detailed Description

A factory for creating stub instances of [Vici](#) Editor objects.

The documentation for this class was generated from the following files:

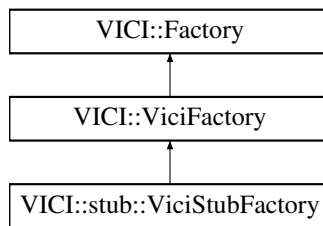
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.153 VICI::ViciFactory Class Reference

An abstract factory for making an instance of [Vici](#).

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


Inheritance diagram for VICI::ViciFactory:



Public Member Functions

- virtual `~ViciFactory ()`
virtual destructor
- virtual `Vici * makeVici (Window *w=nullptr)=0`
create an instance of the Vici class

8.153.1 Detailed Description

An abstract factory for making an instance of [Vici](#).

The implementation will create either the production version, or a stub, or a test version of the [Vici](#) class.

The documentation for this class was generated from the following file:

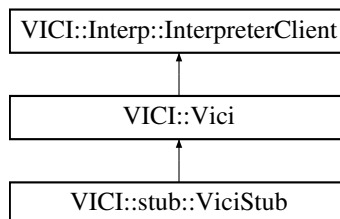
- [vici.h](#)

8.154 VICI::stub::ViciStub Class Reference

A stub version of the [Vici](#) module.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::ViciStub:



Public Member Functions

- virtual `Interp::Interpreter * getInterpreter ()`
Get a reference to the interpreter object.
- virtual void `openFile (const VICI::Path &)`
Set the file to execute.
- virtual void `autoStart ()`
Start the script as soon as the GUI is ready.
- virtual void `setValue (csr varName, csr value)`

- this gets called when a script variable has a change of value*
- virtual void [setFile](#) (int state, [csr](#) filename)
- this gets called when a monitored file changes state.*
- virtual void [setCursor](#) ([ThreadId](#), [NodeId](#))
- this gets called as the shell steps through the script*
- virtual void [breakReached](#) ([ThreadId](#), [NodeId](#))
- this gets called when a break point is reached*
- virtual void [dataReady](#) ([NodeId](#))
- this gets called when data is available on a display*
- virtual void [reportError](#) ([Severity](#), [csr](#))
- This gets called if an error is detected.*
- virtual void [done](#) ()
- this gets called when the script completes*

8.154.1 Detailed Description

A stub version of the [Vici](#) module.

8.154.2 Member Function Documentation

8.154.2.1 void [ViciStub::breakReached](#) ([ThreadId](#) *tid*, [NodeId](#) *node*) [virtual]

this gets called when a break point is reached

Parameters

<i>tid</i>	the thread which reached the breakpoint
<i>node</i>	the location of the breakpoint

Implements [VICI::Interp::InterpreterClient](#).

8.154.2.2 void [ViciStub::dataReady](#) ([NodeId](#) *node*) [virtual]

this gets called when data is available on a display

Parameters

<i>node</i>	the node of the display object in the flowchart
-------------	---

Implements [VICI::Interp::InterpreterClient](#).

8.154.2.3 void [ViciStub::reportError](#) ([Severity](#) *sev*, [csr](#) *msg*) [virtual]

This gets called if an error is detected.

Parameters

<i>msg</i>	The text of the error message.
<i>sev</i>	The severity of the error

Implements [VICI::Interp::InterpreterClient](#).

8.154.2.4 void [ViciStub::setCursor](#) ([ThreadId](#) *tid*, [NodeId](#) *node*) [virtual]

this gets called as the shell steps through the script

Parameters

<i>tid</i>	the thread which changed to executing this node
<i>node</i>	the node that is currently being executed

Implements [VICI::Interp::InterpreterClient](#).

8.154.2.5 void ViciStub::setFile (int *state*, *csr filename*) [virtual]

this gets called when a monitored file changes state.

Parameters

<i>state</i>	this needs to be defined
<i>filename</i>	the name of the file that changed

Implements [VICI::Interp::InterpreterClient](#).

8.154.2.6 void ViciStub::setValue (*csr varName*, *csr value*) [virtual]

this gets called when a script variable has a change of value

Parameters

<i>varName</i>	the name of the variable that changed value
<i>value</i>	the new value of the variable

Implements [VICI::Interp::InterpreterClient](#).

The documentation for this class was generated from the following files:

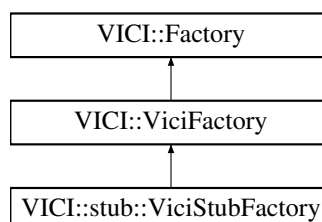
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.155 VICI::stub::ViciStubFactory Class Reference

A factory for creating stub instances of [Vici](#) objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::ViciStubFactory:



Public Member Functions

- virtual [Vici](#) * [makeVici](#) ([Window](#) *w=nullptr)
create an instance of the [Vici](#) class

8.155.1 Detailed Description

A factory for creating stub instances of [Vici](#) objects.

The documentation for this class was generated from the following files:

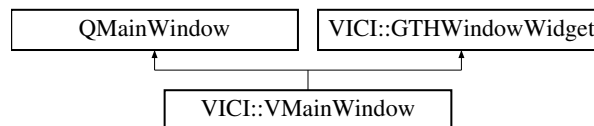
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.156 VICI::VMainWindow Class Reference

A main window class that automatically registers itself.

```
#include <vici/libgui.h>
```

Inheritance diagram for VICI::VMainWindow:



Signals

- void [widgetsRealised](#) ()
Emitted when the widgets have been made visible.

Public Member Functions

- [VMainWindow](#) (QWidget *parent=0, Qt::WindowFlags flags=0)
Constructor.

Protected Slots

- virtual void [startUp](#) ()
This is called to do things that require the widgets to have been realized.

Protected Member Functions

- virtual void [showEvent](#) (QShowEvent *event)
Called when the window gets shown.

8.156.1 Detailed Description

A main window class that automatically registers itself.

The documentation for this class was generated from the following files:

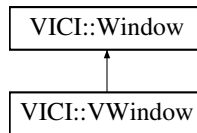
- [libgui.h](#)
- [libgui.cpp](#)

8.157 VICI::VWindow Class Reference

An implementation of [Window](#) for holding Qt's QWidget objects.

```
#include <vici/window.h>
```

Inheritance diagram for VICI::VWindow:



Public Member Functions

- [VWindow](#) (QWidget *w)
Constructor.
- [operator QWidget * \(\) const](#)
type conversion operator
- [QWidget * operator-> \(\)](#)
pointer operator

Protected Attributes

- [QWidget * widget](#)
the window being passed

8.157.1 Detailed Description

An implementation of [Window](#) for holding Qt's QWidget objects.

This class implements the [Window](#) type for QWidgets It allows references to windows to be passed between modules without the Qt libraries leaking into the general interface definitions defined in [vici.h](#).

8.157.2 Constructor & Destructor Documentation

8.157.2.1 VICI::VWindow::VWindow (QWidget * w) [inline]

Constructor.

Parameters

<i>w</i>	the window to encapsulate
----------	---------------------------

The documentation for this class was generated from the following file:

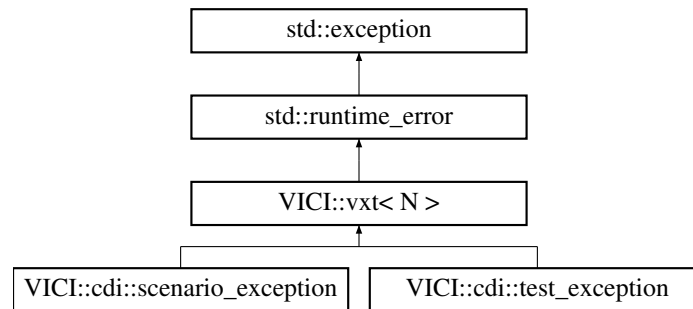
- [window.h](#)

8.158 VICI::vxt< N > Class Template Reference

An exception object with severity levels.

```
#include <vici/vx.h>
```

Inheritance diagram for `VICI::vxt< N >`:



Public Member Functions

- `vxt` ([Severity](#) s)
constructor
- `vxt` ([Severity](#) s, const `std::string` &f, int ln)
constructor
- `template<class T >`
`vxt` & `operator<<` (T x)
Provide stream syntax for the exception object.
- [Severity](#) severity ()
get the severity of the exception
- virtual const char * `what` () const throw ()
get the message from the exception

Static Public Member Functions

- static const `std::string` & `sevToString` ([VICI::Severity](#) s)
Convert a severity into a string.

Protected Attributes

- [Severity](#) mSeverity
severity level of the exception
- `std::string` buff
message built into this
- `std::string` file
source file from which it was thrown
- int line
line number in the source file
- int ptr
insertion point for message text

8.158.1 Detailed Description

```
template<typename N>class VICI::vxt< N >
```

An exception object with severity levels.

The `vxt` class is an exception object that includes severity levels and has stream syntax for creating messages.

Typical usage: `throw vxt(VICI::Error) << "something went wrong: " << strerror(errno);`

A macro version that automatically records the file name and line number is also defined. It would be used like:

`throw VX(Warning) << "there is a problem";`

8.158.2 Constructor & Destructor Documentation

8.158.2.1 `template<typename N> VICI::vxt< N >::vxt (Severity s) [inline],[explicit]`

constructor

Parameters

<i>s</i>	The severity of the problem.
----------	------------------------------

8.158.2.2 `template<typename N> VICI::vxt< N >::vxt (Severity s, const std::string & f, int ln) [inline]`

constructor

Parameters

<i>s</i>	the severity of the exception
<i>f</i>	the file name of the source file
<i>ln</i>	the line number of the source file

The documentation for this class was generated from the following file:

- [vx.h](#)

8.159 VICI::WidgetMgr Class Reference

A singleton class that window widgets register with.

```
#include <vici/libgui.h>
```

Public Member Functions

- void [registerWindowWidget](#) ([GTHWindowWidget](#) *)
Add window to list and inform any clients.
- void [deregisterWindowWidget](#) ([GTHWindowWidget](#) *)
Remove a window and inform any clients.
- void [addClient](#) ([WidgetMgrClient](#) *)
Add client to list and inform it about any registered windows.
- void [removeClient](#) ([WidgetMgrClient](#) *)
Remove a client.

Static Public Member Functions

- static [WidgetMgr](#) & [instance](#) ()
Get a reference to the window widget manager.

8.159.1 Detailed Description

A singleton class that window widgets register with.

The test harness can register as a client to be told about the widgets so that it can ask them to register their widgets for use by the test harness.

The documentation for this class was generated from the following files:

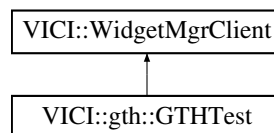
- [libgui.h](#)
- [libgui.cpp](#)

8.160 VICI::WidgetMgrClient Class Reference

An interface for the [WidgetMgr](#) to call its clients when a window registers itself.

```
#include <vici/libgui.h>
```

Inheritance diagram for VICI::WidgetMgrClient:



Public Member Functions

- virtual void [windowHasRegistered](#) ([GTHWindowWidget](#) *)=0
Called to notify of a new window registering itself.
- virtual void [windowHasDeregistered](#) ([GTHWindowWidget](#) *)=0
Called to notify that a window has deregistered itself.

8.160.1 Detailed Description

An interface for the [WidgetMgr](#) to call its clients when a window registers itself.

The documentation for this class was generated from the following file:

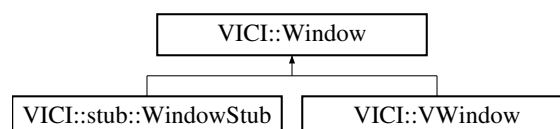
- [libgui.h](#)

8.161 VICI::Window Class Reference

A wrapper class for windows.

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

Inheritance diagram for VICI::Window:



Public Member Functions

- virtual `~Window ()`
virtual destructor

8.161.1 Detailed Description

A wrapper class for windows.

A wrapper class for windows so that the top level design is not dependent on the implementation details of Qt etc.

Pointers to these will be dynamically cast to a derived type that will have a member that is a pointer to the implementation window object.

The documentation for this class was generated from the following file:

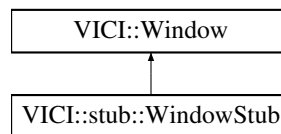
- [vici.h](#)

8.162 VICI::stub::WindowStub Class Reference

A stub version of window objects.

```
#include <vici/libifstubs.h>
```

Inheritance diagram for VICI::stub::WindowStub:



Additional Inherited Members

8.162.1 Detailed Description

A stub version of window objects.

The documentation for this class was generated from the following files:

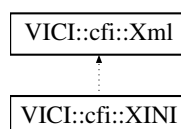
- [libifstubs.h](#)
- [libifstubs.cpp](#)

8.163 VICI::cfi::XINI Class Reference

Load an XML configuration file.

```
#include <vici/xini.h>
```

Inheritance diagram for VICI::cfi::XINI:



Public Member Functions

- bool `getVal` (const std::string &xpath_expression, std::string &val)
get a value from the config file.
- int `getVals` (const std::string &xpath_expr, std::vector< std::string > &results)
get a set of values
- void `getPath` (const std::string &xpath_expr, `VICI::Path` &path)
get a path from a set of paths
- const `Path` & `getConfigFilename` () const
get the name of the config file being used

Static Public Member Functions

- static void `configure` (const `Path` &p)
Configure XINI to use the specified path.
- static void `configure` (int c, char **v)
Configure XINI to use the command line args.
- static `XINI` & `instance` ()
get a reference to the single instance of the class.

Protected Member Functions

- `XINI` ()
constructor

Static Protected Attributes

- static int `argc` = 0
number of command line args
- static char ** `argv` = NULL
command line args
- static `Path` `configFile`
path to config file
- static const std::string `config`
configuration string defined by the using application using the format specified.

Friends

- class `::XiniTestCase`

8.163.1 Detailed Description

Load an XML configuration file.

Ini file, XML format, loaded by looking in the following locations:

- P = path provided on the command line.
- E = path provided in the environment
- C = the current working directory

- H = the user's home directory
- D = directories as listed

Containing application must provide a string containing the keys for the above search locations:

```
const string ini::config = "PECHD:-i:QM_CONF:qmd.xml:.qmd.xml:/etc/qmd/qmd.xml";
```

The above will first look for a command line parameter following -i, then it will look in the environment for a variable QM_CONF, then in the current directory for a file called qmd.xml, then for a file \$HOME/qmd.xml, and finally for /etc/qmd/qmd.xml

Any further paths added at the end are checked in sequence.

8.163.2 Member Function Documentation

8.163.2.1 void XINI::configure (const Path & p) [static]

Configure [XINI](#) to use the specified path.

Parameters

<i>p</i>	The path to the XML config file to use
----------	--

8.163.2.2 void XINI::configure (int c, char ** v) [static]

Configure [XINI](#) to use the command line args.

Parameters

<i>c</i>	The number of command line args
<i>v</i>	the command line args

8.163.2.3 const Path & XINI::getConfigFilename () const

get the name of the config file being used

Returns

path of the configuration file that was found.

8.163.2.4 void XINI::getPath (const std::string & xpath_expr, VICI::Path & path)

get a path from a set of paths

Parameters

<i>xpath_expr</i>	An expression returning a set of paths to search
<i>path</i>	The first path that points to an existing file

8.163.2.5 bool XINI::getVal (const std::string & xpath_expression, std::string & val)

get a value from the config file.

Parameters

<i>xpath_ - expression</i>	an expression to find in the config file.
<i>val</i>	value is returned into this

Returns

true if the value exists, and is unique

8.163.2.6 `int XINI::getVals (const std::string & xpath_expr, std::vector< std::string > & results)`

get a set of values

Parameters

<i>xpath_expr</i>	an expression to find in the config file.
<i>results</i>	set of values returned in this

Returns

the size of the set

8.163.3 Member Data Documentation

8.163.3.1 `const string VICI::cfi::XINI::config` `[static]`, `[protected]`

Initial value:

```
=
    "PECHD:"
    "-i:VICI_CONF:"
    "vici.xml:"
    ".local/share/vici/vici.xml:"
    "$VICI/share/vici/vici.xml:"
    "/usr/local/share/vici/vici.xml:"
    "/usr/share/vici/vici.xml:"
```

configuration string defined by the using application using the format specified.

The documentation for this class was generated from the following files:

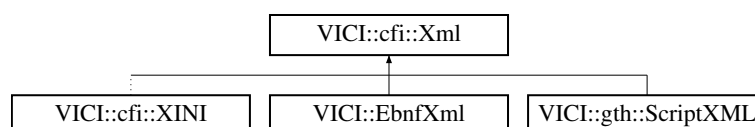
- [xini.h](#)
- [xini.cpp](#)
- [factory.cpp](#)

8.164 VICI::cfi::Xml Class Reference

A C++ wrapper for libxml2.

```
#include <vici/xml.h>
```

Inheritance diagram for VICI::cfi::Xml:



Public Member Functions

- virtual `~Xml ()`
Destructor.
- void `setDtd (csr name, const Path &dtd)`
Set DTD identifier to dtd, which is just the dtd file name.
- void `setDtd (csr name, csr ident, csr url)`
Set DTD identifier.
- void `getDtdIdentifiers (csr dtdName, std::string &publicId, std::string &systemId)`
Get the public identifier for the DTD.
- void `setCompression (int rate)`
Set the compression rate for saving the document.
- void `save ()`
Save the document.
- void `save (const Path &fname)`
Save the document to a new file.
- void `pack (XmlBuffer &)`
Save the document into a buffer.
- bool `dirty ()`
Check if the document has been modified.
- void `setDirty (bool dt)`
Access to dirty flag if we change document outside of this interface.
- bool `safeToSave ()`
Check if can save without clobbering other input.
- void `reload ()`
Reload the XML file.

Protected Member Functions

- void `createXPathContext ()`
Call this whenever mDoc is explicitly updated.
- void `registerNamespace (const std::string &prefix, csr uri)`
- `Xml ()`
Constructor.
- void `open (const Path &fname)`
Open an existing XML file.
- void `create (const Path &fname, csr root_element)`
Create a new xml file and document.
- void `unpack (XmlBuffer &)`
Create a document from the content of a buffer.
- void `freeDoc ()`
release the document from memory
- void `setProp (xmlNodePtr node, csr prop, csr val)`
Set a string property.
- void `setProp (xmlNodePtr node, csr prop, int val)`
Set an integer property.
- void `setProp (xmlNodePtr node, csr prop, double val)`
Set a double property.
- void `setContent (xmlNodePtr node, csr text)`
Set the content for the node.
- void `setCDATAContent (xmlNodePtr node, csr text)`

- Set the content for the node using CDATA.*

 - void [getPropString](#) (xmlNodePtr node, [csr](#) prop, std::string &val)
 - Get the property value as a string.*
 - void [getPropInt](#) (xmlNodePtr node, [csr](#) prop, int &val)
 - Get the property value as an integer.*
 - void [getPropShort](#) (xmlNodePtr node, [csr](#) prop, short &val)
 - Get the property value as a short integer.*
 - void [getPropDouble](#) (xmlNodePtr node, [csr](#) prop, double &val)
 - Get the property value as a double.*
 - std::string [getNodeContent](#) (xmlNodePtr p, int expand=1) const
 - Get the content of the node.*
 - std::string [getNodeName](#) (xmlNodePtr p) const
 - Get the name of the node.*
 - xmlNodePtr [getRoot](#) ()
 - Get the root node.*
 - xmlNodePtr [getChild](#) (xmlNodePtr node, [csr](#) name) const
 - Get a child node by name.*
 - int [getChildren](#) (xmlNodePtr node, std::vector< xmlNodePtr > &nodes) const
 - Get the children of the node.*
 - xmlNodePtr [newNode](#) (xmlNodePtr node, [csr](#) name)
 - Create a new node.*
 - void [deleteNode](#) (xmlNodePtr node)
 - Delete a node and all of its children.*
 - int [find](#) ([csr](#) xpath_expression, std::vector< xmlNodePtr > &nodes)
 - Find nodes matching an XPath.*

Static Protected Member Functions

- static xmlNodePtr [newTextChild](#) (xmlNodePtr node, [csr](#) name, [csr](#) text)
 - Create a new node with text content.*

Protected Attributes

- [Path](#) [mFilename](#)
 - the XML file that the object represents*
- struct stat [xmlStat](#)
 - time the file was last accessed by us*
- xmlDocPtr [mDoc](#)
 - pointer to the root document object*
- xmlXPathContextPtr [mCtx](#)
 - XML XPath context.*
- bool [mDirty](#)
 - true when an update has been made, and saving is required.*
- bool [mIsOpen](#)
 - true if the file is open*
- int [mCompression](#)
 - 0 is uncompressed, thru 9 for max zlib compression*

Static Protected Attributes

- static const int `UMASK_RW_RW_R` = 0664 ^ 0777
Default umask for xml files created.
- static bool `xpathinit` = false
flag to indicate if xpath has been initialized

8.164.1 Detailed Description

A C++ wrapper for libxml2.

The `Xml` class provides an object oriented wrapper for the libxml2 library. Applications will subclass this class to provide application specific functions for accessing the XML data.

8.164.2 Member Function Documentation

8.164.2.1 `void Xml::create (const Path & fname, csr root_element)` [protected]

Create a new xml file and document.

Parameters

<i>fname</i>	the name of the file which will be created on save.
<i>root_element</i>	the name of the root element

8.164.2.2 `void Xml::deleteNode (xmlNodePtr node)` [protected]

Delete a node and all of its children.

Parameters

<i>node</i>	the document node
-------------	-------------------

8.164.2.3 `bool VICI::cfi::Xml::dirty ()` [inline]

Check if the document has been modified.

Returns

true if the document has been modified.

8.164.2.4 `int Xml::find (csr xpath_expression, std::vector< xmlNodePtr > & nodes)` [protected]

Find nodes matching an XPath.

Parameters

<i>xpath_ - expression</i>	the expression to search for.
<i>nodes</i>	a vector for the matching nodes.

8.164.2.5 `xmlNodePtr Xml::getChild (xmlNodePtr node, csr name) const` [protected]

Get a child node by name.

Parameters

<i>node</i>	the document node
<i>name</i>	the name of the node.

8.164.2.6 `int Xml::getChildren (xmlNodePtr node, std::vector< xmlNodePtr > & nodes) const` [protected]

Get the children of the node.

Parameters

<i>node</i>	the document node
<i>nodes</i>	a vector for the children nodes.

8.164.2.7 `void Xml::getDtdIdentifiers (csr dtdName, std::string & publicId, std::string & systemId)`

Get the public identifier for the DTD.

Parameters

<i>dtdName</i>	the name for the DTD entry
<i>publicId</i>	the returned public identifier for the DTD
<i>systemId</i>	the returned system identifier for the DTD

8.164.2.8 `string Xml::getNodeContent (xmlNodePtr p, int expand = 1) const` [protected]

Get the content of the node.

Parameters

<i>p</i>	the document node
<i>expand</i>	if 1 then get the content of child nodes.

8.164.2.9 `string Xml::getNodeName (xmlNodePtr p) const` [protected]

Get the name of the node.

Parameters

<i>p</i>	the document node
----------	-------------------

8.164.2.10 `void Xml::getPropDouble (xmlNodePtr node, csr prop, double & val)` [protected]

Get the property value as a double.

Parameters

<i>node</i>	the document node
<i>prop</i>	the property name for the element
<i>val</i>	the value for the property

8.164.2.11 `void Xml::getPropInt (xmlNodePtr node, csr prop, int & val)` [protected]

Get the property value as an integer.

Parameters

<i>node</i>	the document node
<i>prop</i>	the property name for the element
<i>val</i>	the value for the property

8.164.2.12 void Xml::getPropShort (xmlNodePtr *node*, csr *prop*, short & *val*) [protected]

Get the property value as a short integer.

Parameters

<i>node</i>	the document node
<i>prop</i>	the property name for the element
<i>val</i>	the value for the property

8.164.2.13 void Xml::getPropString (xmlNodePtr *node*, csr *prop*, std::string & *val*) [protected]

Get the property value as a string.

Parameters

<i>node</i>	the document node
<i>prop</i>	the property name for the element
<i>val</i>	the value for the property

8.164.2.14 xmlNodePtr Xml::newNode (xmlNodePtr *node*, csr *name*) [protected]

Create a new node.

Parameters

<i>node</i>	the document node
<i>name</i>	the name for new node.

8.164.2.15 xmlNodePtr Xml::newTextChild (xmlNodePtr *node*, csr *name*, csr *text*) [static],[protected]

Create a new node with text content.

Parameters

<i>node</i>	the document node
<i>name</i>	the name for the node.
<i>text</i>	the content for the node.

8.164.2.16 void Xml::open (const Path & *fname*) [protected]

Open an existing XML file.

Parameters

<i>fname</i>	the name of the file to open.
--------------	-------------------------------

8.164.2.17 void Xml::registerNamespace (const std::string & *prefix*, *csr uri*) [protected]

Call this to register the prefixes for all uri's used in XPath queries. The prefix does not have to be the same as used in the document and you must supply one for the "anonymous" uri's.

8.164.2.18 bool Xml::safeToSave ()

Check if can save without clobbering other input.

Returns

true if its safe to save.

8.164.2.19 void Xml::save (const Path & *fname*)

Save the document to a new file.

Parameters

<i>fname</i>	the new file name.
--------------	--------------------

8.164.2.20 void Xml::setCDATAContent (xmlNodePtr *node*, *csr text*) [protected]

Set the content for the node using CDATA.

Parameters

<i>node</i>	the document node
<i>text</i>	the value to assign to the node.

8.164.2.21 void Xml::setCompression (int *rate*)

Set the compression rate for saving the document.

Parameters

<i>rate</i>	the required compression factor.
-------------	----------------------------------

8.164.2.22 void Xml::setContent (xmlNodePtr *node*, *csr text*) [protected]

Set the content for the node.

Parameters

<i>node</i>	the document node
<i>text</i>	the value to assign to the node content.

8.164.2.23 void VICI::cfi::Xml::setDirty (bool *dt*) [inline]

Access to dirty flag if we change document outside of this interface.

Parameters

<i>dt</i>	new value for the dirty flag
-----------	------------------------------

8.164.2.24 void Xml::setDtd (*csr name*, const Path & *dtd*)

Set DTD identifier to *dtd*, which is just the *dtd* file name.

Parameters

<i>name</i>	the name for the DTD entry.
<i>dtd</i>	The DTD file name

8.164.2.25 void Xml::setDtd (*csr name*, *csr ident*, *csr url*)

Set DTD identifier.

Parameters

<i>name</i>	The name for the DTD entry
<i>ident</i>	The external id for the entry
<i>url</i>	The URL of the DTD.

8.164.2.26 void Xml::setProp (XmlNodePtr *node*, *csr prop*, *csr val*) [protected]

Set a string property.

Parameters

<i>node</i>	the document node
<i>prop</i>	the property name for the element
<i>val</i>	the value for the property

8.164.2.27 void Xml::setProp (XmlNodePtr *node*, *csr prop*, int *val*) [protected]

Set an integer property.

Parameters

<i>node</i>	the document node
<i>prop</i>	the property name for the element
<i>val</i>	the value for the property

8.164.2.28 void Xml::setProp (XmlNodePtr *node*, *csr prop*, double *val*) [protected]

Set a double property.

Parameters

<i>node</i>	the document node
<i>prop</i>	the property name for the element

<i>val</i>	the value for the property
------------	----------------------------

The documentation for this class was generated from the following files:

- [xml.h](#)
- [xml.cpp](#)

8.165 VICI::cfi::XmlBuffer Class Reference

An abstract class defining a buffer object.

```
#include <vici/xml.h>
```

Public Member Functions

- virtual [~XmlBuffer](#) ()
Ensure derived classes have a destructor.
- virtual char * [addr](#) ()=0
Get the address of the data.
- virtual int [len](#) ()=0
Get the length of the data.
- virtual void [addr](#) (char *)=0
Set the address of the data.
- virtual void [len](#) (int)=0
Set the length of the data.

8.165.1 Detailed Description

An abstract class defining a buffer object.

The [XmlBuffer](#) objects hold a complete XML file which can be packed or unpacked by the [Xml](#) objects.

The documentation for this class was generated from the following file:

- [xml.h](#)

Chapter 9

File Documentation

9.1 canvas.h File Reference

The API for handling graphics scenes between modules.

```
#include "vici.h"
```

Classes

- class [VICI::CanvasScene](#)
An implementation of the [Scene](#) abstract class for holding a [QGraphicsScene](#).

Namespaces

- [VICI](#)
The namespace for the [Visual Chart Interpreter](#) project.

9.1.1 Detailed Description

The API for handling graphics scenes between modules.

9.2 discover.h File Reference

Provides the API for objects that need to be discoverable for testing.

```
#include "stringy.h"  
#include <map>  
#include <vector>  
#include <memory>
```

Classes

- struct [VICI::cfi::DiscoverPointer](#)
Holds a pointer to a discoverable object.
- class [VICI::cfi::Discoverable](#)

A mixin class that makes its owner discoverable.

- class [VICI::cfi::DiscoveryMgr](#)

Manager for discoverable objects.

Namespaces

- [VICI](#)

The namespace for the Visual Chart Interpreter project.

- [VICI::cfi](#)

The namespace for the Common Facilities Infrastructure components.

Macros

- `#define` [DISCOVERABLE](#)

A macro which notifies the discovery manager of an object that is to be discoverable.

Typedefs

- typedef `std::shared_ptr`
< `DiscoverPointer` > [VICI::cfi::DiscoverSharedPointer](#)
A shared pointer that will be owned by a discoverable object.
- typedef `std::weak_ptr`
< `DiscoverPointer` > [VICI::cfi::DiscoverWeakPointer](#)
A weak pointer that will be held by the [DiscoveryMgr](#).

9.2.1 Detailed Description

Provides the API for objects that need to be discoverable for testing. This header should be included in the code for modules that need to be tested.

9.2.2 Macro Definition Documentation

9.2.2.1 #define DISCOVERABLE

Value:

```
VICI::cfi::DiscoveryMgr::instance().save( __PRETTY_FUNCTION__, \
    (discover.reset( new VICI::cfi::DiscoverPointer( this ) ),
    discover) );
```

A macro which notifies the discovery manager of an object that is to be discoverable.

9.3 fdstream.h File Reference

Provide C++ stream interface to file descriptor integers.

```
#include <iostream>
#include <string>
```

Classes

- class [VICI::cfi::FD](#)
Wrap file descriptors in a class to ensure closed.
- class [VICI::cfi::fdoutbuf](#)
An output stream buffer.
- class [VICI::cfi::fdostream](#)
A file descriptor output stream.
- class [VICI::cfi::fdinbuf](#)
An input stream buffer.
- class [VICI::cfi::fdistream](#)
A file descriptor input stream.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::cfi](#)
The namespace for the Common Facilities Infrastructure components.

9.3.1 Detailed Description

Provide C++ stream interface to file descriptor integers.

9.4 gth.h File Reference

Provide the interface between the GUI applications and the GUI Test Harness.

```
#include <vici/stringy.h>
#include <functional>
```

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::gth](#)
The namespace for the GUI Test Harness.

Enumerations

- enum [VICI::gth::WidgetType](#) {
[VICI::gth::Action](#), [VICI::gth::Button](#), [VICI::gth::CheckBox](#), [VICI::gth::ComboBox](#),
[VICI::gth::Dock](#), [VICI::gth::List](#), [VICI::gth::ListView](#), [VICI::gth::Label](#),
[VICI::gth::LineEdit](#), [VICI::gth::SpinBox](#), [VICI::gth::Splitter](#), [VICI::gth::Table](#),
[VICI::gth::Tabs](#), [VICI::gth::TextEdit](#), [VICI::gth::Tree](#), [VICI::gth::View](#),
[VICI::gth::Window](#), [VICI::gth::MessageBox](#), [VICI::gth::FileDialog](#), [VICI::gth::FontDialog](#),
[VICI::gth::ColorDialog](#), [VICI::gth::Script](#) }
List the types of widgets that we can interact with during testing.

Functions

- void [VICI::gth::RegnFn](#) (std::function< void(void *, WidgetType, csr)> reg)

Register the widgets for use by the GUI Test Harness.

9.4.1 Detailed Description

Provide the interface between the GUI applications and the GUI Test Harness. The GUI applications should include this header file if they are to support the GUI Test Harness plug-in testing.

9.5 ipc.h File Reference

Declarations for the semaphore component of libcfi.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include "stringy.h"
```

Classes

- class [VICI::cfi::Semaphore](#)

A semaphore for managing exclusive access to resources across multiple processes.

- class [VICI::cfi::SemaphoreLock](#)

Mutual exclusion lock.

Namespaces

- [VICI](#)

The namespace for the Visual Chart Interpreter project.

- [VICI::cfi](#)

The namespace for the Common Facilities Infrastructure components.

9.5.1 Detailed Description

Declarations for the semaphore component of libcfi.

9.6 libgth.h File Reference

Provides the API for the GUI Test Harness library.


```
#include <vici/testmgr.h>
#include <vici/proc.h>
#include <vici/fdstream.h>
#include <vici/xml.h>
#include "libgui.h"
#include "gth.h"
#include <map>
#include <memory>
#include <mutex>
#include <condition_variable>
#include <QThread>
#include <QGraphicsView>
```

Classes

- class [VICI::gth::Adaptor](#)
Defines an abstract base class for widget adaptors.
- class [VICI::gth::LuaScript](#)
Provides a wrapper for a lua_State object.
- struct [VICI::gth::TestAction](#)
A single action that can be applied to a GUI.
- class [VICI::gth::TestsForWindow](#)
Provides a thread for applying test cases to a window.
- class [VICI::gth::UserEscaper](#)
A class for allowing the user to suspend the test.
- class [VICI::gth::GTHTest](#)
An AbstractTest derived class for testing GUI programs.
- class [VICI::gth::GTHTestCase](#)
Interface for test cases that do gui testing.
- class [VICI::gth::DefaultTestCase](#)
Provides a test case to use for actions that don't have an explicit test case.
- class [VICI::gth::ScriptXML](#)
Provides an interface to the script XML file.
- class [VICI::gth::MouseEventReporter](#)
Provides a means of getting scene coordinates in a QGraphicsView.
- class [VICI::gth::AdaptorST< wt >](#)
Adds a static method to [Adaptor](#) to describe it;.
- class [VICI::gth::AdaptorT< wt, T >](#)
The interpreter for the script commands.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::gth](#)
The namespace for the GUI Test Harness.

Typedefs

- typedef `std::vector< std::string >` [VICI::gth::ParamList](#)
ParamList is used for the parameters of a command.
- typedef `std::vector< std::list< std::string > >` [VICI::gth::ListOfLists](#)
ListOfLists is used to describe the commands and parameters of an [Adaptor](#).
- typedef `std::shared_ptr< ScriptXML >` [VICI::gth::ScriptXmlPtr](#)
Provide for automatic destruction when owner is deleted.
- typedef `std::shared_ptr< LuaScript >` [VICI::gth::LuaScriptPtr](#)
Provides for automatic destruction when the owner is deleted.
- typedef `std::shared_ptr< TestsForWindow >` [VICI::gth::TestsForWindowPtr](#)
Provides for automatic destruction when the owner is deleted.

Enumerations

- enum [VICI::gth::ConDes](#) { **Constructor**, **Destructor** }
Enumerates the stages at which the actions for a test case can be run.

9.6.1 Detailed Description

Provides the API for the GUI Test Harness library. This header should be included in the code for the application specific plug-in library that will use the GUI Test Harness library.

9.7 libgui.h File Reference

Provides project wide GUI components.

```
#include <QWidget>
#include <QDialog>
#include <QMainWindow>
#include <QItemDelegate>
#include <QTextBrowser>
#include <QListWidget>
#include <list>
#include <functional>
#include <memory>
#include <thread>
#include "gth.h"
#include <vici/test.h>
```

Classes

- class [VICI::GTHWindowWidget](#)
A mix-in class with the functions required for the gui test harness.
- class [VICI::WidgetMgrClient](#)
An interface for the [WidgetMgr](#) to call its clients when a window registers itself.
- class [VICI::WidgetMgr](#)

- A singleton class that window widgets register with.*

 - class [VICI::VMainWindow](#)

A main window class that automatically registers itself.
- class [VICI::VDialog](#)

A dialog class that automatically registers itself.
- class [VICI::ItemDelegate](#)

An item delegate that uses QLineEdit for editing list and table entries.
- class [VICI::VEditList](#)

Builds on QListWidget to provide better editing ability.
- class [VICI::Metrics](#)

A class to encapsulate measurements.
- class [VICI::SignalToQtSignal](#)

A class to convert operating system signals into Qt signals.
- class [VICI::AboutDialog](#)

A standard About [Vici](#) dialog for the project.

Namespaces

- [VICI](#)
- The namespace for the Visual Chart Interpreter project.*

9.7.1 Detailed Description

Provides project wide GUI components. This header should be included in the code the uses main windows and/or dialogs.

9.8 libifstubs.h File Reference

Provide a stub implementation for all [VICI](#) modules.

```
#include <vector>
#include <map>
#include "vici.h"
```

Classes

- class [VICI::stub::Event](#)
- Define an Abstract base type for [Dispatcher](#) events.*
- class [VICI::stub::Dispatcher](#)
- Send events to registered objects.*
- class [VICI::stub::AnEvent< T >](#)
- A template wrapper for function calls.*
- class [VICI::stub::WindowStub](#)
- A stub version of window objects.*
- class [VICI::stub::EBNF_Stub](#)
- A stub version of the [EBNF](#) module interface.*
- class [VICI::stub::EBNF_Stub_Factory](#)
- A factory for creating instances of the [EBNF_Stub](#).*
- class [VICI::stub::SyntaxStub](#)

- A stub version of the [Syntax](#) module interface.*

 - class [VICI::stub::SyntaxStubFactory](#)

A factory for creating [SyntaxStub](#) objects.
 - class [VICI::stub::ViciAdminStub](#)

A stub version of the [ViciAdmin](#) module interface.
 - class [VICI::stub::ViciAdminStubFactory](#)

A factory for creating stub instances of [ViciAdmin](#).
 - class [VICI::stub::SearchStub](#)

A stub version of the [Search](#) module.
 - class [VICI::stub::SearchStubFactory](#)

A factory for creating stub instances of [Search](#) objects.
 - class [VICI::stub::CommandStub](#)

A stub version of the [Command](#) module.
 - class [VICI::stub::CommandStubFactory](#)

A factory for creating stub instances of [Command](#) objects.
 - class [VICI::stub::SymbolStub](#)

A stub version of the [Symbol](#) class.
 - class [VICI::stub::SymbolMgrStub](#)

A stub version of the [Symbol Manager](#).
 - class [VICI::stub::SymbolStubFactory](#)

A factory for creating stub instances of [Symbol Manager](#) objects.
 - class [VICI::stub::CanvasStub](#)

A stub version of the [Canvas](#) module.
 - class [VICI::stub::CanvasStubFactory](#)

A factory for creating stub instances of [Canvas](#) objects.
 - class [VICI::stub::SecureStub](#)

A stub version of the [Secure](#) module.
 - class [VICI::stub::SecureStubFactory](#)

A factory for creating stub instances of [Security](#) objects.
 - class [VICI::stub::CronStub](#)

A stub version of the [Cron](#) module.
 - class [VICI::stub::CronStubFactory](#)

A factory for creating stub instances of [Cron](#) objects.
 - class [VICI::stub::InstallerStub](#)

A stub version of the [Installer](#) module.
 - class [VICI::stub::InstallerStubFactory](#)

A factory for creating stub instances of [Installer](#) objects.
 - class [VICI::stub::InterpreterStub](#)

A stub version of the [Interpreter](#) module.
 - class [VICI::stub::InterpreterStubFactory](#)

A factory for creating stub instances of [Interpreter](#) objects.
 - class [VICI::stub::ViciEditorStub](#)

A stub version of the [Vici Editor](#) module.
 - class [VICI::stub::ViciEditorStubFactory](#)

A factory for creating stub instances of [Vici Editor](#) objects.
 - class [VICI::stub::ViciStub](#)

A stub version of the [Vici](#) module.
 - class [VICI::stub::ViciStubFactory](#)

A factory for creating stub instances of [Vici](#) objects.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::stub](#)
The namespace for stub versions of modules.

9.8.1 Detailed Description

Provide a stub implementation for all [VICI](#) modules.

9.9 log.h File Reference

Declarations for the logging component of libcfi.

```
#include <map>
#include <fstream>
#include "vx.h"
#include "udpsocket.h"
#include "ipc.h"
```

Classes

- class [VICI::cfi::logstream](#)
A stream class specialized for logging.
- class [VICI::cfi::logger](#)
An abstract base class used by the log stream to write logs.
- class [VICI::cfi::FormattingLogger](#)
Class for producing a formatted log message.
- class [VICI::cfi::StdLogger](#)
Class for logging to the std output streams.
- class [VICI::cfi::FileLogger](#)
Class for logging to a file.
- class [VICI::cfi::PlainFileLogger](#)
- class [VICI::cfi::UDPLogger](#)
Class for logging to a logging server.
- class [VICI::cfi::TraceLogger](#)
Class for logging tracing output.
- class [VICI::cfi::SystemLogger](#)
logger that interfaces to the Linux syslog

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::cfi](#)
The namespace for the Common Facilities Infrastructure components.

9.9.1 Detailed Description

Declarations for the logging component of libcfi.

9.10 parseTree.h File Reference

The API for implementation of the EBNF Parse Tree.

```
#include "vici.h"  
#include <vici/xml.h>  
#include <map>
```

Classes

- class [VICI::EbnfNode](#)
A node of the [EBNF](#) parse tree.
- class [VICI::EbnfTree](#)
An implementation of the [ParseTree](#) type.
- class [VICI::EbnfXml](#)
A specialization of the [Xml](#) class for the parse tree.

Namespaces

- [VICI](#)
The namespace for the [Visual Chart Interpreter](#) project.

9.10.1 Detailed Description

The API for implementation of the EBNF Parse Tree.

9.11 plugin.h File Reference

Declarations for the plug-in component of libcfi.

```
#include "stringy.h"  
#include <map>  
#include <list>  
#include <memory>  
#include <iostream>
```

Classes

- class [VICI::cfi::PlugIn](#)
Base class for objects loaded from dynamically loaded libraries.
- class [VICI::cfi::AutoRunPlugIn](#)
Base class for plug ins that are run immediately that the library is loaded.
- class [VICI::cfi::PlugInFactory](#)
Base class for factories that create plug-in objects.

- class [VICI::cfi::PlugInFamilyFactoryT< F >](#)
Template base class for families of plug-in factories.
- class [VICI::cfi::PlugInFactoryT< F, P >](#)
Template class for factories.
- struct [VICI::cfi::PlugInDescriptor](#)
Descriptor for plug-ins that can be used by the application.
- struct [VICI::cfi::PlugInDetails](#)
Details of plug-ins as provided by the loaded library.
- class [VICI::cfi::PlugInLib](#)
Manages an instance of a dynamically loaded shared library.
- class [VICI::cfi::PlugInMgr](#)
Manage the handling of plug-in shared libraries.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::cfi](#)
The namespace for the Common Facilities Infrastructure components.

Macros

- `#define VICI_PLUGIN_VERSION "0.1"`
Version number for plug-ins. This is checked when the library is loaded.

Typedefs

- typedef `std::shared_ptr`
< `PlugInLib` > [VICI::cfi::PlugInLibPtr](#)
Shared pointer to [PlugInLib](#).
- typedef `std::unique_ptr`
< `AutoRunPlugIn` > [VICI::cfi::AutoRunPlugInPtr](#)
Unique pointer to an [AutoRunPlugIn](#).

Functions

- void [initViciPlugin](#) ()
Initialise the plugin.
- void [closeViciPlugin](#) ()
Shutdown the plugin.

9.11.1 Detailed Description

Declarations for the plug-in component of libcfi. Applications will include this file if the need to load plug-ins such as used by the testing component.

Plug-in libraries will include this file.

9.11.2 Function Documentation

9.11.2.1 void closeViciPlugin ()

Shutdown the plugin.

Each plug-in will implement this function. It should delete the factory objects.

9.11.2.2 void initViciPlugin ()

Initialise the plugin.

Each plug-in will implement this function. It should add a PluginDetails object to the PluginMgr

9.12 proc.h File Reference

Declarations for classes that manage child processes.

```
#include <string>
#include <map>
#include <vector>
#include <signal.h>
#include <mutex>
#include <thread>
```

Classes

- class [VICI::cfi::AbstractChildProcess](#)
Abstract child process interface for [ChildProcessMgr](#).
- class [VICI::cfi::ChildProcess](#)
Represents the state of a child process.
- class [VICI::cfi::ProcessOwner](#)
Interface that allows the owner of a child process to be notified.
- class [VICI::cfi::ChildProcessMgr](#)
Manage the child processes.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::cfi](#)
The namespace for the Common Facilities Infrastructure components.

9.12.1 Detailed Description

Declarations for classes that manage child processes.

9.13 stringy.h File Reference

Useful string functions.

```
#include <string>
#include <vector>
```

Classes

- class [VICI::Path](#)
Manipulate path strings.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.

Typedefs

- typedef const std::string & [csr](#)
Save some time typing and shorten parameter lines.

Functions

- void [VICI::trim](#) (std::string &s)
rip off leading and trailing white spaces
- int [VICI::split](#) (csr text, std::vector< std::string > &result)
split a string into sub-strings at spaces
- std::string [VICI::expandMacros](#) (csr s)
expand a string containing \$ macros
- std::string [VICI::clean](#) (csr s)
Remove everything except alpha, digit, space, dot, hyphen and underscore.

9.13.1 Detailed Description

Useful string functions.

9.14 test.h File Reference

Interface between applications and the test harness.

```
#include <string>
```

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.

Typedefs

- typedef void(* [VICI::AsyncTestEventFn](#))(const std::string &s)
Pointer to function used to enqueue a test event.

Functions

- void [VICI::defaultAsyncTestEvent](#) (const std::string &s)
Function used to enqueue a test event.

Variables

- AsyncTestEventFn [VICI::asyncTestEvent](#) = [VICI::defaultAsyncTestEvent](#)
Pointer to function used to enqueue a test event.

9.14.1 Detailed Description

Interface between applications and the test harness. This file should be included if the code is instrumented with event() calls that are used for asynchronous tests.

9.15 testmgr.h File Reference

Declarations for the test harness.

```
#include "stringy.h"
#include "vx.h"
#include "log.h"
#include "test.h"
#include <map>
#include <list>
#include <set>
#include <fstream>
#include <mutex>
#include <condition_variable>
#include <chrono>
```

Classes

- class [VICI::cdi::Tester](#)
Responsible for managing the testing.
- class [VICI::cdi::test_exception](#)
throw this to abandon a particular test case
- class [VICI::cdi::scenario_exception](#)
throw this to abandon an entire scenario
- class [VICI::cdi::TestEvent](#)
Represent an event in the object under test.
- class [VICI::cdi::TestEventQueue](#)
Responsible for queuing TestEvents.
- class [VICI::cdi::TestFactory](#)
Responsible for creating an object that manages the resources for the entire test.

- class [VICI::cdi::TestFT< T >](#)
Responsible for creating a test object of the required type.
- class [VICI::cdi::AbstractTest](#)
Responsible for managing resources needed for the entire test.
- class [VICI::cdi::TestT< T >](#)
Responsible for installing a factory for making test objects.
- class [VICI::cdi::DefaultTest](#)
Provide a default version of the [AbstractTest](#) object.
- class [VICI::cdi::ScenarioFactory](#)
Define a type for scenario factories.
- class [VICI::cdi::ScenarioFT< T >](#)
Responsible for creating a scenario of some type.
- struct [VICI::cdi::ScenarioResults](#)
Responsible for storing the results of testing for a scenario.
- class [VICI::cdi::AbstractScenario](#)
Define a type for scenarios.
- class [VICI::cdi::ScenarioT< T >](#)
Responsible for installing a factory to create scenarios of the required type.
- class [VICI::cdi::DefaultScenario](#)
Provide a default scenario object.
- class [VICI::cdi::TestCaseFactory](#)
Define a base type for test case factories.
- class [VICI::cdi::TestCaseFT< T >](#)
Responsible for creating a test case of some type.
- class [VICI::cdi::AbstractTestCase](#)
Base class for test cases.
- class [VICI::cdi::TestCaseT< T >](#)
Responsible for installing a factory for the test case.
- class [VICI::cdi::AsyncTestCase](#)
Responsible for handling asynchronous tests.
- class [VICI::cdi::AsyncTestCaseT< T >](#)
Responsible for installing the factory for the test case type.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::cdi](#)
The namespace for the Common Development Infrastructure.

9.15.1 Detailed Description

Declarations for the test harness.

9.16 trace.h File Reference

Support for tracing the execution path.

```
#include "stringy.h"
#include <map>
#include <sstream>
#include <mutex>
```

Classes

- class [VICI::cdi::CallTrace](#)
Class to create a call trace.
- class [VICI::cdi::NullTrace](#)
A class for dummy trace objects.
- class [VICI::cdi::Trace](#)
This class is used to create trace log entries.
- class [VICI::cdi::Tracer](#)
This class manages the tracing for an application.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::cdi](#)
The namespace for the Common Development Infrastructure.

Macros

- `#define TRACE(n) VICI::cdi::NullTrace()`
Macro for adding trace statements to code.
- `#define FN_TRACE(n, x)`
Macro for adding function call tracing to code.

9.16.1 Detailed Description

Support for tracing the execution path.

9.17 udpsocket.h File Reference

Declarations for a wrapper for the UDP socket.

```
#include "stringy.h"
#include <sys/socket.h>
#include <netinet/in.h>
```

Classes

- class [VICI::cfi::UDPSocket](#)
An abstract base class with common stuff for UDP sockets.
- class [VICI::cfi::UDPClientSocket](#)
A UDP client socket.
- class [VICI::cfi::UDPServerSocket](#)
A UDP Server socket.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::cfi](#)
The namespace for the Common Facilities Infrastructure components.

9.17.1 Detailed Description

Declarations for a wrapper for the UDP socket.

9.18 vici.h File Reference

Project wide declarations and definitions.

```
#include <string>
#include <vector>
#include <map>
#include <memory>
#include <vici/vx.h>
#include <vici/stringy.h>
```

Classes

- class [VICI::Factory](#)
An abstract type for factories.
- class [VICI::FactoryFactory](#)
Responsible for creating and supplying factories for the main modules.
- class [VICI::Window](#)
A wrapper class for windows.
- class [VICI::Scene](#)
A wrapper for QGraphicsScene class.
- class [VICI::EBNF::ParseTree](#)
A type for the parsed form of EBNF.
- class [VICI::EBNF::EBNF](#)
A parser for EBNF.
- class [VICI::EBNF::EBNF_Factory](#)
An abstract factory for making an instance of EBNF.
- class [VICI::Syntax::Syntax](#)
Display a syntax chart of command options.
- class [VICI::Syntax::SyntaxFactory](#)
An abstract factory for making an instance of Syntax.
- class [VICI::Admin::ViciAdmin](#)
Application for preparing commands.
- class [VICI::Admin::ViciAdminFactory](#)
An abstract factory for making an instance of ViciAdmin.
- class [VICI::Search::SearchClient](#)
The interface that must implemented for clients of Search.
- class [VICI::Search::Search](#)
Allow the user to search for, and organise commands.

- class [VICI::Search::SearchFactory](#)
An abstract factory for making an instance of [Search](#).
- class [VICI::Interp::InterpreterClient](#)
The interface that must be implemented by clients of the interpreter.
- class [VICI::Interp::Interpreter](#)
The API for interpreter library.
- class [VICI::Interp::InterpreterFactory](#)
An abstract factory for making an instance of [Interpreter](#).
- class [VICI::Symbol::SymbolAttributes](#)
Holds the attributes of a symbol.
- class [VICI::Symbol::TextAttributes](#)
Hold the attributes of a text comment.
- class [VICI::Symbol::SymbolOwner](#)
Represents something that is notified about events occurring on a symbol.
- class [VICI::Symbol::Symbol](#)
Represents something that is drawn on the canvas.
- class [VICI::Symbol::SymbolClient](#)
An interface that is notified of changes to symbol manager.
- class [VICI::Symbol::SymbolMgr](#)
The facade for the symbol library.
- class [VICI::Symbol::SymbolFactory](#)
An abstract factory for making an instance of [SymbolMgr](#).
- class [VICI::Canvas::CanvasClient](#)
An interface that is notified of canvas actions.
- class [VICI::Canvas::Canvas](#)
The facade for canvas library.
- class [VICI::Canvas::CanvasFactory](#)
An abstract factory for making an instance of [Canvas](#).
- class [VICI::Cmnd::CommandClient](#)
An interface that is notified of a command selection.
- class [VICI::Cmnd::Command](#)
The facade for the command library.
- class [VICI::Cmnd::CommandFactory](#)
An abstract factory for making an instance of [Command](#).
- class [VICI::Sec::Secure](#)
Provide security for the scripts.
- class [VICI::Sec::SecureFactory](#)
An abstract factory for making an instance of [Secure](#).
- class [VICI::Cron::Cron](#)
Allow scripts to be scheduled for later running.
- class [VICI::Cron::CronFactory](#)
An abstract factory for making an instance of [Cron](#).
- class [VICI::Inst::Installer](#)
Install a script into the user's desktop.
- class [VICI::Inst::InstallerFactory](#)
An abstract factory for making an instance of [Installer](#).
- class [VICI::Ed::ViciEditor](#)
The flowchart editor.
- class [VICI::Ed::ViciEditorFactory](#)
An abstract factory for making an instance of [ViciEditor](#).
- class [VICI::Vici](#)
An API for the vici runtime gui.
- class [VICI::ViciFactory](#)
An abstract factory for making an instance of [Vici](#).

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::EBNF](#)
An API for the libebnf library.
- [VICI::Syntax](#)
An API for libsyntax.
- [VICI::Admin](#)
An API for the vici-adm program.
- [VICI::Search](#)
An API for libsearch.
- [VICI::Sec](#)
An API for libsecure.
- [VICI::Interp](#)
An API for the vici script interpreter.
- [VICI::Symbol](#)
An API for libsymbols.
- [VICI::Canvas](#)
An API for the [Canvas](#) which handles the drawing and layout.
- [VICI::Cmnd](#)
An API for libcommand.
- [VICI::Cron](#)
an API for libcron
- [VICI::Inst](#)
An API for the installer.
- [VICI::Ed](#)
An API for the vici editor.

Typedefs

- typedef const std::string & [VICI::csr](#)
short cut for string constants
- typedef int [VICI::NodeId](#)
Type for identifying a node of the flowchart.
- typedef int [VICI::ThreadId](#)
type for identifying a thread in the running script
- typedef std::vector< std::string > [VICI::ArgList](#)
Type for a list of command arguments and options.
- typedef std::shared_ptr< Factory > [VICI::FactoryPtr](#)
Shared pointer for factory.
- typedef std::shared_ptr
< EBNF_Factory > [VICI::EBNF::EBNF_FactoryPtr](#)
Shared pointer for [EBNF_Factory](#).
- typedef std::shared_ptr
< SyntaxFactory > [VICI::Syntax::SyntaxFactoryPtr](#)
Shared pointer for [SyntaxFactory](#).
- typedef std::shared_ptr
< ViciAdminFactory > [VICI::Admin::ViciAdminFactoryPtr](#)
Shared pointer for [Admin Factory](#).

- typedef std::shared_ptr
< SearchFactory > [VICI::Search::SearchFactoryPtr](#)
Shared pointer for [Search Factory](#).
- typedef std::shared_ptr
< InterpreterFactory > [VICI::Interp::InterpreterFactoryPtr](#)
Shared pointer for [Interpreter Factory](#).
- typedef long [VICI::Symbol::Colour](#)
Specialisation for holding colour values.
- typedef std::shared_ptr
< SymbolFactory > [VICI::Symbol::SymbolFactoryPtr](#)
Shared pointer for [Symbol Factory](#).
- typedef std::shared_ptr
< CanvasFactory > [VICI::Canvas::CanvasFactoryPtr](#)
Shared pointer for [Canvas Factory](#).
- typedef std::shared_ptr
< CommandFactory > [VICI::Cmnd::CommandFactoryPtr](#)
Shared pointer for [Command Factory](#).
- typedef std::shared_ptr
< SecureFactory > [VICI::Sec::SecureFactoryPtr](#)
Shared pointer for [Secure Factory](#).
- typedef std::shared_ptr
< CronFactory > [VICI::Cron::CronFactoryPtr](#)
Shared pointer for [Cron Factory](#).
- typedef std::shared_ptr
< InstallerFactory > [VICI::Inst::InstallerFactoryPtr](#)
Shared pointer for [Installer Factory](#).
- typedef std::shared_ptr
< ViciEditorFactory > [VICI::Ed::ViciEditorFactoryPtr](#)
Shared pointer for [Editor Factory](#).
- typedef std::shared_ptr
< ViciFactory > [VICI::ViciFactoryPtr](#)
Shared pointer for [Vici Factory](#).

Enumerations

- enum [VICI::Module](#) {
EBNF_Module, Syntax_Module, Admin_Module, Search_Module,
Command_Module, Symbol_Module, Canvas_Module, Secure_Module,
Cron_Module, Installer_Module, Interpreter_Module, Editor_Module,
Vici_Module }
An enum that lists the main modules of [VICI](#).
- enum [VICI::Symbol::Style](#) {
[VICI::Symbol::SymCommand](#), [VICI::Symbol::SymChoice](#), [VICI::Symbol::SymFuncRef](#), [VICI::Symbol::SymFunc](#),
[VICI::Symbol::SymVar](#), [VICI::Symbol::SymConst](#), [VICI::Symbol::SymMutex](#), [VICI::Symbol::SymSem](#),
[VICI::Symbol::SymFile](#), [VICI::Symbol::SymInline](#), [VICI::Symbol::SymPipe](#), [VICI::Symbol::SymDisplay](#),
[VICI::Symbol::SymLock](#), [VICI::Symbol::SymUnlock](#), [VICI::Symbol::SymWait](#), [VICI::Symbol::SymPost](#),
[VICI::Symbol::SymFlow](#), [VICI::Symbol::SymSuccess](#), [VICI::Symbol::SymFail](#), [VICI::Symbol::SymSignal](#),
[VICI::Symbol::SymStdIn](#), [VICI::Symbol::SymStdOut](#), [VICI::Symbol::SymStdErr](#) }
The possible styles for symbols and lines.

Functions

- void * [makeEBNFFactory](#) ()
create a new EBNF Factory
- void * [makeSyntaxFactory](#) ()
create a new Syntax Factory
- void * [makeAdminFactory](#) ()
create a new Admin Factory
- void * [makeSearchFactory](#) ()
create a new Search Factory
- void * [makeCommandFactory](#) ()
create a new Command Factory
- void * [makeSymbolFactory](#) ()
create a new Symbol Factory
- void * [makeCanvasFactory](#) ()
create a new Canvas Factory
- void * [makeSecureFactory](#) ()
create a new Secure Factory
- void * [makeCronFactory](#) ()
create a new Cron Factory
- void * [makeInstallerFactory](#) ()
create a new Installer Factory
- void * [makeInterpreterFactory](#) ()
create a new Interpreter Factory
- void * [makeEditorFactory](#) ()
create a new Editor Factory
- void * [makeViciFactory](#) ()
create a new Vici Factory

9.18.1 Detailed Description

Project wide declarations and definitions.

9.19 vx.h File Reference

An exception object with stream semantics.

```
#include <stdexcept>
#include <string>
#include <sstream>
#include <vector>
#include <errno.h>
#include <string.h>
```

Classes

- class [VICI::vxt< N >](#)
An exception object with severity levels.

Namespaces

- [VICI](#)

The namespace for the Visual Chart Interpreter project.

Macros

- `#define VX\(x\) vx\(x, __FILE__, __LINE__\)`

Macro to include file name and line number in exception messages.

Typedefs

- `using vx = VICI::vxt< VICI::vici_class >`

A project specific exception class.

Enumerations

- `enum VICI::Severity {
VICI::Emergency, VICI::Alert, VICI::Critical, VICI::Error,
VICI::Code, VICI::Warning, VICI::Notice, VICI::Info,
VICI::Debug }`

Severity levels for log messages.

9.19.1 Detailed Description

An exception object with stream semantics.

9.20 window.h File Reference

The API for handling windows between modules.

```
#include "vici.h"
```

Classes

- class [VICI::VWindow](#)

An implementation of [Window](#) for holding Qt's [QWidget](#) objects.

Namespaces

- [VICI](#)

The namespace for the Visual Chart Interpreter project.

9.20.1 Detailed Description

The API for handling windows between modules.

9.21 xini.h File Reference

A class for XML based configuration file.

```
#include "xml.h"
```

Classes

- class [VICI::cfi::XINI](#)
Load an XML configuration file.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::cfi](#)
The namespace for the Common Facilities Infrastructure components.

9.21.1 Detailed Description

A class for XML based configuration file.

9.22 xml.h File Reference

A simple C++ wrapper for libxml2.

```
#include <libxml/tree.h>  
#include <libxml/xpath.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include "stringy.h"  
#include <vector>
```

Classes

- class [VICI::cfi::XmlBuffer](#)
An abstract class defining a buffer object.
- class [VICI::cfi::Xml](#)
A C++ wrapper for libxml2.

Namespaces

- [VICI](#)
The namespace for the Visual Chart Interpreter project.
- [VICI::cfi](#)
The namespace for the Common Facilities Infrastructure components.

Macros

- `#define VICI_RELEASE "V0.1"`

Macro specifying the release version of Vici.

9.22.1 Detailed Description

A simple C++ wrapper for libxml2.

Index

- ~AbstractTestCase
 - VICI::cdi::AbstractTestCase, 45
- absolute
 - VICI::Path, 137
- AbstractScenario
 - VICI::cdi::AbstractScenario, 43
- AbstractTestCase
 - VICI::cdi::AbstractTestCase, 45
- Action
 - VICI::gth, 33
- action
 - VICI::gth::Adaptor, 47
- AdaptorT
 - VICI::gth::AdaptorT, 49
- addClient
 - VICI::stub::SymbolMgrStub, 173
 - VICI::Symbol::SymbolMgr, 171
- addScenario
 - VICI::cdi::Tester, 188
- addSignature
 - VICI::Sec::Secure, 158
 - VICI::stub::SecureStub, 160
- addTestCase
 - VICI::cdi::Tester, 189
- Alert
 - VICI, 24
- AnEvent
 - VICI::stub::AnEvent, 50
- append
 - VICI::Path, 137
- appendExt
 - VICI::Path, 137
- AsyncTestCase
 - VICI::cdi::AsyncTestCase, 51
- AsyncTestCaseT
 - VICI::cdi::AsyncTestCaseT, 52
- AsyncTestEventFn
 - VICI, 24
- attachCommand
 - VICI::stub::SymbolStub, 175
 - VICI::Symbol::Symbol, 166
- AutoRun
 - VICI::cfi::PlugInLib, 146
- BUFFER_SIZE
 - VICI::cfi::UDPSocket, 205
- base
 - VICI::Path, 137
- breakReached
 - VICI::Interp::InterpreterClient, 120
 - VICI::stub::CanvasStub, 63
 - VICI::stub::ViciStub, 218
- Button
 - VICI::gth, 33
- CallTrace
 - VICI::cdi::CallTrace, 55
- canvas.h, 237
- CanvasScene
 - VICI::CanvasScene, 61
- CanvasStub
 - VICI::stub::CanvasStub, 63
- CheckBox
 - VICI::gth, 33
- ChildProcess
 - VICI::cfi::ChildProcess, 68
- Choice
 - VICI::EbnfNode, 92
- clean
 - VICI, 24
- clone
 - VICI::stub::SymbolStub, 175
 - VICI::Symbol::Symbol, 166
- close
 - VICI::cfi::fdoutbuf, 106
- closeViciPlugin
 - plugin.h, 248
- cmdnError
 - VICI::Cmnd::CommandClient, 72
 - VICI::stub::ViciEditorStub, 213
- Code
 - VICI, 24
- ColorDialog
 - VICI::gth, 33
- ComboBox
 - VICI::gth, 33
- CommandStub
 - VICI::stub::CommandStub, 74
- config
 - VICI::cfi::XINI, 228
- configure
 - VICI::cdi::Tester, 189
 - VICI::cfi::XINI, 227
- create
 - VICI::cfi::Xml, 231
- Critical
 - VICI, 24
- CronStub
 - VICI::stub::CronStub, 77

- DISCOVERABLE
 - discover.h, [238](#)
- dataAck
 - VICI::Interp::Interpreter, [117](#)
 - VICI::stub::InterpreterStub, [123](#)
- dataReady
 - VICI::Interp::InterpreterClient, [120](#)
 - VICI::stub::CanvasStub, [63](#)
 - VICI::stub::ViciStub, [218](#)
- Debug
 - VICI, [24](#)
- debugMode
 - VICI::Interp::Interpreter, [117](#)
 - VICI::stub::InterpreterStub, [123](#)
- defaultAsyncTestEvent
 - VICI, [24](#)
- defined
 - VICI::Path, [138](#)
- deleteNode
 - VICI::cfi::Xml, [231](#)
- deregisterChild
 - VICI::cfi::ChildProcessMgr, [69](#)
- dir
 - VICI::Path, [138](#)
- dirty
 - VICI::cfi::Xml, [231](#)
- discover.h, [237](#)
 - DISCOVERABLE, [238](#)
- DiscoverPointer
 - VICI::cfi::DiscoverPointer, [83](#)
- Dock
 - VICI::gth, [33](#)
- dragged
 - VICI::Symbol::SymbolOwner, [174](#)
- draw
 - VICI::stub::SymbolStub, [176](#)
 - VICI::Symbol::Symbol, [166](#)
- Emergency
 - VICI, [24](#)
- enableTracing
 - VICI::stub::Dispatcher, [86](#)
- enqueueEvent
 - VICI::cdi::TestEventQueue, [192](#)
- Error
 - VICI, [24](#)
- event
 - VICI::cdi::TestEventQueue, [192](#)
- expandMacros
 - VICI, [25](#)
- exportTree
 - VICI::EbnfXml, [94](#)
- exportXml
 - VICI::EbnfTree, [93](#)
- ext
 - VICI::Path, [138](#)
- FD
 - VICI::cfi::FD, [99](#)
- fdinbuf
 - VICI::cfi::fdinbuf, [101](#)
- fdistream
 - VICI::cfi::fdistream, [103](#)
- fdostream
 - VICI::cfi::fdostream, [104](#)
- fdoutbuf
 - VICI::cfi::fdoutbuf, [105](#)
- fdstream.h, [238](#)
- fetch
 - VICI::cfi::DiscoveryMgr, [84](#)
- FileDialog
 - VICI::gth, [33](#)
- FileLogger
 - VICI::cfi::FileLogger, [107](#)
- find
 - VICI::cfi::Xml, [231](#)
- finished
 - VICI::cfi::ChildProcess, [68](#)
- flushBuffer
 - logbuff, [129](#)
- FontDialog
 - VICI::gth, [33](#)
- format
 - VICI::cfi::FormattingLogger, [108](#)
- getChild
 - VICI::cfi::Xml, [231](#)
- getChildren
 - VICI::cfi::Xml, [232](#)
- getConfigFilename
 - VICI::cfi::XINI, [227](#)
- getDefaultAttr
 - VICI::stub::SymbolMgrStub, [173](#)
 - VICI::Symbol::SymbolMgr, [171](#)
- getDefaultTextAttr
 - VICI::stub::SymbolMgrStub, [173](#)
 - VICI::Symbol::SymbolMgr, [171](#)
- getDtdIdentifiers
 - VICI::cfi::Xml, [232](#)
- getError
 - VICI::EBNF::EBNF, [87](#)
 - VICI::stub::EBNF_Stub, [89](#)
- getFactory
 - VICI::FactoryFactory, [98](#)
- getMode
 - VICI::cfi::PlugInLib, [146](#)
- getNodeContent
 - VICI::cfi::Xml, [232](#)
- getNodeName
 - VICI::cfi::Xml, [232](#)
- getPath
 - VICI::cfi::PlugInLib, [146](#)
 - VICI::cfi::XINI, [227](#)
- getPropDouble
 - VICI::cfi::Xml, [232](#)
- getPropInt
 - VICI::cfi::Xml, [232](#)
- getPropShort

- VICI::cfi::Xml, 233
- getPropString
 - VICI::cfi::Xml, 233
- getScenario
 - VICI::cdi::Tester, 189
- getStyle
 - VICI::stub::SymbolStub, 176
 - VICI::Symbol::Symbol, 166
- getSymbol
 - VICI::stub::SymbolMgrStub, 173
 - VICI::Symbol::SymbolMgr, 171
- getTest
 - VICI::cdi::Tester, 189
- getTestName
 - VICI::cdi::Tester, 189
- getTrace
 - VICI::stub::Dispatcher, 86
- getType
 - VICI::gth::Adaptor, 47
 - VICI::gth::AdaptorT, 49
- getVal
 - VICI::cfi::XINI, 227
- getVals
 - VICI::cfi::XINI, 228
- Grammar
 - VICI::EbnfNode, 92
- gth.h, 239
- handleEvent
 - VICI::cdi::AsyncTestCase, 51
- id
 - VICI::cdi::TestEvent, 191
- importTree
 - VICI::EbnfXml, 94
- importXml
 - VICI::EbnfTree, 93
- inUse
 - VICI::cfi::PlugInLib, 146
- Info
 - VICI, 24
- initTest
 - VICI::cdi::AsyncTestCase, 51
- initViciPlugin
 - plugin.h, 248
- install
 - VICI::cdi::AsyncTestCaseT, 53
 - VICI::cdi::ScenarioT, 152
 - VICI::cdi::TestCaseT, 187
- InstallerStub
 - VICI::stub::InstallerStub, 114
- instance
 - VICI::cdi::Tester, 189
 - VICI::cdi::TestEventQueue, 192
 - VICI::cfi::logstream, 132
 - VICI::stub::Dispatcher, 86
- InterpreterStub
 - VICI::stub::InterpreterStub, 123
- ipc.h, 240
- isChildOf
 - VICI::Path, 138
- isCommand
 - VICI::stub::SymbolStub, 176
 - VICI::Symbol::Symbol, 167
- Label
 - VICI::gth, 33
- libgth.h, 240
- libgui.h, 242
- libifstubs.h, 243
- LineEdit
 - VICI::gth, 33
- List
 - VICI::gth, 33
- ListView
 - VICI::gth, 33
- load
 - VICI::Canvas::Canvas, 56
 - VICI::cfi::PlugInMgr, 148
 - VICI::stub::CanvasStub, 63
- loadSnapshot
 - VICI::Interp::Interpreter, 117
 - VICI::stub::InterpreterStub, 125
- loaded
 - VICI::cfi::PlugInLib, 147
- log
 - VICI::cdi::Tester, 189
 - VICI::cdi::Tracer, 199
 - VICI::cfi::FileLogger, 107
 - VICI::cfi::logger, 130, 131
 - VICI::cfi::PlainFileLogger, 140
 - VICI::cfi::StdLogger, 164, 165
 - VICI::cfi::SystemLogger, 182
 - VICI::cfi::TraceLogger, 198
 - VICI::cfi::UDPLogger, 201, 202
- log.h, 245
- logbuff, 128
 - flushBuffer, 129
 - overflow, 129
 - sync, 129
- logstream
 - VICI::cfi::logstream, 132
- mBuffer
 - VICI::cfi::fdinbuf, 102
- make
 - VICI::cdi::ScenarioFT, 150
 - VICI::cdi::TestCaseFactory, 185
 - VICI::cdi::TestCaseFT, 185
 - VICI::cdi::TestFactory, 193
 - VICI::cdi::TestFT, 194
 - VICI::cfi::PlugInFactory, 143
 - VICI::cfi::PlugInFactoryT, 144
 - VICI::cfi::PlugInFamilyFactoryT, 145
- makeCron
 - VICI::Cron::CronFactory, 76
 - VICI::stub::CronStubFactory, 79
- makeEBNF

- VICI::EBNF::EBNF_Factory, 88
- VICI::stub::EBNF_Stub_Factory, 90
- makeInstaller
 - VICI::Inst::InstallerFactory, 113
 - VICI::stub::InstallerStubFactory, 115
- makeInterpreter
 - VICI::Interp::InterpreterFactory, 122
 - VICI::stub::InterpreterStubFactory, 127
- makeSearch
 - VICI::Search::SearchFactory, 156
 - VICI::stub::SearchStubFactory, 157
- makeSecure
 - VICI::Sec::SecureFactory, 159
 - VICI::stub::SecureStubFactory, 161
- makeSymbolMgr
 - VICI::stub::SymbolStubFactory, 177
 - VICI::Symbol::SymbolFactory, 170
- makeSyntax
 - VICI::stub::SyntaxStubFactory, 181
 - VICI::Syntax::SyntaxFactory, 179
- makeViciAdmin
 - VICI::Admin::ViciAdminFactory, 209
 - VICI::stub::ViciAdminStubFactory, 210
- MessageBox
 - VICI::gth, 33
- Mode
 - VICI::cfi::PlugInLib, 146
- Name
 - VICI::EbnfNode, 92
- name
 - VICI::Path, 138
- newNode
 - VICI::cfi::Xml, 233
- newSymbol
 - VICI::Canvas::CanvasClient, 59
 - VICI::stub::ViciEditorStub, 214
- newTextChild
 - VICI::cfi::Xml, 233
- noExt
 - VICI::Path, 138
- NodeType
 - VICI::EbnfNode, 92
- Notice
 - VICI, 24
- numberOfChildren
 - VICI::cfi::ChildProcessMgr, 70
- numberOfLiveChildren
 - VICI::cfi::ChildProcessMgr, 70
- OnDemand
 - VICI::cfi::PlugInLib, 146
- open
 - VICI::cfi::Xml, 233
- openDisplay
 - VICI::Interp::Interpreter, 117
 - VICI::stub::InterpreterStub, 125
- opened
 - VICI::Symbol::SymbolOwner, 174
- operator()
 - VICI::cdi::AbstractTestCase, 45
- Option
 - VICI::EbnfNode, 92
- optionsAndParameters
 - VICI::Cmnd::CommandClient, 72
 - VICI::stub::ViciEditorStub, 214
- overflow
 - logbuff, 129
 - VICI::cfi::fdoutbuf, 106
- parse
 - VICI::EBNF::EBNF, 87
 - VICI::stub::EBNF_Stub, 89
- parseTree.h, 246
- Path
 - VICI::Path, 137
- PlainFileLogger
 - VICI::cfi::PlainFileLogger, 139
- PlugInLib
 - VICI::cfi::PlugInLib, 146
- plugin.h, 246
 - closeViciPlugin, 248
 - initViciPlugin, 248
- proc.h, 248
- processTerminated
 - VICI::cfi::ProcessOwner, 148
- Production
 - VICI::EbnfNode, 92
- Quotation
 - VICI::EbnfNode, 92
- recv
 - VICI::cfi::UDPSocket, 204
- registerChild
 - VICI::cfi::ChildProcessMgr, 70
- registerEvent
 - VICI::stub::Dispatcher, 86
- registerFactory
 - VICI::FactoryFactory, 98
- registerNamespace
 - VICI::cfi::Xml, 233
- RegnFn
 - VICI::gth, 34
- Repetition
 - VICI::EbnfNode, 92
- report
 - VICI::cfi::FD, 99
- reportError
 - VICI::Interp::InterpreterClient, 120
 - VICI::stub::CanvasStub, 64
 - VICI::stub::ViciStub, 218
- run
 - VICI::Interp::Interpreter, 118
 - VICI::stub::InterpreterStub, 125
- runTest
 - VICI::cdi::AbstractTestCase, 45
 - VICI::gth::DefaultTestCase, 82

- safeToSave
 - VICI::cfi::Xml, [234](#)
- save
 - VICI::Canvas::Canvas, [56](#)
 - VICI::cfi::DiscoveryMgr, [85](#)
 - VICI::cfi::Xml, [234](#)
 - VICI::stub::CanvasStub, [64](#)
- saveSnapshot
 - VICI::Interp::Interpreter, [118](#)
 - VICI::stub::InterpreterStub, [125](#)
- ScenarioT
 - VICI::cdi::ScenarioT, [152](#)
- Script
 - VICI::gth, [33](#)
- SearchStub
 - VICI::stub::SearchStub, [157](#)
- selectedCommand
 - VICI::Search::SearchClient, [155](#)
 - VICI::stub::ViciEditorStub, [214](#)
- selection
 - VICI::Canvas::Canvas, [56](#)
 - VICI::Cmnd::Command, [71](#)
 - VICI::stub::CanvasStub, [64](#)
 - VICI::stub::CommandStub, [74](#)
 - VICI::stub::ViciEditorStub, [214](#)
 - VICI::Symbol::SymbolClient, [169](#)
- Semaphore
 - VICI::cfi::Semaphore, [162](#)
- SemaphoreLock
 - VICI::cfi::SemaphoreLock, [163](#)
- send
 - VICI::cfi::UDPServerSocket, [203](#)
 - VICI::cfi::UDPSTcpSocket, [204](#)
- sendEvent
 - VICI::stub::Dispatcher, [86](#)
- Sequence
 - VICI::EbnfNode, [92](#)
- setArgs
 - VICI::cfi::ChildProcess, [68](#)
- setAttributes
 - VICI::stub::SymbolStub, [176](#)
 - VICI::Symbol::Symbol, [167](#)
- setBreak
 - VICI::Interp::Interpreter, [118](#)
 - VICI::stub::InterpreterStub, [125](#)
- setCDATAContent
 - VICI::cfi::Xml, [234](#)
- setCommand
 - VICI::Canvas::Canvas, [56](#)
 - VICI::Cmnd::Command, [71](#)
 - VICI::stub::CanvasStub, [64](#)
 - VICI::stub::CommandStub, [74](#)
- setCompression
 - VICI::cfi::Xml, [234](#)
- setContent
 - VICI::cfi::Xml, [234](#)
- setCurrentFile
 - VICI::Cron::Cron, [76](#)
 - VICI::Inst::Installer, [112](#)
 - VICI::stub::CronStub, [78](#)
 - VICI::stub::InstallerStub, [114](#)
- setCursor
 - VICI::Interp::InterpreterClient, [120](#)
 - VICI::stub::CanvasStub, [64](#)
 - VICI::stub::ViciStub, [218](#)
- setDirty
 - VICI::cfi::Xml, [234](#)
- setDtd
 - VICI::cfi::Xml, [235](#)
- setExecution
 - VICI::Canvas::Canvas, [56](#)
 - VICI::stub::CanvasStub, [64](#)
- setFile
 - VICI::Interp::InterpreterClient, [121](#)
 - VICI::stub::CanvasStub, [66](#)
 - VICI::stub::ViciStub, [219](#)
- setHighlight
 - VICI::stub::SymbolStub, [176](#)
 - VICI::Symbol::Symbol, [167](#)
- setInterval
 - VICI::Interp::Interpreter, [118](#)
 - VICI::stub::InterpreterStub, [126](#)
- setMode
 - VICI::cfi::PlugInLib, [147](#)
- setNodeId
 - VICI::stub::SymbolStub, [177](#)
 - VICI::Symbol::Symbol, [167](#)
- setPosn
 - VICI::Interp::Interpreter, [118](#)
 - VICI::stub::InterpreterStub, [126](#)
- setProp
 - VICI::cfi::Xml, [235](#)
- setTest
 - VICI::cdi::Tester, [190](#)
- setValue
 - VICI::Interp::Interpreter, [119](#)
 - VICI::Interp::InterpreterClient, [121](#)
 - VICI::stub::CanvasStub, [66](#)
 - VICI::stub::InterpreterStub, [126](#)
 - VICI::stub::ViciStub, [219](#)
- Severity
 - VICI, [24](#)
- severity
 - VICI::cfi::logstream, [133](#)
- show
 - VICI::stub::SyntaxStub, [180](#)
 - VICI::Syntax::Syntax, [178](#)
- SpinBox
 - VICI::gth, [33](#)
- split
 - VICI, [25](#)
 - VICI::Path, [138](#)
- Splitter
 - VICI::gth, [33](#)
- StayResident
 - VICI::cfi::PlugInLib, [146](#)

- StdLogger
 - VICI::cfi::StdLogger, 164
- step
 - VICI::Interp::Interpreter, 119
 - VICI::stub::InterpreterStub, 126
- stringy.h, 249
- Style
 - VICI::Symbol, 38
- summary
 - VICI::cdi::Tester, 190
- SymChoice
 - VICI::Symbol, 38
- SymCommand
 - VICI::Symbol, 38
- SymConst
 - VICI::Symbol, 38
- SymDisplay
 - VICI::Symbol, 38
- SymFail
 - VICI::Symbol, 38
- SymFile
 - VICI::Symbol, 38
- SymFlow
 - VICI::Symbol, 38
- SymFunc
 - VICI::Symbol, 38
- SymFuncRef
 - VICI::Symbol, 38
- SymInline
 - VICI::Symbol, 38
- SymLock
 - VICI::Symbol, 38
- SymMutex
 - VICI::Symbol, 38
- SymPipe
 - VICI::Symbol, 38
- SymPost
 - VICI::Symbol, 38
- SymSem
 - VICI::Symbol, 38
- SymSignal
 - VICI::Symbol, 38
- SymStdErr
 - VICI::Symbol, 39
- SymStdIn
 - VICI::Symbol, 39
- SymStdOut
 - VICI::Symbol, 39
- SymSuccess
 - VICI::Symbol, 38
- SymUnlock
 - VICI::Symbol, 38
- SymVar
 - VICI::Symbol, 38
- SymWait
 - VICI::Symbol, 38
- symbolAttr
 - VICI::Canvas::Canvas, 58
 - VICI::stub::CanvasStub, 66
 - VICI::stub::ViciEditorStub, 214
 - VICI::Symbol::SymbolClient, 169
- SymbolMgrStub
 - VICI::stub::SymbolMgrStub, 172
- SymbolStub
 - VICI::stub::SymbolStub, 175
- sync
 - logbuff, 129
- SyntaxStub
 - VICI::stub::SyntaxStub, 180
- SystemLogger
 - VICI::cfi::SystemLogger, 182
- Table
 - VICI::gth, 33
- Tabs
 - VICI::gth, 33
- Terminal
 - VICI::EbnfNode, 92
- test
 - VICI::cdi::AbstractTestCase, 45
- test.h, 249
- TestCaseT
 - VICI::cdi::TestCaseT, 186
- TestEvent
 - VICI::cdi::TestEvent, 191
- testScenario
 - VICI::cdi::Tester, 190
- testmgr.h, 250
- TextEdit
 - VICI::gth, 33
- textAttr
 - VICI::Canvas::Canvas, 58
 - VICI::stub::CanvasStub, 66
 - VICI::stub::ViciEditorStub, 214
 - VICI::Symbol::SymbolClient, 169
- timeStamp
 - VICI::cfi::FormattingLogger, 109
- timedOut
 - VICI::cdi::AsyncTestCase, 51
- Trace
 - VICI::cdi::Trace, 197
- trace
 - VICI::stub::Dispatcher, 86
- trace.h, 251
- TraceLogger
 - VICI::cfi::TraceLogger, 198
- Tree
 - VICI::gth, 33
- trim
 - VICI, 25
- typeOfNode
 - VICI::EbnfNode, 92
- UDPCliientSocket
 - VICI::cfi::UDPCliientSocket, 200
- UDPLogger
 - VICI::cfi::UDPLogger, 201

- UDPServerSocket
 - VICI::cfi::UDPServerSocket, 203
- udpsocket.h, 252
- Undefined
 - VICI::EbnfNode, 92
- up
 - VICI::Path, 138
- VICI
 - Alert, 24
 - Code, 24
 - Critical, 24
 - Debug, 24
 - Emergency, 24
 - Error, 24
 - Info, 24
 - Notice, 24
 - Warning, 24
- VICI::EbnfNode
 - Choice, 92
 - Grammar, 92
 - Name, 92
 - Option, 92
 - Production, 92
 - Quotation, 92
 - Repetition, 92
 - Sequence, 92
 - Terminal, 92
 - Undefined, 92
- VICI::Symbol
 - SymChoice, 38
 - SymCommand, 38
 - SymConst, 38
 - SymDisplay, 38
 - SymFail, 38
 - SymFile, 38
 - SymFlow, 38
 - SymFunc, 38
 - SymFuncRef, 38
 - SymInline, 38
 - SymLock, 38
 - SymMutex, 38
 - SymPipe, 38
 - SymPost, 38
 - SymSem, 38
 - SymSignal, 38
 - SymStdErr, 39
 - SymStdIn, 39
 - SymStdOut, 39
 - SymSuccess, 38
 - SymUnlock, 38
 - SymVar, 38
 - SymWait, 38
- VICI::cfi::PlugInLib
 - AutoRun, 146
 - OnDemand, 146
 - StayResident, 146
- VICI::gth
 - Action, 33
 - Button, 33
 - CheckBox, 33
 - ColorDialog, 33
 - ComboBox, 33
 - Dock, 33
 - FileDialog, 33
 - FontDialog, 33
 - Label, 33
 - LineEdit, 33
 - List, 33
 - ListView, 33
 - MessageBox, 33
 - Script, 33
 - SpinBox, 33
 - Splitter, 33
 - Table, 33
 - Tabs, 33
 - TextEdit, 33
 - Tree, 33
 - View, 33
 - Window, 33
- VICI, 21
 - AsyncTestEventFn, 24
 - clean, 24
 - defaultAsyncTestEvent, 24
 - expandMacros, 25
 - Severity, 24
 - split, 25
 - trim, 25
- VICI::AboutDialog, 41
- VICI::Admin, 25
- VICI::Admin::ViciAdmin, 208
- VICI::Admin::ViciAdminFactory, 208
 - makeViciAdmin, 209
- VICI::Canvas, 26
- VICI::Canvas::Canvas, 55
 - load, 56
 - save, 56
 - selection, 56
 - setCommand, 56
 - setExecution, 56
 - symbolAttr, 58
 - textAttr, 58
- VICI::Canvas::CanvasClient, 58
 - newSymbol, 59
- VICI::Canvas::CanvasFactory, 59
- VICI::CanvasScene, 60
 - CanvasScene, 61
- VICI::Cmnd, 29
- VICI::Cmnd::Command, 70
 - selection, 71
 - setCommand, 71
- VICI::Cmnd::CommandClient, 71
 - cmndError, 72
 - optionsAndParameters, 72
- VICI::Cmnd::CommandFactory, 72
- VICI::Cron, 30
- VICI::Cron::Cron, 75

- setCurrentFile, 76
- VICI::Cron::CronFactory, 76
 - makeCron, 76
- VICI::EBNF, 30
- VICI::EBNF::EBNF, 87
 - getError, 87
 - parse, 87
 - validate, 88
- VICI::EBNF::EBNF_Factory, 88
 - makeEBNF, 88
- VICI::EBNF::ParseTree, 135
- VICI::EbnfNode, 91
 - NodeType, 92
 - typeOfNode, 92
- VICI::EbnfTree, 93
 - exportXml, 93
 - importXml, 93
- VICI::EbnfXml, 94
 - exportTree, 94
 - importTree, 94
- VICI::Ed, 31
- VICI::Ed::ViciEditor, 210
- VICI::Ed::ViciEditorFactory, 211
- VICI::Factory, 96
- VICI::FactoryFactory, 97
 - getFactory, 98
 - registerFactory, 98
- VICI::GTHWindowWidget, 111
- VICI::Inst, 34
- VICI::Inst::Installer, 112
 - setCurrentFile, 112
- VICI::Inst::InstallerFactory, 113
 - makeInstaller, 113
- VICI::Interp, 34
- VICI::Interp::Interpreter, 115
 - dataAck, 117
 - debugMode, 117
 - loadSnapshot, 117
 - openDisplay, 117
 - run, 118
 - saveSnapshot, 118
 - setBreak, 118
 - setInterval, 118
 - setPosn, 118
 - setValue, 119
 - step, 119
- VICI::Interp::InterpreterClient, 119
 - breakReached, 120
 - dataReady, 120
 - reportError, 120
 - setCursor, 120
 - setFile, 121
 - setValue, 121
- VICI::Interp::InterpreterFactory, 121
 - makeInterpreter, 122
- VICI::ItemDelegate, 127
- VICI::Metrics, 133
- VICI::Path, 136
 - absolute, 137
 - append, 137
 - appendExt, 137
 - base, 137
 - defined, 138
 - dir, 138
 - ext, 138
 - isChildOf, 138
 - name, 138
 - noExt, 138
 - Path, 137
 - split, 138
 - up, 138
- VICI::Scene, 152
- VICI::Search, 35
- VICI::Search::Search, 154
- VICI::Search::SearchClient, 154
 - selectedCommand, 155
- VICI::Search::SearchFactory, 155
 - makeSearch, 156
- VICI::Sec, 35
- VICI::Sec::Secure, 158
 - addSignature, 158
 - verifySignature, 159
- VICI::Sec::SecureFactory, 159
 - makeSecure, 159
- VICI::SignalToQtSignal, 163
- VICI::Symbol, 37
 - Style, 38
- VICI::Symbol::Symbol, 165
 - attachCommand, 166
 - clone, 166
 - draw, 166
 - getStyle, 166
 - isCommand, 167
 - setAttributes, 167
 - setHighlight, 167
 - setNodeld, 167
- VICI::Symbol::SymbolAttributes, 167
- VICI::Symbol::SymbolClient, 168
 - selection, 169
 - symbolAttr, 169
 - textAttr, 169
- VICI::Symbol::SymbolFactory, 169
 - makeSymbolMgr, 170
- VICI::Symbol::SymbolMgr, 170
 - addClient, 171
 - getDefaultAttr, 171
 - getDefaultTextAttr, 171
 - getSymbol, 171
- VICI::Symbol::SymbolOwner, 174
 - dragged, 174
 - opened, 174
- VICI::Symbol::TextAttributes, 196
- VICI::Syntax, 39
- VICI::Syntax::Syntax, 178
 - show, 178
- VICI::Syntax::SyntaxFactory, 179

- makeSyntax, 179
- VICI::VDialog, 205
- VICI::VEditList, 206
- VICI::VMainWindow, 220
- VICI::VWindow, 221
 - VWindow, 221
- VICI::Vici, 207
- VICI::ViciFactory, 216
- VICI::WidgetMgr, 223
- VICI::WidgetMgrClient, 224
- VICI::Window, 224
- VICI::cdi, 26
- VICI::cdi::AbstractScenario, 42
 - AbstractScenario, 43
 - willRunTests, 43
- VICI::cdi::AbstractTest, 43
- VICI::cdi::AbstractTestCase, 44
 - ~AbstractTestCase, 45
 - AbstractTestCase, 45
 - operator(), 45
 - runTest, 45
 - test, 45
- VICI::cdi::AsyncTestCase, 50
 - AsyncTestCase, 51
 - handleEvent, 51
 - initTest, 51
 - timedOut, 51
- VICI::cdi::AsyncTestCaseT
 - AsyncTestCaseT, 52
 - install, 53
- VICI::cdi::AsyncTestCaseT< T >, 52
- VICI::cdi::CallTrace, 54
 - CallTrace, 55
- VICI::cdi::DefaultScenario, 80
- VICI::cdi::DefaultTest, 80
- VICI::cdi::NullTrace, 135
- VICI::cdi::ScenarioFT
 - make, 150
- VICI::cdi::ScenarioFT< T >, 150
- VICI::cdi::ScenarioFactory, 149
- VICI::cdi::ScenarioResults, 151
- VICI::cdi::ScenarioT
 - install, 152
 - ScenarioT, 152
- VICI::cdi::ScenarioT< T >, 151
- VICI::cdi::TestCaseFT
 - make, 185
- VICI::cdi::TestCaseFT< T >, 185
- VICI::cdi::TestCaseFactory, 184
 - make, 185
- VICI::cdi::TestCaseT
 - install, 187
 - TestCaseT, 186
- VICI::cdi::TestCaseT< T >, 186
- VICI::cdi::TestEvent, 190
 - id, 191
 - TestEvent, 191
- VICI::cdi::TestEventQueue, 191
 - enqueueEvent, 192
 - event, 192
 - instance, 192
- VICI::cdi::TestFT
 - make, 194
- VICI::cdi::TestFT< T >, 193
- VICI::cdi::TestFactory, 193
 - make, 193
- VICI::cdi::TestT< T >, 195
- VICI::cdi::Tester, 187
 - addScenario, 188
 - addTestCase, 189
 - configure, 189
 - getScenario, 189
 - getTest, 189
 - getTestName, 189
 - instance, 189
 - log, 189
 - setTest, 190
 - summary, 190
 - testScenario, 190
- VICI::cdi::Trace, 196
 - Trace, 197
- VICI::cdi::Tracer, 199
 - log, 199
- VICI::cdi::scenario_exception, 149
- VICI::cdi::test_exception, 183
- VICI::cfi, 27
- VICI::cfi::AbstractChildProcess, 41
- VICI::cfi::AutoRunPlugIn, 54
- VICI::cfi::ChildProcess, 67
 - ChildProcess, 68
 - finished, 68
 - setArgs, 68
- VICI::cfi::ChildProcessMgr, 69
 - deregisterChild, 69
 - numberOfChildren, 70
 - numberOfLiveChildren, 70
 - registerChild, 70
- VICI::cfi::DiscoverPointer, 82
 - DiscoverPointer, 83
- VICI::cfi::Discoverable, 82
- VICI::cfi::DiscoveryMgr, 83
 - fetch, 84
 - save, 85
- VICI::cfi::FD, 98
 - FD, 99
 - report, 99
- VICI::cfi::FileLogger, 106
 - FileLogger, 107
 - log, 107
- VICI::cfi::FormattingLogger, 108
 - format, 108
 - timeStamp, 109
- VICI::cfi::PlainFileLogger, 139
 - log, 140
 - PlainFileLogger, 139
- VICI::cfi::PlugIn, 140

- VICI::cfi::PlugInDescriptor, 141
- VICI::cfi::PlugInDetails, 142
- VICI::cfi::PlugInFactory, 142
 - make, 143
- VICI::cfi::PlugInFactoryT
 - make, 144
- VICI::cfi::PlugInFactoryT< F, P >, 143
- VICI::cfi::PlugInFamilyFactoryT
 - make, 145
- VICI::cfi::PlugInFamilyFactoryT< F >, 144
- VICI::cfi::PlugInLib, 145
 - getMode, 146
 - getPath, 146
 - inUse, 146
 - loaded, 147
 - Mode, 146
 - PlugInLib, 146
 - setMode, 147
- VICI::cfi::PlugInMgr, 147
 - load, 148
- VICI::cfi::ProcessOwner, 148
 - processTerminated, 148
- VICI::cfi::Semaphore, 162
 - Semaphore, 162
- VICI::cfi::SemaphoreLock, 162
 - SemaphoreLock, 163
- VICI::cfi::StdLogger, 164
 - log, 164, 165
 - StdLogger, 164
- VICI::cfi::SystemLogger, 181
 - log, 182
 - SystemLogger, 182
- VICI::cfi::TraceLogger, 197
 - log, 198
 - TraceLogger, 198
- VICI::cfi::UDPClientSocket, 200
 - UDPClientSocket, 200
- VICI::cfi::UDPLogger, 201
 - log, 201, 202
 - UDPLogger, 201
- VICI::cfi::UDPServerSocket, 202
 - send, 203
 - UDPServerSocket, 203
- VICI::cfi::UDPSocket, 203
 - BUFFER_SIZE, 205
 - recv, 204
 - send, 204
- VICI::cfi::XINI, 225
 - config, 228
 - configure, 227
 - getConfigFilename, 227
 - getPath, 227
 - getVal, 227
 - getVals, 228
- VICI::cfi::Xml, 228
 - create, 231
 - deleteNode, 231
 - dirty, 231
 - find, 231
 - getChild, 231
 - getChildren, 232
 - getDtdIdentifiers, 232
 - getNodeContent, 232
 - getNodeName, 232
 - getPropDouble, 232
 - getPropInt, 232
 - getPropShort, 233
 - getPropString, 233
 - newNode, 233
 - newTextChild, 233
 - open, 233
 - registerNamespace, 233
 - safeToSave, 234
 - save, 234
 - setCDATAContent, 234
 - setCompression, 234
 - setContent, 234
 - setDirty, 234
 - setDtd, 235
 - setProp, 235
- VICI::cfi::XmlBuffer, 236
- VICI::cfi::fdinbuf, 99
 - fdinbuf, 101
 - mBuffer, 102
- VICI::cfi::fdistream, 102
 - fdistream, 103
- VICI::cfi::fdostream, 103
 - fdostream, 104
- VICI::cfi::fdoutbuf, 104
 - close, 106
 - fdoutbuf, 105
 - overflow, 106
- VICI::cfi::logger, 130
 - log, 130, 131
- VICI::cfi::logstream, 131
 - instance, 132
 - logstream, 132
 - severity, 133
- VICI::gth, 31
 - RegnFn, 34
 - WidgetType, 33
- VICI::gth::Adaptor, 46
 - action, 47
 - getType, 47
- VICI::gth::AdaptorST< wt >, 47
- VICI::gth::AdaptorT
 - AdaptorT, 49
 - getType, 49
- VICI::gth::AdaptorT< wt, T >, 48
- VICI::gth::DefaultTestCase, 81
 - runTest, 82
- VICI::gth::GTHTest, 109
- VICI::gth::GTHTestCase, 110
- VICI::gth::LuaScript, 133
- VICI::gth::MouseEventReporter, 134
- VICI::gth::ScriptXML, 153

- VICI::gth::TestAction, 183
- VICI::gth::TestsForWindow, 194
- VICI::gth::UserEscaper, 205
- VICI::stub, 36
- VICI::stub::AnEvent
 - AnEvent, 50
- VICI::stub::AnEvent< T >, 49
- VICI::stub::CanvasStub, 62
 - breakReached, 63
 - CanvasStub, 63
 - dataReady, 63
 - load, 63
 - reportError, 64
 - save, 64
 - selection, 64
 - setCommand, 64
 - setCursor, 64
 - setExecution, 64
 - setFile, 66
 - setValue, 66
 - symbolAttr, 66
 - textAttr, 66
- VICI::stub::CanvasStubFactory, 67
- VICI::stub::CommandStub, 73
 - CommandStub, 74
 - selection, 74
 - setCommand, 74
- VICI::stub::CommandStubFactory, 74
- VICI::stub::CronStub, 77
 - CronStub, 77
 - setCurrentFile, 78
- VICI::stub::CronStubFactory, 79
 - makeCron, 79
- VICI::stub::Dispatcher, 85
 - enableTracing, 86
 - getTrace, 86
 - instance, 86
 - registerEvent, 86
 - sendEvent, 86
 - trace, 86
- VICI::stub::EBNF_Stub, 89
 - getError, 89
 - parse, 89
 - validate, 90
- VICI::stub::EBNF_Stub_Factory, 90
 - makeEBNF, 90
- VICI::stub::Event, 96
- VICI::stub::InstallerStub, 114
 - InstallerStub, 114
 - setCurrentFile, 114
- VICI::stub::InstallerStubFactory, 115
 - makeInstaller, 115
- VICI::stub::InterpreterStub, 122
 - dataAck, 123
 - debugMode, 123
 - InterpreterStub, 123
 - loadSnapshot, 125
 - openDisplay, 125
 - run, 125
 - saveSnapshot, 125
 - setBreak, 125
 - setInterval, 126
 - setPosn, 126
 - setValue, 126
 - step, 126
- VICI::stub::InterpreterStubFactory, 126
 - makeInterpreter, 127
- VICI::stub::SearchStub, 156
 - SearchStub, 157
- VICI::stub::SearchStubFactory, 157
 - makeSearch, 157
- VICI::stub::SecureStub, 160
 - addSignature, 160
 - verifySignature, 160
- VICI::stub::SecureStubFactory, 161
 - makeSecure, 161
- VICI::stub::SymbolMgrStub, 172
 - addClient, 173
 - getDefaultAttr, 173
 - getDefaultTextAttr, 173
 - getSymbol, 173
 - SymbolMgrStub, 172
- VICI::stub::SymbolStub, 174
 - attachCommand, 175
 - clone, 175
 - draw, 176
 - getStyle, 176
 - isCommand, 176
 - setAttributes, 176
 - setHighlight, 176
 - setNodeld, 177
 - SymbolStub, 175
- VICI::stub::SymbolStubFactory, 177
 - makeSymbolMgr, 177
- VICI::stub::SyntaxStub, 179
 - show, 180
 - SyntaxStub, 180
- VICI::stub::SyntaxStubFactory, 181
 - makeSyntax, 181
- VICI::stub::ViciAdminStub, 209
- VICI::stub::ViciAdminStubFactory, 209
 - makeViciAdmin, 210
- VICI::stub::ViciEditorStub, 212
 - cmndError, 213
 - newSymbol, 214
 - optionsAndParameters, 214
 - selectedCommand, 214
 - selection, 214
 - symbolAttr, 214
 - textAttr, 214
- VICI::stub::ViciEditorStubFactory, 216
- VICI::stub::ViciStub, 217
 - breakReached, 218
 - dataReady, 218
 - reportError, 218
 - setCursor, 218

- setFile, [219](#)
- setValue, [219](#)
- VICI::stub::ViciStubFactory, [219](#)
- VICI::stub::WindowStub, [225](#)
- VICI::vxt
 - vxt, [223](#)
- VICI::vxt< N >, [221](#)
- VWindow
 - VICI::VWindow, [221](#)
- validate
 - VICI::EBNF::EBNF, [88](#)
 - VICI::stub::EBNF_Stub, [90](#)
- verifySignature
 - VICI::Sec::Secure, [159](#)
 - VICI::stub::SecureStub, [160](#)
- vici.h, [253](#)
- View
 - VICI::gth, [33](#)
- vx.h, [257](#)
- vxt
 - VICI::vxt, [223](#)
- Warning
 - VICI, [24](#)
- WidgetType
 - VICI::gth, [33](#)
- willRunTests
 - VICI::cdi::AbstractScenario, [43](#)
- Window
 - VICI::gth, [33](#)
- window.h, [258](#)
- xini.h, [259](#)
- xml.h, [259](#)