

FMI Library: part of JModelica.org

Generated by Doxygen 1.7.6.1

Tue Oct 28 2014 11:16:13

Contents

1	FMI Library: part of JModelica.org	1
1.1	Summary	1
1.2	Configuring and building	1
1.3	Automatic tests	4
1.4	Using logs	5
1.5	License	6
1.6	Acknowledgements	7
2	Module Index	9
2.1	Modules	9
3	Directory Hierarchy	11
3.1	Directories	11
4	Data Structure Index	13
4.1	Data Structures	13
5	File Index	15
5.1	File List	15
6	Module Documentation	19
6.1	FMI import library	19
6.2	Utilities	20
6.2.1	Typedef Documentation	21
6.2.1.1	jm_string	21
6.2.1.2	jm_voidp	21
6.2.1.3	jm_name_ID_map_t	21

6.2.2	Enumeration Type Documentation	21
6.2.2.1	fmi_version_enu_t	21
6.2.2.2	jm_status_enu_t	22
6.2.2.3	jm_log_level_enu_t	22
6.2.3	Function Documentation	22
6.2.3.1	fmi_construct_dll_dir_name	22
6.2.3.2	fmi_construct_dll_file_name	23
6.2.3.3	fmi_version_to_string	23
6.2.3.4	jm_log_level_to_string	23
6.3	A basic stack	24
6.3.1	Define Documentation	24
6.3.1.1	jm_stack	24
6.3.1.2	jm_stack_alloc	24
6.3.1.3	jm_stack_free	25
6.3.1.4	jm_stack_init	25
6.3.1.5	jm_stack_free_data	25
6.3.1.6	jm_stack_get_size	26
6.3.1.7	jm_stack_reserve	26
6.3.1.8	jm_stack_push	26
6.3.1.9	jm_stack_is_empty	26
6.3.1.10	jm_stack_pop	26
6.3.1.11	jm_stack_top	27
6.3.1.12	jm_stack_foreach	27
6.3.1.13	JM_STACK_MINIMAL_CAPACITY	27
6.3.1.14	JM_STACK_MAX_MEMORY_CHUNK	27
6.3.1.15	jm_stack_declare_template	27
6.4	FMI 1.0 import interface	28
6.4.1	Detailed Description	28
6.5	Handling of variable lists	29
6.5.1	Detailed Description	30
6.5.2	Typedef Documentation	30
6.5.2.1	fmi1_import_variable_filter_function_ft	30
6.5.3	Function Documentation	30
6.5.3.1	fmi1_import_alloc_variable_list	30

6.5.3.2	fmi1_import_free_variable_list	30
6.5.3.3	fmi1_import_clone_variable_list	30
6.5.3.4	fmi1_import_get_variable_list_size	31
6.5.3.5	fmi1_import_get_value_referece_list	31
6.5.3.6	fmi1_import_get_variable	31
6.5.3.7	fmi1_import_get_sublist	31
6.5.3.8	fmi1_import_filter_variables	31
6.5.3.9	fmi1_import_join_var_list	32
6.5.3.10	fmi1_import_append_to_var_list	32
6.5.3.11	fmi1_import_prepend_to_var_list	32
6.5.3.12	fmi1_import_var_list_push_back	32
6.6	FMI 2.0 import interface	33
6.6.1	Detailed Description	33
6.7	Handling of variable lists	34
6.7.1	Detailed Description	35
6.7.2	Typedef Documentation	35
6.7.2.1	fmi2_import_variable_filter_function_ft	35
6.7.3	Function Documentation	35
6.7.3.1	fmi2_import_alloc_variable_list	35
6.7.3.2	fmi2_import_free_variable_list	35
6.7.3.3	fmi2_import_clone_variable_list	35
6.7.3.4	fmi2_import_get_variable_list_size	36
6.7.3.5	fmi2_import_get_value_referece_list	36
6.7.3.6	fmi2_import_get_variable	36
6.7.3.7	fmi2_import_get_sublist	36
6.7.3.8	fmi2_import_filter_variables	36
6.7.3.9	fmi2_import_join_var_list	37
6.7.3.10	fmi2_import_append_to_var_list	37
6.7.3.11	fmi2_import_prepend_to_var_list	37
6.7.3.12	fmi2_import_var_list_push_back	37
6.8	Functions and types supporting FMI 1.0 processing.	38
6.8.1	Define Documentation	40
6.8.1.1	fmiComponent	40
6.8.1.2	fmiValueReference	40

6.8.1.3	fmiReal	40
6.8.1.4	fmiInteger	41
6.8.1.5	fmiBoolean	41
6.8.1.6	fmiString	41
6.8.2	Typedef Documentation	41
6.8.2.1	fmi1_callback_logger_ft	41
6.8.2.2	fmi1_callback_allocate_memory_ft	41
6.8.2.3	fmi1_callback_free_memory_ft	41
6.8.2.4	fmi1_step_finished_ft	41
6.8.2.5	fmi1_get_version_ft	41
6.8.2.6	fmi1_set_debug_logging_ft	42
6.8.2.7	fmi1_set_real_ft	42
6.8.2.8	fmi1_set_integer_ft	42
6.8.2.9	fmi1_set_boolean_ft	42
6.8.2.10	fmi1_set_string_ft	42
6.8.2.11	fmi1_get_real_ft	42
6.8.2.12	fmi1_get_integer_ft	42
6.8.2.13	fmi1_get_boolean_ft	42
6.8.2.14	fmi1_get_string_ft	42
6.8.2.15	fmi1_get_model_typesPlatform_ft	43
6.8.2.16	fmi1_instantiate_model_ft	43
6.8.2.17	fmi1_free_model_instance_ft	43
6.8.2.18	fmi1_set_time_ft	43
6.8.2.19	fmi1_set_continuous_states_ft	43
6.8.2.20	fmi1_completed_integrator_step_ft	43
6.8.2.21	fmi1_initialize_ft	43
6.8.2.22	fmi1_get_derivatives_ft	43
6.8.2.23	fmi1_get_event_indicators_ft	43
6.8.2.24	fmi1_event_update_ft	44
6.8.2.25	fmi1_get_continuous_states_ft	44
6.8.2.26	fmi1_get_nominal_continuousStates_ft	44
6.8.2.27	fmi1_get_state_valueReferences_ft	44
6.8.2.28	fmi1_terminate_ft	44
6.8.2.29	fmi1_get_types_platform_ft	44

6.8.2.30	fmi1_instantiate_slave_ft	44
6.8.2.31	fmi1_initialize_slave_ft	44
6.8.2.32	fmi1_terminate_slave_ft	45
6.8.2.33	fmi1_reset_slave_ft	45
6.8.2.34	fmi1_free_slave_instance_ft	45
6.8.2.35	fmi1_set_real_inputDerivatives_ft	45
6.8.2.36	fmi1_get_real_outputDerivatives_ft	45
6.8.2.37	fmi1_cancel_step_ft	45
6.8.2.38	fmi1_do_step_ft	45
6.8.2.39	fmi1_get_status_ft	45
6.8.2.40	fmi1_get_real_status_ft	45
6.8.2.41	fmi1_get_integer_status_ft	46
6.8.2.42	fmi1_get_boolean_status_ft	46
6.8.2.43	fmi1_get_string_status_ft	46
6.8.2.44	fmi1_value_reference_enu_t	46
6.8.3	Enumeration Type Documentation	46
6.8.3.1	fmi1_status_t	46
6.8.3.2	fmi1_status_kind_t	46
6.8.3.3	fmi1_boolean_enu_t	47
6.8.3.4	fmi1_value_reference_enu_t	47
6.8.4	Function Documentation	47
6.8.4.1	fmi1_status_to_string	47
6.8.4.2	fmi1_get_platform	47
6.9	Enum types used with FMI 1.0 libs	48
6.9.1	Typedef Documentation	49
6.9.1.1	fmi1_variable_naming_convension_enu_t	49
6.9.1.2	fmi1_fmu_kind_enu_t	49
6.9.1.3	fmi1_variability_enu_t	49
6.9.1.4	fmi1_causality_enu_t	49
6.9.1.5	fmi1_variable_alias_kind_enu_t	49
6.9.1.6	fmi1_base_type_enu_t	50
6.9.2	Enumeration Type Documentation	50
6.9.2.1	fmi1_variable_naming_convension_enu_t	50
6.9.2.2	fmi1_fmu_kind_enu_t	50

6.9.2.3	fmi1_variability_enu_t	50
6.9.2.4	fmi1_causality_enu_t	51
6.9.2.5	fmi1_variable_alias_kind_enu_t	51
6.9.2.6	fmi1_base_type_enu_t	51
6.9.3	Function Documentation	51
6.9.3.1	fmi1_naming_convention_to_string	51
6.9.3.2	fmi1_fmu_kind_to_string	52
6.9.3.3	fmi1_variability_to_string	52
6.9.3.4	fmi1_causality_to_string	52
6.9.3.5	fmi1_base_type_to_string	52
6.10	Functions and types supporting FMI 2.0 processing.	53
6.10.1	Define Documentation	55
6.10.1.1	fmi2Component	55
6.10.1.2	fmi2ComponentEnvironment	56
6.10.1.3	fmi2FMUstate	56
6.10.1.4	fmi2ValueReference	56
6.10.1.5	fmi2Real	56
6.10.1.6	fmi2Integer	56
6.10.1.7	fmi2Boolean	56
6.10.1.8	fmi2Char	56
6.10.1.9	fmi2String	56
6.10.1.10	fmi2Byte	56
6.10.2	Typedef Documentation	56
6.10.2.1	fmi2_callback_logger_ft	56
6.10.2.2	fmi2_callback_allocate_memory_ft	57
6.10.2.3	fmi2_callback_free_memory_ft	57
6.10.2.4	fmi2_step_finished_ft	57
6.10.2.5	fmi2_get_types_platform_ft	57
6.10.2.6	fmi2_get_version_ft	57
6.10.2.7	fmi2_set_debug_logging_ft	57
6.10.2.8	fmi2_instantiate_ft	57
6.10.2.9	fmi2_free_instance_ft	57
6.10.2.10	fmi2_setup_experiment_ft	58
6.10.2.11	fmi2_enter_initialization_mode_ft	58

6.10.2.12 fmi2_exit_initialization_mode_ft	58
6.10.2.13 fmi2_terminate_ft	58
6.10.2.14 fmi2_reset_ft	58
6.10.2.15 fmi2_get_real_ft	58
6.10.2.16 fmi2_get_integer_ft	58
6.10.2.17 fmi2_get_boolean_ft	58
6.10.2.18 fmi2_get_string_ft	58
6.10.2.19 fmi2_set_real_ft	59
6.10.2.20 fmi2_set_integer_ft	59
6.10.2.21 fmi2_set_boolean_ft	59
6.10.2.22 fmi2_set_string_ft	59
6.10.2.23 fmi2_get_fmu_state_ft	59
6.10.2.24 fmi2_set_fmu_state_ft	59
6.10.2.25 fmi2_free_fmu_state_ft	59
6.10.2.26 fmi2_serialized_fmu_state_size_ft	59
6.10.2.27 fmi2_serialize_fmu_state_ft	60
6.10.2.28 fmi2_de_serialize_fmu_state_ft	60
6.10.2.29 fmi2_get_directional_derivative_ft	60
6.10.2.30 fmi2_enter_event_mode_ft	60
6.10.2.31 fmi2_new_discrete_states_ft	60
6.10.2.32 fmi2_enter_continuous_time_mode_ft	60
6.10.2.33 fmi2_completed_integrator_step_ft	60
6.10.2.34 fmi2_set_time_ft	60
6.10.2.35 fmi2_set_continuous_states_ft	60
6.10.2.36 fmi2_get_derivatives_ft	61
6.10.2.37 fmi2_get_event_indicators_ft	61
6.10.2.38 fmi2_get_continuous_states_ft	61
6.10.2.39 fmi2_get_nominals_of_continuous_states_ft	61
6.10.2.40 fmi2_set_real_input_derivatives_ft	61
6.10.2.41 fmi2_get_real_output_derivatives_ft	61
6.10.2.42 fmi2_do_step_ft	61
6.10.2.43 fmi2_cancel_step_ft	61
6.10.2.44 fmi2_get_status_ft	62
6.10.2.45 fmi2_get_real_status_ft	62

6.10.2.46	fmi2_get_integer_status_ft	62
6.10.2.47	fmi2_get_boolean_status_ft	62
6.10.2.48	fmi2_get_string_status_ft	62
6.10.3	Enumeration Type Documentation	62
6.10.3.1	fmi2_status_t	62
6.10.3.2	fmi2_type_t	63
6.10.3.3	fmi2_status_kind_t	63
6.10.3.4	fmi2_boolean_enu_t	63
6.10.4	Function Documentation	63
6.10.4.1	fmi2_status_to_string	63
6.10.4.2	fmi2_get_types_platform	63
6.11	Enum types used with FMI 2.0 libs	64
6.11.1	Define Documentation	66
6.11.1.1	FMI2_ME_CAPABILITIES	66
6.11.1.2	FMI2_CS_CAPABILITIES	66
6.11.1.3	FMI2_SI_BASE_UNITS	67
6.11.2	Typedef Documentation	67
6.11.2.1	fmi2_variable_naming_convension_enu_t	67
6.11.2.2	fmi2_fmu_kind_enu_t	67
6.11.2.3	fmi2_variability_enu_t	67
6.11.2.4	fmi2_causality_enu_t	67
6.11.2.5	fmi2_initial_enu_t	67
6.11.2.6	fmi2_variable_alias_kind_enu_t	68
6.11.2.7	fmi2_base_type_enu_t	68
6.11.2.8	fmi2_capabilities_enu_t	68
6.11.2.9	fmi2_SI_base_units_enu_t	68
6.11.2.10	fmi2_dependency_factor_kind_enu_t	68
6.11.3	Enumeration Type Documentation	68
6.11.3.1	fmi2_variable_naming_convension_enu_t	68
6.11.3.2	fmi2_fmu_kind_enu_t	68
6.11.3.3	fmi2_variability_enu_t	69
6.11.3.4	fmi2_causality_enu_t	69
6.11.3.5	fmi2_initial_enu_t	69
6.11.3.6	fmi2_variable_alias_kind_enu_t	70

6.11.3.7	fmi2_base_type_enumeration_t	70
6.11.3.8	fmi2_capabilities_enumeration_t	70
6.11.3.9	fmi2_SI_base_units_enumeration_t	70
6.11.3.10	fmi2_dependency_factor_kind_enumeration_t	70
6.11.4	Function Documentation	71
6.11.4.1	fmi2_naming_convention_to_string	71
6.11.4.2	fmi2_fmu_kind_to_string	71
6.11.4.3	fmi2_variability_to_string	71
6.11.4.4	fmi2_causality_to_string	71
6.11.4.5	fmi2_initial_to_string	71
6.11.4.6	fmi2_get_default_initial	71
6.11.4.7	fmi2_get_valid_initial	72
6.11.4.8	fmi2_base_type_to_string	72
6.11.4.9	fmi2_capability_to_string	72
6.11.4.10	fmi2_SI_base_unit_to_string	72
6.11.4.11	fmi2_SI_base_unit_exp_to_string	73
6.11.4.12	fmi2_dependency_factor_kind_to_string	73
6.12	Definition of XML callbacks struct	74
6.12.1	Typedef Documentation	74
6.12.1.1	fmi2_xml_callbacks_t	74
6.12.1.2	fmi2_xml_element_start_handle_ft	75
6.12.1.3	fmi2_xml_element_data_handle_ft	75
6.12.1.4	fmi2_xml_element_end_handle_ft	75
6.12.2	Variable Documentation	75
6.12.2.1	startHandle	76
6.12.2.2	dataHandle	76
6.12.2.3	endHandle	76
6.12.2.4	context	76
6.13	Definition of callbacks struct and supporting functions	77
6.13.1	Define Documentation	79
6.13.1.1	JM_MAX_ERROR_MESSAGE_SIZE	79
6.13.2	Typedef Documentation	79
6.13.2.1	jm_callbacks	79
6.13.2.2	jm_malloc_f	79

6.13.2.3	jm_realloc_f	79
6.13.2.4	jm_calloc_f	79
6.13.2.5	jm_free_f	79
6.13.2.6	jm_logger_f	79
6.13.3	Function Documentation	80
6.13.3.1	jm_get_last_error	80
6.13.3.2	jm_clear_last_error	80
6.13.3.3	jm_set_default_callbacks	80
6.13.3.4	jm_get_default_callbacks	80
6.13.3.5	jm_default_logger	80
6.13.3.6	jm_log	81
6.13.3.7	jm_log_v	81
6.13.3.8	jm_log_fatal_v	81
6.13.3.9	jm_log_fatal	81
6.13.3.10	jm_log_error_v	81
6.13.3.11	jm_log_error	82
6.13.3.12	jm_log_warning_v	82
6.13.3.13	jm_log_warning	82
6.13.3.14	jm_log_info_v	82
6.13.3.15	jm_log_info	82
6.13.3.16	jm_log_verbose_v	82
6.13.3.17	jm_log_verbose	82
6.13.3.18	jm_log_debug_v	82
6.13.3.19	jm_log_debug	83
6.14	Named objects	84
6.14.1	Typedef Documentation	84
6.14.1.1	jm_named_ptr	84
6.14.2	Function Documentation	85
6.14.2.1	jm_named_alloc	85
6.14.2.2	jm_named_alloc_v	85
6.14.2.3	jm_named_free	85
6.14.2.4	jm_vector_declare_template	85
6.14.2.5	jm_named_vector_free_data	85
6.14.2.6	jm_named_vector_free	86

6.15 Handling platform specific defines and functions	87
6.15.1 Typedef Documentation	88
6.15.1.1 jm_dll_function_ptr	88
6.15.2 Function Documentation	88
6.15.2.1 jm_portability_load_dll_handle	88
6.15.2.2 jm_portability_free_dll_handle	88
6.15.2.3 jm_portability_load_dll_function	88
6.15.2.4 jm_portability_get_last_dll_error	88
6.15.2.5 jm_portability_get_current_working_directory	88
6.15.2.6 jm_portability_set_current_working_directory	88
6.15.2.7 jm_get_system_temp_dir	88
6.15.2.8 jm_mktemp	89
6.15.2.9 jm_get_dir_abspath	89
6.15.2.10 jm_mk_temp_dir	89
6.15.2.11 jm_create_URL_from_abs_path	90
6.15.2.12 jm_mkdir	90
6.15.2.13 jm_rmdir	90
6.15.2.14 jm_vsnprintf	90
6.15.2.15 jm_snprintf	90
6.16 A set of strings	91
6.16.1 Typedef Documentation	91
6.16.1.1 jm_string_set	91
6.16.2 Function Documentation	91
6.16.2.1 jm_string_set_find	91
6.16.2.2 jm_string_set_put	91
6.17 A vector of items (dynamic array)	93
6.17.1 Define Documentation	95
6.17.1.1 jm_mangle_ex	95
6.17.1.2 jm_mangle	95
6.17.1.3 jm_vector	95
6.17.1.4 jm_vector_alloc	95
6.17.1.5 jm_vector_free	95
6.17.1.6 jm_vector_init	95
6.17.1.7 jm_vector_free_data	95

6.17.1.8	jm_vector_get_size	96
6.17.1.9	jm_vector_get_item	96
6.17.1.10	jm_vector_get_itemp	96
6.17.1.11	jm_vector_get_last	96
6.17.1.12	jm_vector_get_lastp	96
6.17.1.13	jm_define_comp_f	96
6.17.1.14	jm_diff	97
6.17.1.15	jm_vector_find	97
6.17.1.16	jm_vector_find_index	97
6.17.1.17	jm_vector_qsort	97
6.17.1.18	jm_vector_bsearch	97
6.17.1.19	jm_vector_bsearch_index	98
6.17.1.20	jm_vector_set_item	98
6.17.1.21	jm_vector_zero	98
6.17.1.22	jm_vector_resize	98
6.17.1.23	jm_vector_reserve	98
6.17.1.24	jm_vector_copy	98
6.17.1.25	jm_vector_clone	99
6.17.1.26	jm_vector_append	99
6.17.1.27	jm_vector_insert	99
6.17.1.28	jm_vector_remove_item	99
6.17.1.29	jm_vector_resize1	99
6.17.1.30	jm_vector_push_back	99
6.17.1.31	jm_vector_foreach	100
6.17.1.32	jm_vector_foreach_c	100
6.17.1.33	JM_VECTOR_MINIMAL_CAPACITY	100
6.17.1.34	JM_VECTOR_MAX_MEMORY_CHUNK	100
6.17.1.35	jm_vector_declare_template	100
6.17.2	Typedef Documentation	100
6.17.2.1	jm_compare_ft	100
6.17.3	Function Documentation	100
6.17.3.1	jm_vector_declare_template	100
6.17.3.2	jm_vector_declare_template	101
6.17.3.3	jm_define_comp_f	101

6.17.3.4	jm_define_comp_f	101
6.17.3.5	jm_define_comp_f	101
6.17.3.6	jm_vector	101
6.17.4	Variable Documentation	101
6.17.4.1	char	101
6.17.4.2	size_t	101
6.17.4.3	jm_name_ID_map_t	101
6.17.4.4	jm_diff_name	101
6.17.4.5	capacity	101
6.18	Library initialization	102
6.18.1	Detailed Description	102
6.18.2	Typedef Documentation	103
6.18.2.1	fmi_import_context_t	103
6.18.2.2	fmi1_import_t	103
6.18.2.3	fmi2_import_t	103
6.18.3	Function Documentation	103
6.18.3.1	fmi_import_allocate_context	103
6.18.3.2	fmi_import_free_context	103
6.18.3.3	fmi_import_get_fmi_version	104
6.18.3.4	fmi1_import_parse_xml	104
6.18.3.5	fmi2_import_parse_xml	104
6.19	Utility functions supporting interactions with the library	105
6.19.1	Function Documentation	105
6.19.1.1	fmi_import_mk_temp_dir	105
6.19.1.2	fmi_import_rmdir	105
6.19.1.3	fmi_import_create_URL_from_abs_path	106
6.19.1.4	fmi_import_get_dll_path	106
6.19.1.5	fmi_import_get_model_description_path	106
6.20	Constuction, destruction and error handling	107
6.20.1	Function Documentation	107
6.20.1.1	fmi1_import_parse_xml	107
6.20.1.2	fmi1_import_get_last_error	107
6.20.1.3	fmi1_import_clear_last_error	108
6.20.1.4	fmi1_import_free	108

6.21	General information retrieval	109
6.21.1	Detailed Description	110
6.21.2	Function Documentation	111
6.21.2.1	fmi1_import_get_model_name	111
6.21.2.2	fmi1_import_get_model_identifier	111
6.21.2.3	fmi1_import_get_GUID	111
6.21.2.4	fmi1_import_get_description	111
6.21.2.5	fmi1_import_get_author	111
6.21.2.6	fmi1_import_get_model_version	112
6.21.2.7	fmi1_import_get_model_standard_version	112
6.21.2.8	fmi1_import_get_generation_tool	112
6.21.2.9	fmi1_import_get_generation_date_and_time	112
6.21.2.10	fmi1_import_get_naming_convention	113
6.21.2.11	fmi1_import_get_number_of_continuous_states	113
6.21.2.12	fmi1_import_get_number_of_event_indicators	113
6.21.2.13	fmi1_import_get_default_experiment_start	113
6.21.2.14	fmi1_import_get_default_experiment_stop	113
6.21.2.15	fmi1_import_get_default_experiment_tolerance	113
6.21.2.16	fmi1_import_get_fmu_kind	113
6.21.2.17	fmi1_import_get_capabilities	114
6.21.2.18	fmi1_import_get_type_definitions	114
6.21.2.19	fmi1_import_get_unit_definitions	114
6.21.2.20	fmi1_import_get_direct_dependency	114
6.21.2.21	fmi1_import_get_variable_alias_base	114
6.21.2.22	fmi1_import_get_variable_aliases	114
6.21.2.23	fmi1_import_get_variable_list	115
6.21.2.24	fmi1_import_get_variable_list_alphabetical_order	115
6.21.2.25	fmi1_import_create_var_list	115
6.22	Interface to the standard FMI 1.0 "C" API	116
6.22.1	Detailed Description	116
6.23	Functions to retrieve capability flags.	117
6.23.1	Detailed Description	117
6.23.2	Typedef Documentation	118
6.23.2.1	fmi1_import_capabilities_t	118

6.23.3	Function Documentation	118
6.23.3.1	fmi1_import_get_canHandleVariableCommunication- StepSize	118
6.23.3.2	fmi1_import_get_canHandleEvents	118
6.23.3.3	fmi1_import_get_canRejectSteps	118
6.23.3.4	fmi1_import_get_canInterpolateInputs	118
6.23.3.5	fmi1_import_get_maxOutputDerivativeOrder	118
6.23.3.6	fmi1_import_get_canRunAsynchronuously	118
6.23.3.7	fmi1_import_get_canSignalEvents	118
6.23.3.8	fmi1_import_get_canBeInstantiatedOnlyOncePer- Process	119
6.23.3.9	fmi1_import_get_canNotUseMemoryManagement- Functions	119
6.24	FMI 1.0 Constructor and Destructor	120
6.24.1	Detailed Description	120
6.24.2	Function Documentation	120
6.24.2.1	fmi1_import_create_dllfmu	120
6.24.2.2	fmi1_import_destroy_dllfmu	121
6.24.2.3	fmi1_import_set_debug_mode	121
6.25	FMI 1.0 (ME) Model Exchange functions	122
6.25.1	Detailed Description	123
6.25.2	Function Documentation	123
6.25.2.1	fmi1_import_get_model_types_platform	123
6.25.2.2	fmi1_import_instantiate_model	123
6.25.2.3	fmi1_import_free_model_instance	123
6.25.2.4	fmi1_import_set_time	124
6.25.2.5	fmi1_import_set_continuous_states	124
6.25.2.6	fmi1_import_completed_integrator_step	124
6.25.2.7	fmi1_import_initialize	125
6.25.2.8	fmi1_import_get_derivatives	125
6.25.2.9	fmi1_import_get_event_indicators	125
6.25.2.10	fmi1_import_eventUpdate	126
6.25.2.11	fmi1_import_get_continuous_states	126
6.25.2.12	fmi1_import_get_nominal_continuous_states	127
6.25.2.13	fmi1_import_get_state_value_references	127

6.25.2.14 fmi1_import_terminate	127
6.26 FMI 1.0 (CS) Co-Simulation functions	129
6.26.1 Detailed Description	130
6.26.2 Function Documentation	130
6.26.2.1 fmi1_import_get_types_platform	130
6.26.2.2 fmi1_import_instantiate_slave	130
6.26.2.3 fmi1_import_initialize_slave	131
6.26.2.4 fmi1_import_terminate_slave	131
6.26.2.5 fmi1_import_reset_slave	132
6.26.2.6 fmi1_import_free_slave_instance	132
6.26.2.7 fmi1_import_set_real_input_derivatives	132
6.26.2.8 fmi1_import_get_real_output_derivatives	133
6.26.2.9 fmi1_import_cancel_step	133
6.26.2.10 fmi1_import_do_step	133
6.26.2.11 fmi1_import_get_status	134
6.26.2.12 fmi1_import_get_real_status	134
6.26.2.13 fmi1_import_get_integer_status	135
6.26.2.14 fmi1_import_get_boolean_status	135
6.26.2.15 fmi1_import_get_string_status	135
6.27 FMI 1.0 (ME & CS) Common functions	137
6.27.1 Detailed Description	137
6.27.2 Function Documentation	138
6.27.2.1 fmi1_import_get_version	138
6.27.2.2 fmi1_import_set_debug_logging	138
6.27.2.3 fmi1_import_set_real	138
6.27.2.4 fmi1_import_set_integer	139
6.27.2.5 fmi1_import_set_boolean	139
6.27.2.6 fmi1_import_set_string	139
6.27.2.7 fmi1_import_get_real	140
6.27.2.8 fmi1_import_get_integer	140
6.27.2.9 fmi1_import_get_boolean	141
6.27.2.10 fmi1_import_get_string	141
6.28 Convenience functions.	142
6.28.1 Detailed Description	142

6.28.2	Function Documentation	143
6.28.2.1	fmi1_import_collect_model_counts	143
6.28.2.2	fmi1_import_expand_variable_references	143
6.28.2.3	fmi1_log_forwarding	143
6.28.2.4	fmi1_log_forwarding_v	143
6.28.2.5	fmi1_default_callback_logger	144
6.28.2.6	fmi1_import_init_logger	144
6.29	Functions to retrieve co-simulation related information.	145
6.29.1	Function Documentation	145
6.29.1.1	fmi1_import_get_entry_point	145
6.29.1.2	fmi1_import_get_mime_type	145
6.29.1.3	fmi1_import_get_manual_start	145
6.29.1.4	fmi1_import_get_number_of_additional_models	145
6.29.1.5	fmi1_import_get_additional_model_name	145
6.30	Support for processing variable types	147
6.30.1	Typedef Documentation	149
6.30.1.1	fmi1_import_real_typedef_t	149
6.30.1.2	fmi1_import_integer_typedef_t	149
6.30.1.3	fmi1_import_enumeration_typedef_t	149
6.30.1.4	fmi1_import_variable_typedef_t	149
6.30.1.5	fmi1_import_type_definitions_t	149
6.30.2	Function Documentation	149
6.30.2.1	fmi1_import_get_type_definition_number	149
6.30.2.2	fmi1_import_get_typedef	149
6.30.2.3	fmi1_import_get_type_display_unit	150
6.30.2.4	fmi1_import_get_type_name	150
6.30.2.5	fmi1_import_get_type_description	150
6.30.2.6	fmi1_import_get_base_type	150
6.30.2.7	fmi1_import_get_type_as_real	150
6.30.2.8	fmi1_import_get_type_as_int	151
6.30.2.9	fmi1_import_get_type_as_enum	151
6.30.2.10	fmi1_import_get_type_quantity	151
6.30.2.11	fmi1_import_get_real_type_min	151
6.30.2.12	fmi1_import_get_real_type_max	151

6.30.2.13	fmi1_import_get_real_type_nominal	152
6.30.2.14	fmi1_import_get_real_type_unit	152
6.30.2.15	fmi1_import_get_real_type_is_relative_quantity	152
6.30.2.16	fmi1_import_get_integer_type_min	152
6.30.2.17	fmi1_import_get_integer_type_max	152
6.30.2.18	fmi1_import_get_enum_type_min	152
6.30.2.19	fmi1_import_get_enum_type_max	153
6.30.2.20	fmi1_import_get_enum_type_size	153
6.30.2.21	fmi1_import_get_enum_type_item_name	153
6.30.2.22	fmi1_import_get_enum_type_item_description	153
6.31	Functions for handling unit definitions.	154
6.31.1	Typedef Documentation	155
6.31.1.1	fmi1_import_unit_t	155
6.31.1.2	fmi1_import_display_unit_t	155
6.31.1.3	fmi1_import_unit_definitions_t	155
6.31.2	Function Documentation	155
6.31.2.1	fmi1_import_get_unit_definitions_number	155
6.31.2.2	fmi1_import_get_unit	155
6.31.2.3	fmi1_import_get_unit_name	155
6.31.2.4	fmi1_import_get_unit_display_unit_number	155
6.31.2.5	fmi1_import_get_unit_display_unit	156
6.31.2.6	fmi1_import_get_base_unit	156
6.31.2.7	fmi1_import_get_display_unit_name	156
6.31.2.8	fmi1_import_get_display_unit_gain	156
6.31.2.9	fmi1_import_get_display_unit_offset	156
6.31.2.10	fmi1_import_convert_to_display_unit	156
6.31.2.11	fmi1_import_convert_from_display_unit	157
6.32	Functions for handling variable definitions.	158
6.32.1	Detailed Description	160
6.32.2	Typedef Documentation	160
6.32.2.1	fmi1_import_variable_t	160
6.32.2.2	fmi1_import_real_variable_t	161
6.32.2.3	fmi1_import_integer_variable_t	161
6.32.2.4	fmi1_import_string_variable_t	161

6.32.2.5	fmi1_import_enum_variable_t	161
6.32.2.6	fmi1_import_bool_variable_t	161
6.32.2.7	fmi1_import_variable_list_t	161
6.32.3	Function Documentation	161
6.32.3.1	fmi1_import_get_variable_by_name	161
6.32.3.2	fmi1_import_get_variable_by_vr	162
6.32.3.3	fmi1_import_get_variable_name	162
6.32.3.4	fmi1_import_get_variable_description	162
6.32.3.5	fmi1_import_get_variable_vr	162
6.32.3.6	fmi1_import_get_variable_declared_type	163
6.32.3.7	fmi1_import_get_variable_base_type	163
6.32.3.8	fmi1_import_get_variable_has_start	163
6.32.3.9	fmi1_import_get_variable_is_fixed	163
6.32.3.10	fmi1_import_get_variability	163
6.32.3.11	fmi1_import_get_causality	163
6.32.3.12	fmi1_import_get_variable_as_real	163
6.32.3.13	fmi1_import_get_variable_as_integer	164
6.32.3.14	fmi1_import_get_variable_as_enum	164
6.32.3.15	fmi1_import_get_variable_as_string	164
6.32.3.16	fmi1_import_get_variable_as_boolean	164
6.32.3.17	fmi1_import_get_real_variable_start	164
6.32.3.18	fmi1_import_get_real_variable_max	165
6.32.3.19	fmi1_import_get_real_variable_min	165
6.32.3.20	fmi1_import_get_real_variable_nominal	165
6.32.3.21	fmi1_import_get_real_variable_unit	165
6.32.3.22	fmi1_import_get_real_variable_display_unit	165
6.32.3.23	fmi1_import_get_string_variable_start	165
6.32.3.24	fmi1_import_get_boolean_variable_start	165
6.32.3.25	fmi1_import_get_integer_variable_start	166
6.32.3.26	fmi1_import_get_integer_variable_min	166
6.32.3.27	fmi1_import_get_integer_variable_max	166
6.32.3.28	fmi1_import_get_enum_variable_start	166
6.32.3.29	fmi1_import_get_enum_variable_min	166
6.32.3.30	fmi1_import_get_enum_variable_max	166

6.32.3.31 fmi1_import_get_variable_alias_kind	166
6.32.3.32 fmi1_import_get_variable_original_order	166
6.33 Basic support for vendor annotations.	167
6.33.1 Typedef Documentation	167
6.33.1.1 fmi1_import_vendor_list_t	167
6.33.1.2 fmi1_import_vendor_t	168
6.33.1.3 fmi1_import_annotation_t	168
6.33.2 Function Documentation	168
6.33.2.1 fmi1_import_get_number_of_vendors	168
6.33.2.2 fmi1_import_get_vendor	168
6.33.2.3 fmi1_import_get_vendor_name	168
6.33.2.4 fmi1_import_get_number_of_vendor_annotations	168
6.33.2.5 fmi1_import_get_vendor_annotation	168
6.33.2.6 fmi1_import_get_annotation_name	169
6.33.2.7 fmi1_import_get_annotation_value	169
6.33.2.8 fmi1_import_get_vendor_list	169
6.34 Constuction, destruction and error handling	170
6.34.1 Function Documentation	170
6.34.1.1 fmi2_import_get_last_error	170
6.34.1.2 fmi2_import_clear_last_error	170
6.34.1.3 fmi2_import_free	171
6.35 General information retrieval	172
6.35.1 Detailed Description	175
6.35.2 Function Documentation	175
6.35.2.1 fmi2_import_get_model_name	175
6.35.2.2 fmi2_import_get_capability	175
6.35.2.3 fmi2_import_get_model_identifier_ME	175
6.35.2.4 fmi2_import_get_model_identifier_CS	176
6.35.2.5 fmi2_import_get_GUID	176
6.35.2.6 fmi2_import_get_description	176
6.35.2.7 fmi2_import_get_author	176
6.35.2.8 fmi2_import_get_copyright	176
6.35.2.9 fmi2_import_get_license	177
6.35.2.10 fmi2_import_get_model_version	177

6.35.2.11 fmi2_import_get_model_standard_version	177
6.35.2.12 fmi2_import_get_generation_tool	177
6.35.2.13 fmi2_import_get_generation_date_and_time	177
6.35.2.14 fmi2_import_get_naming_convention	178
6.35.2.15 fmi2_import_get_number_of_continuous_states	178
6.35.2.16 fmi2_import_get_number_of_event_indicators	178
6.35.2.17 fmi2_import_get_default_experiment_start	178
6.35.2.18 fmi2_import_get_default_experiment_stop	178
6.35.2.19 fmi2_import_get_default_experiment_tolerance	178
6.35.2.20 fmi2_import_get_default_experiment_step	178
6.35.2.21 fmi2_import_get_fmu_kind	178
6.35.2.22 fmi2_import_get_type_definitions	179
6.35.2.23 fmi2_import_get_unit_definitions	179
6.35.2.24 fmi2_import_get_variable_alias_base	179
6.35.2.25 fmi2_import_get_variable_aliases	179
6.35.2.26 fmi2_import_get_variable_list	179
6.35.2.27 fmi2_import_create_var_list	180
6.35.2.28 fmi2_import_get_vendors_num	180
6.35.2.29 fmi2_import_get_vendor_name	180
6.35.2.30 fmi2_import_get_log_categories_num	180
6.35.2.31 fmi2_import_get_log_category	180
6.35.2.32 fmi2_import_get_log_category_description	180
6.35.2.33 fmi2_import_get_source_files_me_num	181
6.35.2.34 fmi2_import_get_source_file_me	181
6.35.2.35 fmi2_import_get_source_files_cs_num	181
6.35.2.36 fmi2_import_get_source_file_cs	181
6.35.2.37 fmi2_import_get_variable_by_name	181
6.35.2.38 fmi2_import_get_variable_by_vr	181
6.35.2.39 fmi2_import_get_outputs_list	182
6.35.2.40 fmi2_import_get_derivatives_list	182
6.35.2.41 fmi2_import_get_discrete_states_list	182
6.35.2.42 fmi2_import_get_initial_unknowns_list	183
6.35.2.43 fmi2_import_get_outputs_dependencies	183
6.35.2.44 fmi2_import_get_derivatives_dependencies	183

6.35.2.45	fmi2_import_get_discrete_states_dependencies	184
6.35.2.46	fmi2_import_get_initial_unknowns_dependencies	184
6.36	Interface to the standard FMI 2.0 "C" API	186
6.36.1	Detailed Description	186
6.37	FMI 2.0 Constructor and Destructor	187
6.37.1	Detailed Description	187
6.37.2	Function Documentation	187
6.37.2.1	fmi2_import_create_dllfmu	187
6.37.2.2	fmi2_import_destroy_dllfmu	188
6.37.2.3	fmi2_import_set_debug_mode	188
6.38	FMI 2.0 (ME) Model Exchange functions	189
6.38.1	Detailed Description	189
6.38.2	Function Documentation	190
6.38.2.1	fmi2_import_enter_event_mode	190
6.38.2.2	fmi2_import_new_discrete_states	190
6.38.2.3	fmi2_import_enter_continuous_time_mode	190
6.38.2.4	fmi2_import_set_time	191
6.38.2.5	fmi2_import_set_continuous_states	191
6.38.2.6	fmi2_import_completed_integrator_step	191
6.38.2.7	fmi2_import_get_derivatives	192
6.38.2.8	fmi2_import_get_event_indicators	192
6.38.2.9	fmi2_import_get_continuous_states	192
6.38.2.10	fmi2_import_get_nominals_of_continuous_states	193
6.39	FMI 2.0 (CS) Co-Simulation functions	194
6.39.1	Detailed Description	194
6.39.2	Function Documentation	195
6.39.2.1	fmi2_import_set_real_input_derivatives	195
6.39.2.2	fmi2_import_get_real_output_derivatives	195
6.39.2.3	fmi2_import_cancel_step	195
6.39.2.4	fmi2_import_do_step	196
6.39.2.5	fmi2_import_get_status	196
6.39.2.6	fmi2_import_get_real_status	197
6.39.2.7	fmi2_import_get_integer_status	197
6.39.2.8	fmi2_import_get_boolean_status	197

6.39.2.9	fmi2_import_get_string_status	198
6.40	FMI 2.0 (ME & CS) Common functions	199
6.40.1	Detailed Description	200
6.40.2	Function Documentation	201
6.40.2.1	fmi2_import_get_version	201
6.40.2.2	fmi2_import_set_debug_logging	201
6.40.2.3	fmi2_import_instantiate	201
6.40.2.4	fmi2_import_free_instance	202
6.40.2.5	fmi2_import_setup_experiment	202
6.40.2.6	fmi2_import_enter_initialization_mode	203
6.40.2.7	fmi2_import_exit_initialization_mode	203
6.40.2.8	fmi2_import_terminate	203
6.40.2.9	fmi2_import_reset	203
6.40.2.10	fmi2_import_set_real	204
6.40.2.11	fmi2_import_set_integer	204
6.40.2.12	fmi2_import_set_boolean	205
6.40.2.13	fmi2_import_set_string	205
6.40.2.14	fmi2_import_get_real	205
6.40.2.15	fmi2_import_get_integer	206
6.40.2.16	fmi2_import_get_boolean	206
6.40.2.17	fmi2_import_get_string	207
6.40.2.18	fmi2_import_get_types_platform	207
6.40.2.19	fmi2_import_get_fmu_state	207
6.40.2.20	fmi2_import_set_fmu_state	208
6.40.2.21	fmi2_import_free_fmu_state	208
6.40.2.22	fmi2_import_serialized_fmu_state_size	208
6.40.2.23	fmi2_import_serialize_fmu_state	209
6.40.2.24	fmi2_import_de_serialize_fmu_state	209
6.40.2.25	fmi2_import_get_directional_derivative	209
6.41	Convenience functions.	211
6.41.1	Detailed Description	211
6.41.2	Function Documentation	212
6.41.2.1	fmi2_import_collect_model_counts	212
6.41.2.2	fmi2_import_expand_variable_references	212

6.41.2.3	fmi2_log_forwarding	212
6.41.2.4	fmi2_log_forwarding_v	212
6.41.2.5	fmi2_default_callback_logger	213
6.41.2.6	fmi2_import_init_logger	213
6.42	Support for processing variable types	214
6.42.1	Typedef Documentation	216
6.42.1.1	fmi2_import_real_typedef_t	216
6.42.1.2	fmi2_import_integer_typedef_t	216
6.42.1.3	fmi2_import_enumeration_typedef_t	216
6.42.1.4	fmi2_import_variable_typedef_t	216
6.42.1.5	fmi2_import_type_definitions_t	216
6.42.2	Function Documentation	216
6.42.2.1	fmi2_import_get_type_definition_number	216
6.42.2.2	fmi2_import_get_typedef	217
6.42.2.3	fmi2_import_get_type_display_unit	217
6.42.2.4	fmi2_import_get_type_name	217
6.42.2.5	fmi2_import_get_type_description	217
6.42.2.6	fmi2_import_get_base_type	217
6.42.2.7	fmi2_import_get_type_as_real	218
6.42.2.8	fmi2_import_get_type_as_int	218
6.42.2.9	fmi2_import_get_type_as_enum	218
6.42.2.10	fmi2_import_get_type_quantity	218
6.42.2.11	fmi2_import_get_real_type_min	218
6.42.2.12	fmi2_import_get_real_type_max	219
6.42.2.13	fmi2_import_get_real_type_nominal	219
6.42.2.14	fmi2_import_get_real_type_unit	219
6.42.2.15	fmi2_import_get_real_type_is_relative_quantity	219
6.42.2.16	fmi2_import_get_real_type_is_unbounded	219
6.42.2.17	fmi2_import_get_integer_type_min	219
6.42.2.18	fmi2_import_get_integer_type_max	219
6.42.2.19	fmi2_import_get_enum_type_min	220
6.42.2.20	fmi2_import_get_enum_type_max	220
6.42.2.21	fmi2_import_get_enum_type_size	220
6.42.2.22	fmi2_import_get_enum_type_item_name	220

6.42.2.23 fmi2_import_get_enum_type_item_value	220
6.42.2.24 fmi2_import_get_enum_type_item_description	220
6.42.2.25 fmi2_import_get_enum_type_value_name	220
6.43 Functions for handling unit definitions.	221
6.43.1 Typedef Documentation	222
6.43.1.1 fmi2_import_unit_t	222
6.43.1.2 fmi2_import_display_unit_t	222
6.43.1.3 fmi2_import_unit_definitions_t	222
6.43.2 Function Documentation	222
6.43.2.1 fmi2_import_get_unit_definitions_number	222
6.43.2.2 fmi2_import_get_unit	223
6.43.2.3 fmi2_import_get_unit_name	223
6.43.2.4 fmi2_import_get_unit_display_unit_number	223
6.43.2.5 fmi2_import_get_SI_unit_exponents	223
6.43.2.6 fmi2_import_get_SI_unit_factor	223
6.43.2.7 fmi2_import_get_SI_unit_offset	223
6.43.2.8 fmi2_import_convert_to_SI_base_unit	223
6.43.2.9 fmi2_import_convert_from_SI_base_unit	223
6.43.2.10 fmi2_import_get_unit_display_unit	224
6.43.2.11 fmi2_import_get_base_unit	224
6.43.2.12 fmi2_import_get_display_unit_name	224
6.43.2.13 fmi2_import_get_display_unit_factor	224
6.43.2.14 fmi2_import_get_display_unit_offset	224
6.43.2.15 fmi2_import_convert_to_display_unit	224
6.43.2.16 fmi2_import_convert_from_display_unit	225
6.44 Functions for handling variable definitions.	226
6.44.1 Detailed Description	228
6.44.2 Typedef Documentation	229
6.44.2.1 fmi2_import_variable_t	229
6.44.2.2 fmi2_import_real_variable_t	229
6.44.2.3 fmi2_import_integer_variable_t	229
6.44.2.4 fmi2_import_string_variable_t	229
6.44.2.5 fmi2_import_enum_variable_t	229
6.44.2.6 fmi2_import_bool_variable_t	229

6.44.2.7	fmi2_import_variable_list_t	229
6.44.3	Function Documentation	230
6.44.3.1	fmi2_import_get_variable_name	230
6.44.3.2	fmi2_import_get_variable_description	230
6.44.3.3	fmi2_import_get_variable_vr	230
6.44.3.4	fmi2_import_get_variable_declared_type	230
6.44.3.5	fmi2_import_get_variable_base_type	230
6.44.3.6	fmi2_import_get_variable_has_start	230
6.44.3.7	fmi2_import_get_variability	230
6.44.3.8	fmi2_import_get_causality	231
6.44.3.9	fmi2_import_get_initial	231
6.44.3.10	fmi2_import_get_previous	231
6.44.3.11	fmi2_import_get_canHandleMultipleSetPerTimeInstant	231
6.44.3.12	fmi2_import_get_variable_as_real	231
6.44.3.13	fmi2_import_get_variable_as_integer	231
6.44.3.14	fmi2_import_get_variable_as_enum	232
6.44.3.15	fmi2_import_get_variable_as_string	232
6.44.3.16	fmi2_import_get_variable_as_boolean	232
6.44.3.17	fmi2_import_get_real_variable_start	232
6.44.3.18	fmi2_import_get_real_variable_derivative_of	232
6.44.3.19	fmi2_import_get_real_variable_reinit	233
6.44.3.20	fmi2_import_get_real_variable_max	233
6.44.3.21	fmi2_import_get_real_variable_min	233
6.44.3.22	fmi2_import_get_real_variable_nominal	233
6.44.3.23	fmi2_import_get_real_variable_unit	233
6.44.3.24	fmi2_import_get_real_variable_display_unit	234
6.44.3.25	fmi2_import_get_string_variable_start	234
6.44.3.26	fmi2_import_get_boolean_variable_start	234
6.44.3.27	fmi2_import_get_integer_variable_start	234
6.44.3.28	fmi2_import_get_integer_variable_min	234
6.44.3.29	fmi2_import_get_integer_variable_max	234
6.44.3.30	fmi2_import_get_enum_variable_start	234
6.44.3.31	fmi2_import_get_enum_variable_min	234
6.44.3.32	fmi2_import_get_enum_variable_max	235

6.44.3.33 fmi2_import_get_variable_alias_kind	235
6.44.3.34 fmi2_import_get_variable_original_order	235
7 Directory Documentation	237
7.1 build/ Directory Reference	237
7.2 build/doc/ Directory Reference	237
7.3 src/Util/include/FMI/ Directory Reference	237
7.4 src/Import/include/FMI/ Directory Reference	238
7.5 Test/FMI1/ Directory Reference	238
7.6 src/Util/include/FMI1/ Directory Reference	238
7.7 src/Import/include/FMI1/ Directory Reference	239
7.8 Test/FMI2/ Directory Reference	239
7.9 src/Util/include/FMI2/ Directory Reference	239
7.10 src/Import/include/FMI2/ Directory Reference	240
7.11 Test/FMI2/fmu_dummy/ Directory Reference	240
7.12 Test/FMI1/fmu_dummy/ Directory Reference	240
7.13 src/Import/ Directory Reference	240
7.14 src/Util/include/ Directory Reference	241
7.15 src/Import/include/ Directory Reference	241
7.16 src/Util/include/JM/ Directory Reference	241
7.17 src/ Directory Reference	241
7.18 Test/ Directory Reference	242
7.19 src/Util/ Directory Reference	242
8 Data Structure Documentation	243
8.1 component_t Struct Reference	243
8.1.1 Detailed Description	244
8.1.2 Field Documentation	244
8.1.2.1 states	244
8.1.2.2 states_nom	244
8.1.2.3 states_vr	244
8.1.2.4 states_der	245
8.1.2.5 event_indicators	245
8.1.2.6 reals	245
8.1.2.7 integers	245

8.1.2.8	booleans	245
8.1.2.9	strings	245
8.1.2.10	loggingOn	245
8.1.2.11	instanceName	245
8.1.2.12	GUID	245
8.1.2.13	functions	245
8.1.2.14	fmtime	246
8.1.2.15	callEventUpdate	246
8.1.2.16	toleranceControlled	246
8.1.2.17	relativeTolerance	246
8.1.2.18	eventInfo	246
8.1.2.19	states_prev	246
8.1.2.20	fmuLocation	246
8.1.2.21	contentType	246
8.1.2.22	timeout	246
8.1.2.23	visible	246
8.1.2.24	interactive	247
8.1.2.25	tStart	247
8.1.2.26	StopTimeDefined	247
8.1.2.27	tStop	247
8.1.2.28	input_real	247
8.1.2.29	output_real	247
8.1.2.30	states	247
8.1.2.31	states_nom	247
8.1.2.32	states_der	247
8.1.2.33	event_indicators	247
8.1.2.34	reals	248
8.1.2.35	integers	248
8.1.2.36	booleans	248
8.1.2.37	strings	248
8.1.2.38	loggingOn	248
8.1.2.39	functions	248
8.1.2.40	fmtime	248
8.1.2.41	toleranceControlled	248

8.1.2.42	relativeTolerance	248
8.1.2.43	eventInfo	248
8.1.2.44	states_prev	249
8.1.2.45	visible	249
8.1.2.46	tStart	249
8.1.2.47	StopTimeDefined	249
8.1.2.48	tStop	249
8.1.2.49	input_real	249
8.1.2.50	output_real	249
8.2	fmi1_callback_functions_t Struct Reference	249
8.2.1	Detailed Description	250
8.2.2	Field Documentation	250
8.2.2.1	logger	250
8.2.2.2	allocateMemory	250
8.2.2.3	freeMemory	250
8.2.2.4	stepFinished	250
8.3	fmi1_event_info_t Struct Reference	250
8.3.1	Detailed Description	251
8.3.2	Field Documentation	251
8.3.2.1	iterationConverged	251
8.3.2.2	stateValueReferencesChanged	251
8.3.2.3	stateValuesChanged	251
8.3.2.4	terminateSimulation	251
8.3.2.5	upcomingTimeEvent	251
8.3.2.6	nextEventTime	251
8.4	fmi1_import_model_counts_t Struct Reference	251
8.4.1	Detailed Description	252
8.4.2	Field Documentation	252
8.4.2.1	num_constants	252
8.4.2.2	num_parameters	253
8.4.2.3	num_discrete	253
8.4.2.4	num_continuous	253
8.4.2.5	num_inputs	253
8.4.2.6	num_outputs	253

8.4.2.7	num_internal	253
8.4.2.8	num_causality_none	253
8.4.2.9	num_real_vars	253
8.4.2.10	num_integer_vars	254
8.4.2.11	num_enum_vars	254
8.4.2.12	num_bool_vars	254
8.4.2.13	num_string_vars	254
8.5	fmi1_me_callback_functions_t Struct Reference	254
8.5.1	Detailed Description	254
8.5.2	Field Documentation	255
8.5.2.1	logger	255
8.5.2.2	allocateMemory	255
8.5.2.3	freeMemory	255
8.6	fmi2_callback_functions_t Struct Reference	255
8.6.1	Detailed Description	255
8.6.2	Field Documentation	255
8.6.2.1	logger	255
8.6.2.2	allocateMemory	256
8.6.2.3	freeMemory	256
8.6.2.4	stepFinished	256
8.6.2.5	componentEnvironment	256
8.7	fmi2_event_info_t Struct Reference	256
8.7.1	Detailed Description	256
8.7.2	Field Documentation	257
8.7.2.1	newDiscreteStatesNeeded	257
8.7.2.2	terminateSimulation	257
8.7.2.3	nominalsOfContinuousStatesChanged	257
8.7.2.4	valuesOfContinuousStatesChanged	257
8.7.2.5	nextEventTimeDefined	257
8.7.2.6	nextEventTime	257
8.8	fmi2_import_model_counts_t Struct Reference	257
8.8.1	Detailed Description	258
8.8.2	Field Documentation	258
8.8.2.1	num_constants	258

8.8.2.2	num_fixed	259
8.8.2.3	num_tunable	259
8.8.2.4	num_discrete	259
8.8.2.5	num_continuous	259
8.8.2.6	num_parameters	259
8.8.2.7	num_calculated_parameters	259
8.8.2.8	num_inputs	259
8.8.2.9	num_outputs	259
8.8.2.10	num_local	260
8.8.2.11	num_independent	260
8.8.2.12	num_real_vars	260
8.8.2.13	num_integer_vars	260
8.8.2.14	num_enum_vars	260
8.8.2.15	num_bool_vars	260
8.8.2.16	num_string_vars	260
8.9	fmi2_xml_callbacks_t Struct Reference	261
8.9.1	Detailed Description	261
8.10	fmul_t Struct Reference	261
8.10.1	Detailed Description	261
8.10.2	Field Documentation	261
8.10.2.1	fmu	261
8.10.2.2	context	262
8.10.2.3	callbacks	262
8.10.2.4	callBackFunctions	262
8.11	jm_callbacks Struct Reference	262
8.11.1	Detailed Description	263
8.11.2	Field Documentation	263
8.11.2.1	malloc	263
8.11.2.2	calloc	263
8.11.2.3	realloc	263
8.11.2.4	free	263
8.11.2.5	logger	263
8.11.2.6	log_level	263
8.11.2.7	context	263

8.11.2.8	errMessageBuffer	264
8.12	jm_name_ID_map_t Struct Reference	264
8.12.1	Detailed Description	264
8.12.2	Field Documentation	264
8.12.2.1	name	264
8.12.2.2	ID	264
8.13	jm_named_ptr Struct Reference	265
8.13.1	Detailed Description	265
8.13.2	Field Documentation	265
8.13.2.1	ptr	265
8.13.2.2	name	265
9	File Documentation	267
9.1	build/doc/fmilib.h File Reference	267
9.1.1	Detailed Description	267
9.2	build/doc/fmilib_config.h File Reference	267
9.2.1	Detailed Description	268
9.2.2	Define Documentation	268
9.2.2.1	FMI_FILE_SEP	268
9.2.2.2	FMI_DLL_EXT	268
9.2.2.3	FMI_PLATFORM	268
9.2.2.4	FMI_BINARIES	268
9.2.2.5	FMI_MODEL_DESCRIPTION_XML	269
9.2.2.6	FMILIB_EXPORT	269
9.2.2.7	FMILIB_PRIVATE	269
9.2.2.8	FMILIB_SIZET_FORMAT	269
9.2.2.9	HAVE___VA_COPY	269
9.2.3	Function Documentation	269
9.2.3.1	fmilib_get_build_stamp	269
9.3	build/doc/fmilib_mainpage.h File Reference	269
9.3.1	Detailed Description	269
9.4	src/Import/include/FMI/fmi_import_context.h File Reference	270
9.4.1	Detailed Description	270
9.5	src/Import/include/FMI/fmi_import_util.h File Reference	271

9.6	src/Import/include/FMI1/fmi1_import.h File Reference	271
9.6.1	Detailed Description	273
9.7	src/Import/include/FMI1/fmi1_import_capabilities.h File Reference	273
9.7.1	Detailed Description	274
9.8	src/Import/include/FMI1/fmi1_import_capi.h File Reference	275
9.8.1	Detailed Description	278
9.9	src/Import/include/FMI1/fmi1_import_convenience.h File Reference	278
9.9.1	Detailed Description	279
9.10	src/Import/include/FMI1/fmi1_import_cosim.h File Reference	279
9.10.1	Detailed Description	279
9.11	src/Import/include/FMI1/fmi1_import_type.h File Reference	279
9.11.1	Detailed Description	281
9.12	src/Import/include/FMI1/fmi1_import_unit.h File Reference	282
9.12.1	Detailed Description	283
9.13	src/Import/include/FMI1/fmi1_import_variable.h File Reference	283
9.13.1	Detailed Description	286
9.14	src/Import/include/FMI1/fmi1_import_variable_list.h File Reference	286
9.14.1	Detailed Description	287
9.15	src/Import/include/FMI1/fmi1_import_vendor_annotations.h File - Reference	287
9.15.1	Detailed Description	288
9.16	src/Import/include/FMI2/fmi2_import.h File Reference	288
9.16.1	Detailed Description	292
9.17	src/Import/include/FMI2/fmi2_import_capi.h File Reference	292
9.17.1	Detailed Description	295
9.18	src/Import/include/FMI2/fmi2_import_convenience.h File Reference	296
9.18.1	Detailed Description	297
9.19	src/Import/include/FMI2/fmi2_import_type.h File Reference	297
9.19.1	Detailed Description	299
9.20	src/Import/include/FMI2/fmi2_import_unit.h File Reference	299
9.20.1	Detailed Description	301
9.21	src/Import/include/FMI2/fmi2_import_variable.h File Reference	301
9.21.1	Detailed Description	304
9.22	src/Import/include/FMI2/fmi2_import_variable_list.h File Reference	304

9.22.1 Detailed Description	305
9.23 src/Util/include/FMI/fmi_util.h File Reference	305
9.23.1 Detailed Description	305
9.24 src/Util/include/FMI/fmi_version.h File Reference	306
9.24.1 Detailed Description	306
9.25 src/Util/include/FMI1/fmi1_enums.h File Reference	306
9.25.1 Detailed Description	307
9.26 src/Util/include/FMI1/fmi1_functions.h File Reference	308
9.26.1 Detailed Description	310
9.27 src/Util/include/FMI1/fmi1_types.h File Reference	310
9.27.1 Detailed Description	311
9.28 src/Util/include/FMI2/fmi2_enums.h File Reference	311
9.28.1 Detailed Description	313
9.28.2 Define Documentation	314
9.28.2.1 FMI2_EXPAND_ME_CAPABILITIES_ENU	314
9.28.2.2 FMI2_EXPAND_CS_CAPABILITIES_ENU	314
9.28.2.3 FMI2_EXPAND_SI_BASE_UNIT_ENU	314
9.29 src/Util/include/FMI2/fmi2_functions.h File Reference	314
9.29.1 Detailed Description	316
9.30 src/Util/include/FMI2/fmi2_types.h File Reference	316
9.30.1 Detailed Description	317
9.31 src/Util/include/FMI2/fmi2_xml_callbacks.h File Reference	317
9.31.1 Detailed Description	318
9.32 src/Util/include/JM/jm_callbacks.h File Reference	318
9.32.1 Detailed Description	320
9.33 src/Util/include/JM/jm_named_ptr.h File Reference	320
9.33.1 Detailed Description	321
9.33.2 Define Documentation	321
9.33.2.1 jm_diff_named	321
9.34 src/Util/include/JM/jm_portability.h File Reference	321
9.34.1 Detailed Description	322
9.34.2 Define Documentation	323
9.34.2.1 DLL_HANDLE	323
9.35 src/Util/include/JM/jm_stack.h File Reference	323

9.36	src/Util/include/JM/jm_string_set.h File Reference	323
9.36.1	Detailed Description	324
9.37	src/Util/include/JM/jm_types.h File Reference	324
9.37.1	Detailed Description	325
9.38	src/Util/include/JM/jm_vector.h File Reference	325
9.38.1	Detailed Description	327
9.38.2	Define Documentation	327
9.38.2.1	jm_diff_name	327
9.39	src/Util/include/JM/jm_vector_template.h File Reference	327
9.39.1	Detailed Description	328
9.39.2	Define Documentation	328
9.39.2.1	jm_vector_ptr2index	328
9.40	Test/compress_test_fmu_zip.c File Reference	328
9.40.1	Function Documentation	328
9.40.1.1	importlogger	328
9.40.1.2	main	328
9.41	Test/FMI1/compress_test_fmu_zip.c File Reference	329
9.41.1	Function Documentation	329
9.41.1.1	importlogger	329
9.41.1.2	main	329
9.42	Test/FMI1/fmi1_capi_cs_test.c File Reference	329
9.42.1	Define Documentation	331
9.42.1.1	MODEL_IDENTIFIER	331
9.42.1.2	INSTANCE_NAME	331
9.42.2	Function Documentation	332
9.42.2.1	importlogger	332
9.42.2.2	fmilogger	332
9.42.2.3	do_exit	332
9.42.2.4	test_create_dllfmu	332
9.42.2.5	test_load_dll	332
9.42.2.6	test_load_dll_fcn	332
9.42.2.7	test_fmi_get_version	332
9.42.2.8	test_get_types_platform	332
9.42.2.9	test_instantiate_slave	333

9.42.2.10	test_initialize_slave	333
9.42.2.11	test_set_debug_logging	333
9.42.2.12	test_cancel_step	333
9.42.2.13	test_do_step	333
9.42.2.14	test_get_status	333
9.42.2.15	test_get_real_status	333
9.42.2.16	test_get_integer_status	333
9.42.2.17	test_get_boolean_status	334
9.42.2.18	test_get_string_status	334
9.42.2.19	test_set_get_string	334
9.42.2.20	test_set_get_integer	334
9.42.2.21	test_set_get_boolean	334
9.42.2.22	test_set_get_real	334
9.42.2.23	test_reset_slave	334
9.42.2.24	test_set_real_input_derivatives	335
9.42.2.25	test_get_real_output_derivatives	335
9.42.2.26	test_terminate_slave	335
9.42.2.27	test_free_slave_instance	335
9.42.2.28	test_free_dll	335
9.42.2.29	test_destroy_dllfmu	335
9.42.2.30	main	335
9.42.3	Variable Documentation	336
9.42.3.1	fmu	336
9.42.3.2	callbacks	336
9.43	Test/FMI1/fmi1_capi_me_test.c File Reference	336
9.43.1	Define Documentation	338
9.43.1.1	MODEL_IDENTIFIER	338
9.43.1.2	INSTANCE_NAME	338
9.43.2	Function Documentation	338
9.43.2.1	importlogger	338
9.43.2.2	fmllogger	338
9.43.2.3	do_exit	338
9.43.2.4	test_create_dllfmu	338
9.43.2.5	test_load_dll	339

9.43.2.6	test_load_dll_fcn	339
9.43.2.7	test_fmi_get_version	339
9.43.2.8	test_fmi_get_model_types_platform	339
9.43.2.9	test_instantiate_model	339
9.43.2.10	test_fmi_set_time	339
9.43.2.11	test_set_continuous_states	339
9.43.2.12	test_initialize	339
9.43.2.13	test_completed_integrator_step	340
9.43.2.14	test_get_derivatives	340
9.43.2.15	test_get_event_indicators	340
9.43.2.16	test_event_update	340
9.43.2.17	test_get_continuous_states	340
9.43.2.18	test_get_nominal_continuous_states	340
9.43.2.19	test_get_state_value_references	340
9.43.2.20	test_set_debug_logging	340
9.43.2.21	test_set_get_string	341
9.43.2.22	test_set_get_integer	341
9.43.2.23	test_set_get_boolean	341
9.43.2.24	test_set_get_real	341
9.43.2.25	test_terminate	341
9.43.2.26	test_free_model_instance	341
9.43.2.27	test_free_dll	341
9.43.2.28	test_destroy_dllfmu	342
9.43.2.29	main	342
9.43.3	Variable Documentation	342
9.43.3.1	fmu	342
9.43.3.2	callbacks	342
9.44	Test/FMI1/fmi1_import_test.c File Reference	342
9.44.1	Define Documentation	343
9.44.1.1	BUFFER	343
9.44.2	Function Documentation	343
9.44.2.1	fmi1logger	343
9.44.2.2	fmi1_test	343
9.45	Test/FMI1/fmi1_logger_test.c File Reference	343

9.45.1	Define Documentation	344
9.45.1.1	MESSAGE_SIZE_TO_EXPAND_AND_PRINT	344
9.45.2	Function Documentation	344
9.45.2.1	logger	344
9.45.2.2	do_exit	344
9.45.2.3	test_logger	344
9.45.2.4	main	344
9.45.3	Variable Documentation	344
9.45.3.1	logFile	344
9.46	Test/FMI1/fmi2_import_xml_test.cc File Reference	344
9.46.1	Function Documentation	345
9.46.1.1	mylogger	345
9.46.1.2	annotation_start_handle	345
9.46.1.3	annotation_data_handle	345
9.46.1.4	annotation_end_handle	345
9.46.1.5	do_exit	345
9.46.1.6	print_int	345
9.46.1.7	print_dbl	346
9.46.1.8	printTypeInfo	346
9.46.1.9	testVariableSearch	346
9.46.1.10	printVariableInfo	346
9.46.1.11	printCapabilitiesInfo	346
9.46.1.12	printDependenciesInfo	346
9.46.1.13	main	346
9.46.2	Variable Documentation	346
9.46.2.1	annotation_callbacks	346
9.47	Test/FMI2/fmi2_import_xml_test.cc File Reference	347
9.47.1	Function Documentation	347
9.47.1.1	mylogger	347
9.47.1.2	annotation_start_handle	347
9.47.1.3	annotation_data_handle	348
9.47.1.4	annotation_end_handle	348
9.47.1.5	do_exit	348
9.47.1.6	print_int	348

9.47.1.7	print_dbl	348
9.47.1.8	printTypeInfo	348
9.47.1.9	testVariableSearch	348
9.47.1.10	printVariableInfo	348
9.47.1.11	printCapabilitiesInfo	348
9.47.1.12	printDependenciesInfo	348
9.47.1.13	main	349
9.47.2	Variable Documentation	349
9.47.2.1	annotation_callbacks	349
9.48	Test/FMI1/fmi_import_cs_test.c File Reference	349
9.48.1	Define Documentation	349
9.48.1.1	BUFFER	349
9.48.2	Function Documentation	350
9.48.2.1	importlogger	350
9.48.2.2	fmilogger	350
9.48.2.3	do_exit	350
9.48.2.4	test_simulate_cs	350
9.48.2.5	main	350
9.49	Test/FMI1/fmi_import_me_test.c File Reference	350
9.49.1	Define Documentation	351
9.49.1.1	BUFFER	351
9.49.1.2	NUMBER_OF_TESTS	351
9.49.2	Function Documentation	351
9.49.2.1	do_exit	351
9.49.2.2	test_simulate_me	351
9.49.2.3	load	351
9.49.2.4	destroy	351
9.49.2.5	main	351
9.50	Test/FMI1/fmi_import_xml_test.cc File Reference	351
9.50.1	Function Documentation	352
9.50.1.1	mylogger	352
9.50.1.2	do_exit	352
9.50.1.3	print_int	352
9.50.1.4	print_dbl	352

9.50.1.5	printTypeInfo	352
9.50.1.6	testVariableSearch	352
9.50.1.7	printVariableInfo	352
9.50.1.8	printCapabilitiesInfo	353
9.50.1.9	main	353
9.51	Test/FMI1/fmi_total_test.c File Reference	353
9.51.1	Define Documentation	353
9.51.1.1	PRINT_MY_DEBUG	353
9.51.2	Function Documentation	353
9.51.2.1	mylogger	353
9.51.2.2	do_pause	354
9.51.2.3	main	354
9.52	Test/FMI1/fmi_zip_unzip_test.c File Reference	354
9.52.1	Function Documentation	354
9.52.1.1	do_exit	354
9.52.1.2	importlogger	354
9.52.1.3	main	354
9.53	Test/fmi_zip_unzip_test.c File Reference	355
9.53.1	Function Documentation	355
9.53.1.1	do_exit	355
9.53.1.2	importlogger	355
9.53.1.3	main	355
9.54	Test/FMI1/fmi_zip_zip_test.c File Reference	355
9.54.1	Define Documentation	356
9.54.1.1	PRINT_MY_DEBUG	356
9.54.2	Function Documentation	356
9.54.2.1	do_exit	356
9.54.2.2	importlogger	356
9.54.2.3	main	356
9.55	Test/fmi_zip_zip_test.c File Reference	356
9.55.1	Define Documentation	357
9.55.1.1	PRINT_MY_DEBUG	357
9.55.2	Function Documentation	357
9.55.2.1	do_exit	357

9.55.2.2	importlogger	357
9.55.2.3	main	357
9.56	Test/FMI1/fmu_dummy/fmu1_model.c File Reference	357
9.56.1	Define Documentation	359
9.56.1.1	FMI_TEST_LOGGER_TEST_RESULT_FILE	359
9.56.1.2	FMI_TEST_LOGGER_TEST_SOURCE_FILE	359
9.56.2	Function Documentation	359
9.56.2.1	calc_initialize	359
9.56.2.2	calc_get_derivatives	360
9.56.2.3	calc_get_event_indicators	360
9.56.2.4	calc_event_update	360
9.56.2.5	fmi_get_version	360
9.56.2.6	fmi_set_debug_logging	360
9.56.2.7	fmi_get_real	360
9.56.2.8	fmi_get_integer	360
9.56.2.9	fmi_get_boolean	360
9.56.2.10	fmi_get_string	360
9.56.2.11	fmi_set_real	361
9.56.2.12	fmi_set_integer	361
9.56.2.13	fmi_set_boolean	361
9.56.2.14	fmi_set_string	361
9.56.2.15	fmi_get_model_types_platform	361
9.56.2.16	fmi_instantiate_model	361
9.56.2.17	fmi_free_model_instance	361
9.56.2.18	fmi_set_time	361
9.56.2.19	fmi_set_continuous_states	361
9.56.2.20	fmi_completed_integrator_step	362
9.56.2.21	fmi_initialize	362
9.56.2.22	fmi_get_derivatives	362
9.56.2.23	fmi_get_event_indicators	362
9.56.2.24	fmi_event_update	362
9.56.2.25	fmi_get_continuous_states	362
9.56.2.26	fmi_get_nominal_continuousstates	362
9.56.2.27	fmi_get_state_value_references	362

9.56.2.28	fmi_terminate	362
9.56.2.29	fmi_get_types_platform	363
9.56.2.30	fmi_instantiate_slave	363
9.56.2.31	fmi_initialize_slave	363
9.56.2.32	fmi_terminate_slave	363
9.56.2.33	fmi_reset_slave	363
9.56.2.34	fmi_free_slave_instance	363
9.56.2.35	fmi_set_real_input_derivatives	363
9.56.2.36	fmi_get_real_output_derivatives	363
9.56.2.37	fmi_cancel_step	363
9.56.2.38	fmi_do_step	364
9.56.2.39	fmi_get_status	364
9.56.2.40	fmi_get_real_status	364
9.56.2.41	fmi_get_integer_status	364
9.56.2.42	fmi_get_boolean_status	364
9.56.2.43	fmi_get_string_status	364
9.57	Test/FMI1/fmu_dummy/fmu1_model.h File Reference	364
9.57.1	Typedef Documentation	365
9.57.1.1	component_ptr_t	365
9.57.2	Function Documentation	366
9.57.2.1	fmi_get_version	366
9.57.2.2	fmi_set_debug_logging	366
9.57.2.3	fmi_get_real	366
9.57.2.4	fmi_get_integer	366
9.57.2.5	fmi_get_boolean	366
9.57.2.6	fmi_get_string	366
9.57.2.7	fmi_set_real	366
9.57.2.8	fmi_set_integer	366
9.57.2.9	fmi_set_boolean	367
9.57.2.10	fmi_set_string	367
9.57.2.11	fmi_get_model_types_platform	367
9.57.2.12	fmi_instantiate_model	367
9.57.2.13	fmi_free_model_instance	367
9.57.2.14	fmi_set_time	367

9.57.2.15 fmi_set_continuous_states	367
9.57.2.16 fmi_completed_integrator_step	367
9.57.2.17 fmi_initialize	367
9.57.2.18 fmi_get_derivatives	368
9.57.2.19 fmi_get_event_indicators	368
9.57.2.20 fmi_event_update	368
9.57.2.21 fmi_get_continuous_states	368
9.57.2.22 fmi_get_nominal_continuousstates	368
9.57.2.23 fmi_get_state_value_references	368
9.57.2.24 fmi_terminate	368
9.58 Test/FMI1/fmu_dummy/fmu1_model_cs.c File Reference	368
9.58.1 Define Documentation	370
9.58.1.1 MODEL_IDENTIFIER	370
9.58.2 Function Documentation	370
9.58.2.1 fmiGetVersion	370
9.58.2.2 fmiSetDebugLogging	370
9.58.2.3 fmiGetReal	370
9.58.2.4 fmiGetInteger	370
9.58.2.5 fmiGetBoolean	370
9.58.2.6 fmiGetString	370
9.58.2.7 fmiSetReal	370
9.58.2.8 fmiSetInteger	371
9.58.2.9 fmiSetBoolean	371
9.58.2.10 fmiSetString	371
9.58.2.11 fmiGetTypesPlatform	371
9.58.2.12 fmiInstantiateSlave	371
9.58.2.13 fmiInitializeSlave	371
9.58.2.14 fmiTerminateSlave	371
9.58.2.15 fmiResetSlave	371
9.58.2.16 fmiFreeSlaveInstance	371
9.58.2.17 fmiSetRealInputDerivatives	372
9.58.2.18 fmiGetRealOutputDerivatives	372
9.58.2.19 fmiCancelStep	372
9.58.2.20 fmiDoStep	372

9.58.2.21 fmiGetStatus	372
9.58.2.22 fmiGetRealStatus	372
9.58.2.23 fmiGetIntegerStatus	372
9.58.2.24 fmiGetBooleanStatus	372
9.58.2.25 fmiGetStringStatus	373
9.59 Test/FMI1/fmu_dummy/fmu1_model_defines.h File Reference	373
9.59.1 Define Documentation	373
9.59.1.1 STRINGIFY	373
9.59.1.2 STRINGIFY2	373
9.59.1.3 MODEL_IDENTIFIER_STR	374
9.59.1.4 BUFFER	374
9.59.1.5 MAGIC_TEST_VALUE	374
9.59.1.6 VAR_R_HIGHT	374
9.59.1.7 VAR_R_HIGHT_SPEED	374
9.59.1.8 VAR_R_GRATIVY	374
9.59.1.9 VAR_R_BOUNCE_CONF	374
9.59.1.10 EVENT_HIGHT	374
9.59.1.11 VAR_S_LOGGER_TEST	374
9.59.1.12 N_STATES	374
9.59.1.13 N_EVENT_INDICATORS	375
9.59.1.14 N_REAL	375
9.59.1.15 N_INTEGER	375
9.59.1.16 N_BOOLEAN	375
9.59.1.17 N_STRING	375
9.59.1.18 N_INPUT_REAL	375
9.59.1.19 N_INPUT_REAL_MAX_ORDER	375
9.59.1.20 N_OUTPUT_REAL	375
9.59.1.21 N_OUTPUT_REAL_MAX_ORDER	375
9.59.1.22 FMI_VERSION	375
9.59.1.23 FMI_GUID	376
9.60 Test/FMI1/fmu_dummy/fmu1_model_me.c File Reference	376
9.60.1 Define Documentation	377
9.60.1.1 MODEL_IDENTIFIER	377
9.60.2 Enumeration Type Documentation	377

9.60.2.1	fmiStatusKind	377
9.60.3	Function Documentation	377
9.60.3.1	fmiGetVersion	377
9.60.3.2	fmiSetDebugLogging	378
9.60.3.3	fmiGetReal	378
9.60.3.4	fmiGetInteger	378
9.60.3.5	fmiGetBoolean	378
9.60.3.6	fmiGetString	378
9.60.3.7	fmiSetReal	378
9.60.3.8	fmiSetInteger	378
9.60.3.9	fmiSetBoolean	378
9.60.3.10	fmiSetString	378
9.60.3.11	fmiGetModelTypesPlatform	379
9.60.3.12	fmiInstantiateModel	379
9.60.3.13	fmiFreeModelInstance	379
9.60.3.14	fmiSetTime	379
9.60.3.15	fmiSetContinuousStates	379
9.60.3.16	fmiCompletedIntegratorStep	379
9.60.3.17	fmiInitialize	379
9.60.3.18	fmiGetDerivatives	379
9.60.3.19	fmiGetEventIndicators	379
9.60.3.20	fmiEventUpdate	380
9.60.3.21	fmiGetContinuousStates	380
9.60.3.22	fmiGetNominalContinuousStates	380
9.60.3.23	fmiGetStateValueReferences	380
9.60.3.24	fmiTerminate	380
9.61	Test/FMI1/jm_vector_test.c File Reference	380
9.61.1	Define Documentation	381
9.61.1.1	TESTVAL	381
9.61.1.2	VINIT_SIZE	381
9.61.2	Function Documentation	381
9.61.2.1	print_int	381
9.61.2.2	print_dbl	381
9.61.2.3	compar_int	381

9.61.2.4	log_error	381
9.61.2.5	main	381
9.61.3	Variable Documentation	381
9.61.3.1	return_code	381
9.62	Test/jm_vector_test.c File Reference	382
9.62.1	Define Documentation	382
9.62.1.1	TESTVAL	382
9.62.1.2	VINIT_SIZE	382
9.62.2	Function Documentation	382
9.62.2.1	print_int	382
9.62.2.2	print_dbl	382
9.62.2.3	compar_int	383
9.62.2.4	log_error	383
9.62.2.5	main	383
9.62.3	Variable Documentation	383
9.62.3.1	return_code	383
9.63	Test/FMI2/fmi2_import_cs_test.c File Reference	383
9.63.1	Define Documentation	383
9.63.1.1	BUFFER	383
9.63.2	Function Documentation	384
9.63.2.1	importlogger	384
9.63.2.2	fmilogger	384
9.63.2.3	do_exit	384
9.63.2.4	test_simulate_cs	384
9.63.2.5	main	384
9.64	Test/FMI2/fmi2_import_me_test.c File Reference	384
9.64.1	Define Documentation	385
9.64.1.1	BUFFER	385
9.64.2	Function Documentation	385
9.64.2.1	do_exit	385
9.64.2.2	do_event_iteration	385
9.64.2.3	test_simulate_me	385
9.64.2.4	main	385
9.65	Test/FMI2/fmi2_import_test.c File Reference	385

9.65.1	Define Documentation	386
9.65.1.1	BUFFER	386
9.65.2	Function Documentation	386
9.65.2.1	fmi2logger	386
9.65.2.2	stepFinished	386
9.65.2.3	fmi2_test	386
9.66	Test/FMI2/fmu_dummy/fmu2_model.c File Reference	386
9.66.1	Function Documentation	388
9.66.1.1	calc_initialize	388
9.66.1.2	calc_get_derivatives	388
9.66.1.3	calc_get_event_indicators	388
9.66.1.4	calc_event_update	388
9.66.1.5	fmi_get_version	388
9.66.1.6	fmi_set_debug_logging	388
9.66.1.7	fmi_get_real	388
9.66.1.8	fmi_get_integer	389
9.66.1.9	fmi_get_boolean	389
9.66.1.10	fmi_get_string	389
9.66.1.11	fmi_set_real	389
9.66.1.12	fmi_set_integer	389
9.66.1.13	fmi_set_boolean	389
9.66.1.14	fmi_set_string	389
9.66.1.15	fmi_get_model_types_platform	389
9.66.1.16	fmi_instantiate	390
9.66.1.17	fmi_free_instance	390
9.66.1.18	fmi_setup_experiment	390
9.66.1.19	fmi_enter_initialization_mode	390
9.66.1.20	fmi_exit_initialization_mode	390
9.66.1.21	fmi_enter_event_mode	390
9.66.1.22	fmi_new_discrete_states	390
9.66.1.23	fmi_enter_continuous_time_mode	390
9.66.1.24	fmi_set_time	390
9.66.1.25	fmi_set_continuous_states	391
9.66.1.26	fmi_completed_integrator_step	391

9.66.1.27	fmi_get_derivatives	391
9.66.1.28	fmi_get_event_indicators	391
9.66.1.29	fmi_get_continuous_states	391
9.66.1.30	fmi_get_nominals_of_continuousstates	391
9.66.1.31	fmi_terminate	391
9.66.1.32	fmi_get_types_platform	391
9.66.1.33	fmi_reset	391
9.66.1.34	fmi_set_real_input_derivatives	392
9.66.1.35	fmi_get_real_output_derivatives	392
9.66.1.36	fmi_cancel_step	392
9.66.1.37	fmi_do_step	392
9.66.1.38	fmi_get_status	392
9.66.1.39	fmi_get_real_status	392
9.66.1.40	fmi_get_integer_status	392
9.66.1.41	fmi_get_boolean_status	392
9.66.1.42	fmi_get_string_status	393
9.67	Test/FMI2/fmu_dummy/fmu2_model.h File Reference	393
9.67.1	Define Documentation	394
9.67.1.1	FMI2_Export	394
9.67.2	Typedef Documentation	394
9.67.2.1	component_ptr_t	394
9.67.3	Function Documentation	394
9.67.3.1	fmi_get_version	394
9.67.3.2	fmi_set_debug_logging	395
9.67.3.3	fmi_instantiate	395
9.67.3.4	fmi_free_instance	395
9.67.3.5	fmi_setup_experiment	395
9.67.3.6	fmi_enter_initialization_mode	395
9.67.3.7	fmi_exit_initialization_mode	395
9.67.3.8	fmi_terminate	395
9.67.3.9	fmi_reset	395
9.67.3.10	fmi_get_real	395
9.67.3.11	fmi_get_integer	396
9.67.3.12	fmi_get_boolean	396

9.67.3.13 fmi_get_string	396
9.67.3.14 fmi_set_real	396
9.67.3.15 fmi_set_integer	396
9.67.3.16 fmi_set_boolean	396
9.67.3.17 fmi_set_string	396
9.67.3.18 fmi_get_model_types_platform	396
9.67.3.19 fmi_enter_event_mode	396
9.67.3.20 fmi_new_discrete_states	397
9.67.3.21 fmi_enter_continuous_time_mode	397
9.67.3.22 fmi_set_time	397
9.67.3.23 fmi_set_continuous_states	397
9.67.3.24 fmi_completed_integrator_step	397
9.67.3.25 fmi_get_derivatives	397
9.67.3.26 fmi_get_event_indicators	397
9.67.3.27 fmi_get_continuous_states	397
9.67.3.28 fmi_get_nominals_of_continuousstates	397
9.68 Test/FMI2/fmu_dummy/fmu2_model_cs.c File Reference	398
9.68.1 Function Documentation	399
9.68.1.1 fmi2GetVersion	399
9.68.1.2 fmi2SetDebugLogging	399
9.68.1.3 fmi2Instantiate	399
9.68.1.4 fmi2FreeInstance	399
9.68.1.5 fmi2SetupExperiment	399
9.68.1.6 fmi2EnterInitializationMode	399
9.68.1.7 fmi2ExitInitializationMode	400
9.68.1.8 fmi2GetReal	400
9.68.1.9 fmi2GetInteger	400
9.68.1.10 fmi2GetBoolean	400
9.68.1.11 fmi2GetString	400
9.68.1.12 fmi2SetReal	400
9.68.1.13 fmi2SetInteger	400
9.68.1.14 fmi2SetBoolean	400
9.68.1.15 fmi2SetString	400
9.68.1.16 fmi2GetTypesPlatform	401

9.68.1.17 fmi2Terminate	401
9.68.1.18 fmi2Reset	401
9.68.1.19 fmi2SetRealInputDerivatives	401
9.68.1.20 fmi2GetRealOutputDerivatives	401
9.68.1.21 fmi2CancelStep	401
9.68.1.22 fmi2DoStep	401
9.68.1.23 fmi2GetStatus	401
9.68.1.24 fmi2GetRealStatus	401
9.68.1.25 fmi2GetIntegerStatus	402
9.68.1.26 fmi2GetBooleanStatus	402
9.68.1.27 fmi2GetStringStatus	402
9.69 Test/FMI2/fmu_dummy/fmu2_model_defines.h File Reference	402
9.69.1 Define Documentation	402
9.69.1.1 BUFFER	402
9.69.1.2 MAGIC_TEST_VALUE	403
9.69.1.3 VAR_R_HIGHT	403
9.69.1.4 VAR_R_HIGHT_SPEED	403
9.69.1.5 VAR_R_GRATIVY	403
9.69.1.6 VAR_R_BOUNCE_CONF	403
9.69.1.7 EVENT_HIGHT	403
9.69.1.8 VAR_S_LOGGER_TEST	403
9.69.1.9 N_STATES	403
9.69.1.10 N_EVENT_INDICATORS	403
9.69.1.11 N_REAL	403
9.69.1.12 N_INTEGER	404
9.69.1.13 N_BOOLEAN	404
9.69.1.14 N_STRING	404
9.69.1.15 N_INPUT_REAL	404
9.69.1.16 N_INPUT_REAL_MAX_ORDER	404
9.69.1.17 N_OUTPUT_REAL	404
9.69.1.18 N_OUTPUT_REAL_MAX_ORDER	404
9.69.1.19 FMI_VERSION	404
9.69.1.20 FMI_GUID	404
9.70 Test/FMI2/fmu_dummy/fmu2_model_me.c File Reference	404

9.70.1	Function Documentation	406
9.70.1.1	fmi2GetVersion	406
9.70.1.2	fmi2SetDebugLogging	406
9.70.1.3	fmi2Instantiate	406
9.70.1.4	fmi2FreeInstance	406
9.70.1.5	fmi2SetupExperiment	406
9.70.1.6	fmi2EnterInitializationMode	406
9.70.1.7	fmi2ExitInitializationMode	406
9.70.1.8	fmi2GetReal	407
9.70.1.9	fmi2GetInteger	407
9.70.1.10	fmi2GetBoolean	407
9.70.1.11	fmi2GetString	407
9.70.1.12	fmi2SetReal	407
9.70.1.13	fmi2SetInteger	407
9.70.1.14	fmi2SetBoolean	407
9.70.1.15	fmi2SetString	407
9.70.1.16	fmi2GetTypesPlatform	407
9.70.1.17	fmi2EnterEventMode	408
9.70.1.18	fmi2NewDiscreteStates	408
9.70.1.19	fmi2EnterContinuousTimeMode	408
9.70.1.20	fmi2SetTime	408
9.70.1.21	fmi2SetContinuousStates	408
9.70.1.22	fmi2CompletedIntegratorStep	408
9.70.1.23	fmi2GetDerivatives	408
9.70.1.24	fmi2GetEventIndicators	408
9.70.1.25	fmi2GetContinuousStates	408
9.70.1.26	fmi2GetNominalsOfContinuousStates	409
9.70.1.27	fmi2Terminate	409
9.70.1.28	fmi2Reset	409
9.71	Test/fmi_import_test.c File Reference	409
9.71.1	Define Documentation	409
9.71.1.1	BUFFER	409
9.71.2	Function Documentation	409
9.71.2.1	fmi1_test	409

9.71.2.2	fmi2_test	410
9.71.2.3	importlogger	410
9.71.2.4	do_exit	410
9.71.2.5	main	410

Chapter 1

FMI Library: part of JModelica.org

Version

2.0.1

Date

24 October 2014

1.1 Summary

FMI library is intended as a foundation for applications interfacing FMUs (Functional - Mockup Units) that follow FMI Standard. This version of the library supports FMI 1.0 and FMI2.0. See <<http://www.fmi-standard.org/>>

The test codes provided with the library can serve as starting point for the development of custom applications. See Section [Automatic tests](#) for details.

1.2 Configuring and building

CMake (see <<http://www.cmake.org/>>) is used to generate the native build scripts for the library. It is recommended to use "cmake-gui" on Windows or "ccmake <FMIL source dir>" to configure the build. All the required third party libraries are included into the distribution.

CMake 2.8.6 is required since this is the version used in development both on Windows and Linux. The build script is KNOWN NOT TO WORK WITH CMake 2.8.3 and below (due to ExternalProject interface changes). CMake 2.8.4 and 2.8.5 are not tested.

To build from a terminal command line on Linux or Mac with default settings use:

```
mkdir build-fmil; cd build-fmil
cmake -DFMILIB_INSTALL_PREFIX=<prefix> <path to FMIL source>
make install test
```

To build in MSYS terminal with g++/gcc on Windows:

```
mkdir build-fmil; cd build-fmil
cmake -DFMILIB_INSTALL_PREFIX=<prefix> -G "MSYS Makefiles" <path to
FMIL source>
make install test
```

To build from command line with Microsoft Visual Studio compilers on Windows:

```
mkdir build-fmil; cd build-fmil
cmake -DFMILIB_INSTALL_PREFIX=<prefix> -G "Visual Studio 10" <path to
FMIL source>
cmake --build . --config MinSizeRel --target install
```

The primary targets of the library build script are:

- `<prefix>/include/fmilib.h`
The include file to be used in client applications.
- Library files under `<prefix>/lib/`
Static library is named 'fmilib' and shared library 'fmilib_shared'. The prefix/suffix of the library files differ depending on the platform. Note that if you have configure and built both static and shared library on Windows but want to link with the static library compile time define "FMILIB_BUILDING_LIBRARY" must be set.
- Doxygen generated documentation under `<prefix>/doc/`. Note that documentation is not generated as part of install target and you need to build the 'doc' target separately, i.e., run 'make doc' or explicitly build the project in Visual Studio.

The following build configuration options are provided:

- **FMILIB_INSTALL_PREFIX** - prefix prepended to install directories.
Default: `"../install"`
This is the main install directory name. Include files will be located in the "include" subdirectory and library files in the "lib" subdirectory. Client applications should only include "fmilib.h"
- **FMILIB_THIRDPARTYLIBS** - thirdparty libraries are currently shipped with the library.
- **FMILIB_FMI_STANDARD_HEADERS** - Path to the FMI standard headers directory. Header for specific standards files are expected in subdirectories FMI1, FMI2, etc.
Default: `"ThirdParty/FMI/default"`
- **FMILIB_DEFAULT_BUILD_TYPE_RELEASE** - Controls build-type used for - Makefile generation.
Default: ON
If this option is on then 'Release' mode compile flags are used. Otherwise, '-Debug' mode flags are generated into the Makefile. The option may be overwritten by explicitly setting CMAKE_BUILD_TYPE.

- **FMILIB_BUILD_WITH_STATIC_RT_LIB** Use static run-time libraries (/MT or /MTd code generation flags).
Default: OFF
This is only used when generating Microsoft Visual Studio solutions. If the option is on then the library will be built against static runtime, otherwise - dynamic runtime (/MD or /MDd). Make sure the client code is using matching runtime.
- **FMILIB_BUILD_STATIC_LIB** Build the library as static.
Default: ON
'fmilib' may be used for static linking.
- **FMILIB_BUILD_SHARED_LIB** Build the library as shared (dll/so/dylib).
Default: ON
'fmilib_shared' may be used for dynamic linking.
- **FMILIB_FMI_PLATFORM** - FMI platform defines the subdirectory within FMU where binary is located.
The build system will automatically detect win32, win64, linux32, linux64, darwin32, darwin64.
- **FMILIB_BUILD_FOR_SHARED_LIBS** The static library 'fmilib' can be linked into shared libraries.
Default: ON
On LINUX position independent code (-fPIC) must be used on all files to be linked into a shared library (.so file). On other systems this is not needed (either is default or relocation is done). Set this option to OFF if you are building an application on Linux and use static library only.
- **FMILIB_ENABLE_LOG_LEVEL_DEBUG** Enable log level 'debug'.
Default: OFF
If the option is OFF then the debug level messages are not compiled in.
- **FMILIB_GENERATE_DOXYGEN_DOC** Enable doxygen target.
Default: ON
You need Doxygen to be installed on the system for this option to have an effect.
- **FMILIB_BUILD_TESTS** Enable build of the tests supplied with the library.
Default: ON
RUN_TESTS - target will be provided in Visual Studio. 'make test' will run tests on Makefile based platforms.
- **FMILIB_BUILD_BEFORE_TESTS** Force build before testing, i.e., building the FMI library becomes the first test.
Default: ON

- **FMILIB_LINK_TEST_TO_SHAREDLIB** Link the tests to `fmilib_shared` (if built) instead of static `fmilib`.

Default: ON

- **FMILIB_GENERATE_BUILD_STAMP** Generate a build time stamp and include in into the library.

Default: OFF

The function `fmilib_get_build_stamp()` may be used to retrieve the time stamp.

```
const char* fmilib_get_build_stamp(void);
```

1.3 Automatic tests

The FMI library comes with a number of automatic tests. Building of the test is controlled by `FMILIB_BUILD_TESTS` configuration option. The test programs are also intended as examples of library usage.

The tests can be run in Visual Studio by building project `RUN_TESTS`. For Makefile based configurations (MSYS, Linux, Mac OSx) run 'make test'.

Output from the test programs and test logs can be found in the *Testing* folder in the build directory.

The supplied tests are:

- **ctest_build_all**

Build the library. This test is controlled by `FMILIB_BUILD_BEFORE_TESTS` configuration option.

- **ctest_fmi_zip_unzip_test**

Basic unzip functionality test. Test executable is `fmi_zip_unzip_test`.

- **ctest_fmi_zip_zip_test**

Basic zip functionality test. Test executable is `fmi_zip_zip_test`.

- **ctest_fmi_import_me_test**

Load a basic model exchange FMU and simulate it. Test executable is `fmi_import_me_test`, main source file is `fmi_import_me_test.c`.

- **ctest_fmi_import_cs_test**

Load a basic co-simulation FMU and simulate it. Test executable is `fmi_import_cs_test`, main source file is `fmi_import_cs_test.c`.

- **ctest_fmi_import_cs_tc_test**

Load a basic co-simulation tools coupling FMU and simulate it. Test executable is `fmi_import_cs_test`, main source file is `fmi_import_cs_test.c`.

- **ctest_fmi_import_xml_test, ctest_fmi_import_xml_test_empty, ctest_fmi_import_xml_test_mf**

Load a small model description XML file and print out detailed information on it. Test executable is *fmi_import_xml_test*, main source file is [fmi_import_xml_test.cc](#). The test is run on three different XML files. This test depends on the success of the *ctest_fmi_import_cs_test*.

- **ctest_fmi1_capi_cs_test**

Low level test of FMI CAPI functionality for co-simulation. Test executable is *fmi1_capi_cs_test*.

- **ctest_fmi1_capi_me_test**

Low level test of FMI CAPI functionality for model exchange. Test executable is *fmi1_capi_me_test*.

- **ctest_fmi1_logger_test_run**

Run logger test generating output *fmi1_logger_test_output.txt*. Test executable is *fmi1_logger_test*.

- **ctest_fmi1_logger_test_check**

Check that the logger test output generated by **ctest_fmi1_logger_test_run** is identical with the reference file found in the **Test** subdirectory.

- **ctest_fmi2_import_xml_test_empty, ctest_fmi2_import_xml_test_me, ctest_fmi2_import_xml_test_cs, ctest_fmi2_import_xml_test_mf**

Same as **ctest_fmi_import_xml_test** for FMI 2.0 modules. Test executable is *fmi2_import_xml_test.exe*, main source file [fmi2_import_xml_test.cc](#).

- **ctest_fmi2_import_test_me**

Load a basic FMI 2.0 model exchange FMU and simulate it. Test executable is *fmi2_import_me_test*, main source file is [fmi2_import_me_test.c](#).

- **ctest_fmi2_import_test_cs** \ Load a basic FMI 2.0 co-simulation FMU and simulate it. Test executable is *fmi2_import_cs_test*, main source file is [fmi2_import_me_test.c](#).

- **ctest_fmi_import_test_no_xml, ctest_fmi_import_test_me_1, ctest_fmi_import_test_cs_1, ctest_fmi_import_test_me_2, ctest_fmi_import_test_cs_2**

Test library functionality of handling FMUs with different standard interfaces within a single application. Test executable is *fmi_import_test.exe*. Main source file is [fmi_import_test.c](#)

1.4 Using logs

In the text below we consider an FMU importing application (referenced as an 'application') that uses the FMIL. The library is designed to send log messages to a logger callback function [jm_logger_f](#) provided as a part of [jm_callbacks](#) structure in the call to

[fmi_import_allocate_context\(\)](#). The logging/error reporting functions within FMIL support following modes:

1. An importing application relies on default logging functions provided by FMIL and only chooses the log-level
 - FMIL provides default jm logger callback (See [jm_callbacks.h](#) [jm_default_logger\(\)](#))
 - FMIL provides default fmi logger callback for each standard version. - See [fmi1_import_convenience.h](#) [fmi1_default_callback_logger\(\)](#) and [fmi2_import_convenience.h](#) [fmi2_default_callback_logger\(\)](#).
2. An application provides a callback for reporting error messages according to [jm_callbacks.h](#) ([jm_logger_f](#))
 - Only errors from FMIL can be propagated to this callback in a thread-safe manner for FMI1. Imported FMUs may still use the default callback provided by the library. This is since logger function in FMI 1.0 standard is context independent. The non-thread safe implementation on the FMI1 callback can be found in [fmi1_import_convenience.h](#) [fmi1_log_forwarding\(\)](#).
In FMI 2 the `fmiComponentEnvironment` can be utilized to forward messages from the default fmi2 logger function as provided by the library to the user defined [jm_logger_f](#) function. The [fmi2_log_forwarding\(\)](#) function expects the context to be set to [fmi2_import_t](#).
3. An importing application provides logging function according to the specific FMI standard version.
 - Errors from FMIL may be forwarded to this callback. There is a function that translates calls according to [jm_callbacks.h](#) [jm_logger_f](#) into calls according to [fmi_functions.h](#) [fmi_callbacks_logger_ft](#) (see [fmi1_import_init_logger\(\)](#) or [fmi2_import_init_logger\(\)](#) that are used to setup 'forwarding').
Setting of the callback can only be done at the stage where FMU standard is known.
4. An importing application may choose not to use logging function but rely on return codes and [jm_get_last_error\(\)](#)
 - [jm_logger](#) function should be set to NULL
 - Errors from a FMI 1.0 `fmu1` cannot be handled this way in a thread-safe way (see point 2 above). It works fine with FMI 2.0.

1.5 License

Copyright

(C) 2012 Modelon AB

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Modelon AB nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL MODELON AB BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.6 Acknowledgements

The FMI Library utilizes contains code from third party tools and packages. The respective copyright information and licenses are listed with URL references to the full texts.

- **eXPat**

The parsing of the file modelDescription.xml is using eXpat.

Homepage: <<http://www.libexpat.org/>>

MIT License: <<http://www.jclark.com/xml/copying.txt>>

- **Minizip**

The unzipping of the FMU is performed partially by using Minizip library.

Homepage: <<http://www.winimage.com/zLibDll/minizip.html>>.

Condition of use and distribution are the same as Zlib <http://www.zlib.net/zlib_license.html>.

- **Zlib**

The unzipping is performed by using Zlib library called via *minizip* library.

Homepage: <<http://www.zlib.net/>>.

Zlib License: <http://www.zlib.net/zlib_license.html>.

- **C99** snprintf

C99 snprintf library is used to realize the snprintf functionality staying with C89 code.

Homepage: <http://www.jhweiss.de/software/snprintf.html>

The code is free to use. See notice in the COPYING file included with the code.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

FMI import library	19
Library initialization	102
FMI 1.0 import interface	28
Handling of variable lists	29
Constuction, destruction and error handling	107
General information retrieval	109
Interface to the standard FMI 1.0 "C" API	116
FMI 1.0 Constructor and Destructor	120
FMI 1.0 (ME) Model Exchange functions	122
FMI 1.0 (CS) Co-Simulation functions	129
FMI 1.0 (ME & CS) Common functions	137
Functions to retrieve capability flags.	117
Convenience functions.	142
Functions to retrieve co-simulation related information.	145
Support for processing variable types	147
Functions for handling unit definitions.	154
Functions for handling variable definitions.	158
Basic support for vendor annotations.	167
FMI 2.0 import interface	33
Handling of variable lists	34
Constuction, destruction and error handling	170
General information retrieval	172
Interface to the standard FMI 2.0 "C" API	186
FMI 2.0 Constructor and Destructor	187
FMI 2.0 (ME) Model Exchange functions	189
FMI 2.0 (CS) Co-Simulation functions	194
FMI 2.0 (ME & CS) Common functions	199
Convenience functions.	211
Support for processing variable types	214

Functions for handling unit definitions.	221
Functions for handling variable definitions.	226
Utility functions supporting interactions with the library	105
Utilities	20
A basic stack	24
Functions and types supporting FMI 1.0 processing.	38
Enum types used with FMI 1.0 libs	48
Functions and types supporting FMI 2.0 processing.	53
Enum types used with FMI 2.0 libs	64
Definition of XML callbacks struct	74
Definition of callbacks struct and supporting functions	77
Named objects	84
Handling platform specific defines and functions	87
A set of strings	91
A vector of items (dynamic array)	93

Chapter 3

Directory Hierarchy

3.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

build	237
doc	237
src	241
Import	240
include	241
FMI	238
FMI1	239
FMI2	240
Util	242
include	241
FMI	237
FMI1	238
FMI2	239
JM	241
Test	242
FMI1	238
fmu_dummy	240
FMI2	239
fmu_dummy	240

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

component_t	243
fmi1_callback_functions_t	249
fmi1_event_info_t	250
fmi1_import_model_counts_t	
Collection of counters providing model information	251
fmi1_me_callback_functions_t	254
fmi2_callback_functions_t	255
fmi2_event_info_t	256
fmi2_import_model_counts_t	
Collection of counters providing model information	257
fmi2_xml_callbacks_t	
XML callbacks are used to process parts of XML that are not handled by the library	261
fmul_t	261
jm_callbacks	
The callbacks struct is sent to all the modules in the library	262
jm_name_ID_map_t	
Mapping between a string and an integer ID	264
jm_named_ptr	
Name and object pointer pair	265

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

build/doc/fmilib.h	
Include file to be used in client applications of the FMI Library	267
build/doc/fmilib_config.h	
Library configuration file generated by the build system	267
build/doc/fmilib_mainpage.h	
Autogenerated file with documentation	269
src/Import/include/FMI/fmi_import_context.h	
Import context is the entry point to the library. It is used to initialize, unzip, get FMI version and start parsing	270
src/Import/include/FMI/fmi_import_util.h	271
src/Import/include/FMI1/fmi1_import.h	
Public interface to the FMI import C-library	271
src/Import/include/FMI1/fmi1_import_capabilities.h	273
src/Import/include/FMI1/fmi1_import_capi.h	275
src/Import/include/FMI1/fmi1_import_convenience.h	
Public interface to the FMI import C-library. Convenience functions .	278
src/Import/include/FMI1/fmi1_import_cosim.h	279
src/Import/include/FMI1/fmi1_import_type.h	
Public interface to the FMI XML C-library: variable types handling . .	279
src/Import/include/FMI1/fmi1_import_unit.h	
Public interface to the FMI import C-library. Handling of variable units	282
src/Import/include/FMI1/fmi1_import_variable.h	
Public interface to the FMI import C-library. Handling of model vari- ables	283
src/Import/include/FMI1/fmi1_import_variable_list.h	
Public interface to the FMI XML C-library. Handling of variable lists .	286
src/Import/include/FMI1/fmi1_import_vendor_annotations.h	
Public interface to the FMI XML C-library. Handling of vendor anno- tations	287

src/Import/include/FMI2/fmi2_import.h	
Public interface to the FMI import C-library	288
src/Import/include/FMI2/fmi2_import_capi.h	292
src/Import/include/FMI2/fmi2_import_convenience.h	
Public interface to the FMI import C-library. Convenience functions	296
src/Import/include/FMI2/fmi2_import_type.h	
Public interface to the FMI XML C-library: variable types handling	297
src/Import/include/FMI2/fmi2_import_unit.h	
Public interface to the FMI import C-library. Handling of variable units	299
src/Import/include/FMI2/fmi2_import_variable.h	
Public interface to the FMI import C-library. Handling of model variables	301
src/Import/include/FMI2/fmi2_import_variable_list.h	
Public interface to the FMI XML C-library. Handling of variable lists	304
src/Util/include/FMI/fmi_util.h	
Some low-level utility functions suitable for all standards	305
src/Util/include/FMI/fmi_version.h	
Enum defining supported FMI versions	306
src/Util/include/FMI1/fmi1_enums.h	
Definitions the enum types used with FMI 1.0 libs	306
src/Util/include/FMI1/fmi1_functions.h	308
src/Util/include/FMI1/fmi1_types.h	310
src/Util/include/FMI2/fmi2_enums.h	
Definitions the enum types used with FMI 2.0 libs	311
src/Util/include/FMI2/fmi2_functions.h	314
src/Util/include/FMI2/fmi2_types.h	316
src/Util/include/FMI2/fmi2_xml_callbacks.h	317
src/Util/include/JM/jm_callbacks.h	318
src/Util/include/JM/jm_named_ptr.h	320
src/Util/include/JM/jm_portability.h	321
src/Util/include/JM/jm_stack.h	323
src/Util/include/JM/jm_string_set.h	323
src/Util/include/JM/jm_types.h	324
src/Util/include/JM/jm_vector.h	325
src/Util/include/JM/jm_vector_template.h	
Vector template definition	327
Test/compress_test_fmu_zip.c	328
Test/fmi_import_test.c	409
Test/fmi_zip_unzip_test.c	355
Test/fmi_zip_zip_test.c	356
Test/jm_vector_test.c	382
Test/FMI1/compress_test_fmu_zip.c	329
Test/FMI1/fmi1_capi_cs_test.c	329
Test/FMI1/fmi1_capi_me_test.c	336
Test/FMI1/fmi1_import_test.c	342
Test/FMI1/fmi1_logger_test.c	343
Test/FMI1/fmi2_import_xml_test.cc	344
Test/FMI1/fmi_import_cs_test.c	349
Test/FMI1/fmi_import_me_test.c	350
Test/FMI1/fmi_import_xml_test.cc	351

Test/FMI1/fmi_total_test.c	353
Test/FMI1/fmi_zip_unzip_test.c	354
Test/FMI1/fmi_zip_zip_test.c	355
Test/FMI1/jm_vector_test.c	380
Test/FMI1/fmu_dummy/fmu1_model.c	357
Test/FMI1/fmu_dummy/fmu1_model.h	364
Test/FMI1/fmu_dummy/fmu1_model_cs.c	368
Test/FMI1/fmu_dummy/fmu1_model_defines.h	373
Test/FMI1/fmu_dummy/fmu1_model_me.c	376
Test/FMI2/fmi2_import_cs_test.c	383
Test/FMI2/fmi2_import_me_test.c	384
Test/FMI2/fmi2_import_test.c	385
Test/FMI2/fmi2_import_xml_test.cc	347
Test/FMI2/fmu_dummy/fmu2_model.c	386
Test/FMI2/fmu_dummy/fmu2_model.h	393
Test/FMI2/fmu_dummy/fmu2_model_cs.c	398
Test/FMI2/fmu_dummy/fmu2_model_defines.h	402
Test/FMI2/fmu_dummy/fmu2_model_me.c	404

Chapter 6

Module Documentation

6.1 FMI import library

Modules

- [Library initialization](#)
- [FMI 1.0 import interface](#)
- [FMI 2.0 import interface](#)
- [Utility functions supporting interactions with the library](#)

6.2 Utilities

Data Structures

- struct [jm_name_ID_map_t](#)
Mapping between a string and an integer ID.

Modules

- [A basic stack](#)
- [Functions and types supporting FMI 1.0 processing.](#)
- [Functions and types supporting FMI 2.0 processing.](#)
- [Definition of callbacks struct and supporting functions](#)
- [Named objects](#)
- [Handling platform specific defines and functions](#)
- [A set of strings](#)
- [A vector of items \(dynamic array\)](#)

Typedefs

- typedef const [char](#) * [jm_string](#)
A constant string.
- typedef void * [jm_voidp](#)
A void pointer.
- typedef struct [jm_name_ID_map_t](#) [jm_name_ID_map_t](#)
Mapping between a string and an integer ID.

Enumerations

- enum [fmi_version_enumeration_t](#) { [fmi_version_unknown_enumeration](#) = 0, [fmi_version_1_enumeration](#), [fmi_version_2_0_enumeration](#), [fmi_version_unsupported_enumeration](#) }
Supported versions of FMI standard.
- enum [jm_status_enumeration_t](#) { [jm_status_error](#) = -1, [jm_status_success](#) = 0, [jm_status_warning](#) = 1 }
Return status codes.
- enum [jm_log_level_enumeration_t](#) { [jm_log_level_nothing](#) = 0, [jm_log_level_fatal](#), [jm_log_level_error](#), [jm_log_level_warning](#), [jm_log_level_info](#), [jm_log_level_verbose](#), [jm_log_level_debug](#), [jm_log_level_all](#) }
Log levels supported via the logger functions in [jm_callbacks](#).

Functions

- **FMLIB_EXPORT** char * **fmi_construct_dll_dir_name** (jm_callbacks *callbacks, const char *fmu_unzipped_path)
Given directory name fmu_unzipped_path and construct the directory path for Dll/so.
- **FMLIB_EXPORT** char * **fmi_construct_dll_file_name** (jm_callbacks *callbacks, const char *dll_dir_name, const char *model_identifier)
Given model_identifier construct the dll/so name by adding platform suffix.
- **FMLIB_EXPORT** const char * **fmi_version_to_string** (fmi_version_enumeration_t v)
- **FMLIB_EXPORT** const char * **jm_log_level_to_string** (jm_log_level_enumeration_t level)
Convert log level into a string.

6.2.1 Typedef Documentation

6.2.1.1 typedef const char* jm_string

A constant string.

Definition at line 33 of file jm_types.h.

6.2.1.2 typedef void* jm_voidp

A void pointer.

Definition at line 35 of file jm_types.h.

6.2.1.3 typedef struct jm_name_ID_map_t jm_name_ID_map_t

Mapping between a string and an integer ID.

6.2.2 Enumeration Type Documentation

6.2.2.1 enum fmi_version_enumeration_t

Supported versions of FMI standard.

Enumerator:

fmi_version_unknown_enumeration
fmi_version_1_enumeration
fmi_version_2_0_enumeration
fmi_version_unsupported_enumeration

Definition at line 34 of file fmi_version.h.

6.2.2.2 enum jm_status_enu_t

Return status codes.

Enumerator:

jm_status_error
jm_status_success
jm_status_warning

Definition at line 44 of file jm_types.h.

6.2.2.3 enum jm_log_level_enu_t

Log levels supported via the logger functions in [jm_callbacks](#).

Enumerator:

jm_log_level_nothing
jm_log_level_fatal Must be first in this enum. May be usefull in application relying solely on [jm_get_last_error\(\)](#)
jm_log_level_error Unrecoverable errors.
jm_log_level_warning Errors that may be not critical for some FMUs.
jm_log_level_info Non-critical issues.
jm_log_level_verbose Informative messages.
jm_log_level_debug Verbose messages.
jm_log_level_all Debug messages. Only enabled if library is configured with FMILIB_ENABLE_LOG_LEVEL_DEBUG. Must be last in this enum.

Definition at line 51 of file jm_types.h.

6.2.3 Function Documentation

6.2.3.1 FMILIB_EXPORT char* fmi_construct_dll_dir_name (jm_callbacks * callbacks, const char * fmu_unzipped_path)

Given directory name fmu_unzipped_path and construct the directory path for Dll/so.

Parameters

<i>fmu_unzipped_path</i>	Directory name where FMU is unpacked.
<i>callbacks</i>	Callbacks for memory allocation.

Returns

Pointer to a string with the directory name (last symbol is directory separator). -
Caller is responsible for freeing the memory.

6.2.3.2 FMILIB_EXPORT char* fmi_construct_dll_file_name (jm_callbacks *
callbacks, const char * dll_dir_name, const char * model_identifier)

Given model_identifier construct the dll/so name by adding platform suffix.

Parameters

<i>callbacks</i>	Callbacks for memory allocation.
<i>dll_dir_name</i>	Directory path for Dll/so as returned by fmi_construct_dll_dir_name
<i>model_identifier</i>	The FMU model identifier.

Returns

Pointer to a string with the file name. Caller is responsible for freeing the memory.

6.2.3.3 FMILIB_EXPORT const char* fmi_version_to_string (fmi_version_enu_t v
)

Convert version enum into string

6.2.3.4 FMILIB_EXPORT const char* jm_log_level_to_string (jm_log_level_enu_t
level)

Convert log level into a string.

6.3 A basic stack

Defines

- `#define jm_stack(T) jm_mangle(jm_stack, T)`
A basic stack of items.
- `#define jm_stack_alloc(T) jm_mangle(jm_stack_alloc, T)`
Allocates a stack with the given reserved memory.
- `#define jm_stack_free(T) jm_mangle(jm_stack_free, T)`
Release memory allocated for a stack.
- `#define jm_stack_init(T) jm_mangle(jm_stack_init, T)`
*Initializes a *jm_stack* allocated on stack.*
- `#define jm_stack_free_data(T) jm_mangle(jm_stack_free_data, T)`
Releases memory allocated for stack data.
- `#define jm_stack_get_size(T) jm_mangle(jm_stack_get_size, T)`
Get the number of elements in the stack.
- `#define jm_stack_reserve(T) jm_mangle(jm_stack_reserve, T)`
Preallocate memory for the stack (to speed up consequent push).
- `#define jm_stack_push(T) jm_mangle(jm_stack_push, T)`
Put an element on the stack.
- `#define jm_stack_is_empty(T) jm_mangle(jm_stack_is_empty, T)`
- `#define jm_stack_pop(T) jm_mangle(jm_stack_pop, T)`
- `#define jm_stack_top(T) jm_mangle(jm_stack_top, T)`
- `#define jm_stack_foreach(T) jm_mangle(jm_stack_foreach, T)`
- `#define JM_STACK_MINIMAL_CAPACITY JM_VECTOR_MINIMAL_CAPACITY`
- `#define JM_STACK_MAX_MEMORY_CHUNK JM_VECTOR_MAX_MEMORY_CHUNK`
- `#define jm_stack_declare_template(T)`

6.3.1 Define Documentation

6.3.1.1 `#define jm_stack( T ) jm_mangle(jm_stack, T)`

A basic stack of items.

Stack is implemented on top of `jm_vector` right now. There is a couple of extra methonds that are convenient.

Definition at line 41 of file `jm_stack.h`.

6.3.1.2 `#define jm_stack_alloc( T ) jm_mangle(jm_stack_alloc, T)`

Allocates a stack with the given reserved memory.

```
jm_stack(T) * jm_stack_alloc(T) (size_t capacity, jm_callbacks*c );
```

Parameters

<i>capacity</i>	- initial stack capacity, can be 0
<i>c</i>	- jm_callbacks callbacks, can be zero

Returns

Newly allocated stack

Definition at line 52 of file `jm_stack.h`.

6.3.1.3 `#define jm_stack_free( T ) jm_mangle(jm_stack_free, T)`

Release memory allocated for a stack.

```
extern void jm_stack_free(T) (jm_stack(T) * a);
```

Definition at line 60 of file `jm_stack.h`.

6.3.1.4 `#define jm_stack_init( T ) jm_mangle(jm_stack_init, T)`

Initializes a [jm_stack](#) allocated on stack.

Parameters

<i>a</i>	- pointer to the stack to be initialized;
<i>c</i>	- jm_callbacks callbacks, can be zero

```
void jm_stack_init(T) (jm_stack(T) * a, jm_callbacks* c)
```

Definition at line 71 of file `jm_stack.h`.

6.3.1.5 `#define jm_stack_free_data( T ) jm_mangle(jm_stack_free_data, T)`

Releases memory allocated for stack data.

This only needs to be called both for stack allocated [jm_stack](#) structs.

```
inline void jm_stack_free_data(T) (jm_stack(T) * a)
```

Parameters

<i>a</i>	- pointer to the stack.
----------	-------------------------

Definition at line 83 of file `jm_stack.h`.

6.3.1.6 `#define jm_stack_get_size( T ) jm_mangle(jm_stack_get_size, T)`

Get the number of elements in the stack.

```
inline size_t jm_stack_get_size(T) (jm_stack(T) * a)
```

Definition at line 92 of file `jm_stack.h`.

6.3.1.7 `#define jm_stack_reserve( T ) jm_mangle(jm_stack_reserve, T)`

Preallocate memory for the stack (to speed up consequent push).

Returns

The actually reserved space. Can be smaller than "capacity" if memory allocation failed. Can be larger than "capacity" if more memory was previously allocated.
`size_t jm_stack_reserve(T)(jm_stack(T)* a, size_t capacity)`

Definition at line 101 of file `jm_stack.h`.

6.3.1.8 `#define jm_stack_push( T ) jm_mangle(jm_stack_push, T)`

Put an element on the stack.

Returns

A pointer to the inserted element or zero pointer if failed.

```
T* jm_stack_push_back(jm_stack(T) * a, T item);
```

Definition at line 111 of file `jm_stack.h`.

6.3.1.9 `#define jm_stack_is_empty( T ) jm_mangle(jm_stack_is_empty, T)`

`jm_stack_is_empty` returns 1 if the stack is empty and 0 otherwise. `int jm_stack_is_empty(jm_stack(T)*)`

Definition at line 117 of file `jm_stack.h`.

6.3.1.10 `#define jm_stack_pop( T ) jm_mangle(jm_stack_pop, T)`

`jm_stack_pop` gets the stack head and moves to the next element. Popping an empty stack gives assertion failure. `T jm_stack_pop(jm_stack(T)* a)`

Definition at line 123 of file `jm_stack.h`.

6.3.1.11 **#define jm_stack_top(T) jm_mangle(jm_stack_top, T)**

jm_stack_top gets the stack top. Call on an empty stack gives assertion failure. T
jm_stack_top(jm_stack(T)* a)

Definition at line 129 of file jm_stack.h.

6.3.1.12 **#define jm_stack_foreach(T) jm_mangle(jm_stack_foreach, T)**

jm_stack_foreach calls f for each element in the stack. "data" parameter is forwarded to the function as the second argument. void jm_stack_foreach(T)(jm_stack(T)* a, void (*f)(T, void*), void * data)

Definition at line 136 of file jm_stack.h.

6.3.1.13 **#define JM_STACK_MINIMAL_CAPACITY JM_VECTOR_MINIMAL_CAPACITY**

minimal number of items always allocated for the stack

Definition at line 140 of file jm_stack.h.

6.3.1.14 **#define JM_STACK_MAX_MEMORY_CHUNK JM_VECTOR_MAX_MEMORY_CHUNK**

maximum memory chunk (in items) to be allocated in push.

Definition at line 143 of file jm_stack.h.

6.3.1.15 **#define jm_stack_declare_template(T)**

Declare stack for the specific type.

Definition at line 146 of file jm_stack.h.

6.4 FMI 1.0 import interface

Modules

- [Handling of variable lists](#)

Variable lists are provided to handle sets of variables.

- [Constuction, destruction and error handling](#)
- [General information retrieval](#)

Functions for retrieving general model information. Memory for the strings is allocated and deallocated in the module. All the functions take an FMU object as returned by [fmi1_import_parse_xml\(\)](#) as a parameter. The information is retrieved from the XML file.

- [Interface to the standard FMI 1.0 "C" API](#)

Convenient functions for calling the FMI functions. This interface wrappes the "C" API.

- [Functions to retrieve capability flags.](#)
- [Convenience functions.](#)

The functions in this module are provided for convenience. The functionality is already available via other lower level functions.

- [Functions to retrieve co-simulation related information.](#)
- [Support for processing variable types](#)
- [Functions for handling unit definitions.](#)
- [Functions for handling variable definitions.](#)

All the functions in this group take a pointer to [fmi1_import_variable_t](#) as a parameter. A variable pointer may be obtained via a [Handling of variable lists](#) module or via functions [fmi1_import_get_variable_by_name\(\)](#) and [fmi1_import_get_variable_by_vr\(\)](#).

- [Basic support for vendor annotations.](#)

6.4.1 Detailed Description

All the structures used in the interfaces are intended to be treated as opaque objects by the client code.

6.5 Handling of variable lists

Variable lists are provided to handle sets of variables.

Functions

- `fmi1_import_variable_list_t * fmi1_import_alloc_variable_list (fmi1_import_t *fmu, size_t size)`
- `FMILIB_EXPORT void fmi1_import_free_variable_list (fmi1_import_variable_list_t *vl)`
Free a variable list. Note that variable lists are allocated dynamically and must be freed when not needed any longer.
- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_clone_variable_list (fmi1_import_variable_list_t *vl)`
Make a copy of the list.
- `FMILIB_EXPORT size_t fmi1_import_get_variable_list_size (fmi1_import_variable_list_t *vl)`
Get number of variables in a list.
- `FMILIB_EXPORT const fmi1_value_reference_t * fmi1_import_get_value_reference_list (fmi1_import_variable_list_t *vl)`
Get a pointer to the list of the value references for all the variables.
- `FMILIB_EXPORT fmi1_import_variable_t * fmi1_import_get_variable (fmi1_import_variable_list_t *vl, unsigned int index)`
Get a single variable from the list.

Operations on variable lists. Every operation creates a new list.

- `typedef int(* fmi1_import_variable_filter_function_ft)(fmi1_import_variable_t *vl, void *data)`
Callback function typedef for the fmiFilterVariables.
- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_sublist (fmi1_import_variable_list_t *vl, unsigned int fromIndex, unsigned int toIndex)`
Select sub-lists.
- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_filter_variables (fmi1_import_variable_list_t *vl, fmi1_import_variable_filter_function_ft filter, void *context)`
Call the provided 'filter' function on every variable in the list and create a new list.
- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_join_var_list (fmi1_import_variable_list_t *a, fmi1_import_variable_list_t *b)`
Create a new variable list by concatenating two lists.
- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_append_to_var_list (fmi1_import_variable_list_t *vl, fmi1_import_variable_t *v)`
Append a variable to the variable list.
- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_prepend_to_var_list (fmi1_import_variable_list_t *vl, fmi1_import_variable_t *v)`

Prepend a variable to the variable list.

- `FMILIB_EXPORT jm_status_enu_t fmi1_import_var_list_push_back (fmi1_import_variable_list_t *vl, fmi1_import_variable_t *v)`

Add a variable to a variable list.

6.5.1 Detailed Description

Variable lists are provided to handle sets of variables. Note that variable lists are allocated dynamically and must be freed when not needed any longer.

6.5.2 Typedef Documentation

6.5.2.1 `typedef int(* fmi1_import_variable_filter_function_ft)(fmi1_import_variable_t *vl, void *data)`

Callback function typedef for the `fmiFilterVariables`.

The function should return 0 to prevent a variable from coming to the output list.

Definition at line 81 of file `fmi1_import_variable_list.h`.

6.5.3 Function Documentation

6.5.3.1 `fmi1_import_variable_list_t* fmi1_import_alloc_variable_list (fmi1_import_t *fmu, size_t size)`

6.5.3.2 `FMILIB_EXPORT void fmi1_import_free_variable_list (fmi1_import_variable_list_t * vl)`

Free a variable list. Note that variable lists are allocated dynamically and must be freed when not needed any longer.

Parameters

<code>vl</code>	A variable list.
-----------------	------------------

6.5.3.3 `FMILIB_EXPORT fmi1_import_variable_list_t* fmi1_import_clone_variable_list ( fmi1_import_variable_list_t * vl )`

Make a copy of the list.

Parameters

<code>vl</code>	A variable list.
-----------------	------------------

6.5.3.4 FMILIB_EXPORT `size_t fmi1_import_get_variable_list_size ( fmi1_import_variable_list_t * vl )`

Get number of variables in a list.

6.5.3.5 FMILIB_EXPORT `const fmi1_value_reference_t* fmi1_import_get_value_referece_list ( fmi1_import_variable_list_t * vl )`

Get a pointer to the list of the value references for all the variables.

6.5.3.6 FMILIB_EXPORT `fmi1_import_variable_t* fmi1_import_get_variable ( fmi1_import_variable_list_t * vl, unsigned int index )`

Get a single variable from the list.

6.5.3.7 FMILIB_EXPORT `fmi1_import_variable_list_t* fmi1_import_get_sublist ( fmi1_import_variable_list_t * vl, unsigned int fromIndex, unsigned int toIndex )`

Select sub-lists.

Parameters

<i>vl</i>	A variable list.
<i>fromIndex</i>	Zero based start index, inclusive.
<i>toIndex</i>	Zero based end index, inclusive.

Returns

A sublist. NULL is returned if toIndex is less than fromIndex or is larger than the list size or if memory allocation failed.

6.5.3.8 FMILIB_EXPORT `fmi1_import_variable_list_t* fmi1_import_filter_variables ( fmi1_import_variable_list_t * vl, fmi1_import_variable_filter_function_ft filter, void * context )`

Call the provided 'filter' function on every variable in the list and create a new list.

Parameters

<i>vl</i>	A variable list.
<i>filter</i>	A filter function according to fmi1_import_variable_filter_function_ft .
<i>context</i>	A parameter to be forwarded to the filter function.

Returns

A sub-list with the variables for which filter returned non-zero value.

6.5.3.9 FMILIB_EXPORT fmi1_import_variable_list_t* fmi1_import_join_var_list (fmi1_import_variable_list_t * a, fmi1_import_variable_list_t * b)

Create a new variable list by concatenating two lists.

Parameters

<i>a</i>	A variable list.
<i>b</i>	A variable list.

6.5.3.10 FMILIB_EXPORT fmi1_import_variable_list_t* fmi1_import_append_to_var_list (fmi1_import_variable_list_t * vl, fmi1_import_variable_t * v)

Append a variable to the variable list.

Parameters

<i>vl</i>	A variable list.
<i>v</i>	A variable.

6.5.3.11 FMILIB_EXPORT fmi1_import_variable_list_t* fmi1_import_prepend_to_var_list (fmi1_import_variable_list_t * vl, fmi1_import_variable_t * v)

Prepend a variable to the variable list.

Parameters

<i>vl</i>	A variable list.
<i>v</i>	A variable.

6.5.3.12 FMILIB_EXPORT jm_status_enu_t fmi1_import_var_list_push_back (fmi1_import_variable_list_t * vl, fmi1_import_variable_t * v)

Add a variable to a variable list.

Parameters

<i>vl</i>	A variable list.
<i>v</i>	A variable.

6.6 FMI 2.0 import interface

Modules

- [Handling of variable lists](#)

Variable lists are provided to handle sets of variables.

- [Constuction, destruction and error handling](#)
- [General information retrieval](#)

Functions for retrieving general model information. Memory for the strings is allocated and deallocated in the module. All the functions take an FMU object as returned by [fmi2_import_parse_xml\(\)](#) as a parameter. The information is retrieved from the XML file.

- [Interface to the standard FMI 2.0 "C" API](#)

Convenient functions for calling the FMI functions. This interface wrappes the "C" API.

- [Convenience functions.](#)

The functions in this module are provided for convenience. The functionality is already available via other lower level functions.

- [Support for processing variable types](#)
- [Functions for handling unit definitions.](#)
- [Functions for handling variable definitions.](#)

All the functions in this group take a pointer to [fmi2_import_variable_t](#) as a parameter. A variable pointer may be obtained via a [Handling of variable lists](#) module or via functions [fmi2_import_get_variable_by_name\(\)](#) and [fmi2_import_get_variable_by_vr\(\)](#).

6.6.1 Detailed Description

All the structures used in the interfaces are intended to be treated as opaque objects by the client code.

6.7 Handling of variable lists

Variable lists are provided to handle sets of variables.

Functions

- `fmi2_import_variable_list_t * fmi2_import_alloc_variable_list (fmi2_import_t *fmu, size_t size)`
Allocate an empty list.
- `FMILIB_EXPORT void fmi2_import_free_variable_list (fmi2_import_variable_list_t *vl)`
Free a variable list. Note that variable lists are allocated dynamically and must be freed when not needed any longer.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_clone_variable_list (fmi2_import_variable_list_t *vl)`
Make a copy of the list.
- `FMILIB_EXPORT size_t fmi2_import_get_variable_list_size (fmi2_import_variable_list_t *vl)`
Get number of variables in a list.
- `FMILIB_EXPORT const fmi2_value_reference_t * fmi2_import_get_value_references_list (fmi2_import_variable_list_t *vl)`
Get a pointer to the list of the value references for all the variables.
- `FMILIB_EXPORT fmi2_import_variable_t * fmi2_import_get_variable (fmi2_import_variable_list_t *vl, size_t index)`
Get a single variable from the list.

Operations on variable lists. Every operation creates a new list.

- `typedef int(* fmi2_import_variable_filter_function_ft)(fmi2_import_variable_t *vl, void *data)`
Callback function typedef for the fmiFilterVariables.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_sublist (fmi2_import_variable_list_t *vl, size_t fromIndex, size_t toIndex)`
Select sub-lists.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_filter_variables (fmi2_import_variable_list_t *vl, fmi2_import_variable_filter_function_ft filter, void *context)`
Call the provided 'filter' function on every variable in the list and create a new list.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_join_var_list (fmi2_import_variable_list_t *a, fmi2_import_variable_list_t *b)`
Create a new variable list by concatenating two lists.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_append_to_var_list (fmi2_import_variable_list_t *vl, fmi2_import_variable_t *v)`
Append a variable to the variable list.

- `FMLIB_EXPORT fmi2_import_variable_list_t * fmi2_import_prepend_to_var_list (fmi2_import_variable_list_t *vl, fmi2_import_variable_t *v)`
Prepend a variable to the variable list.
- `FMLIB_EXPORT jm_status_enu_t fmi2_import_var_list_push_back (fmi2_import_variable_list_t *vl, fmi2_import_variable_t *v)`
Add a variable to a variable list.

6.7.1 Detailed Description

Variable lists are provided to handle sets of variables. Note that variable lists are allocated dynamically and must be freed when not needed any longer.

6.7.2 Typedef Documentation

6.7.2.1 `typedef int(* fmi2_import_variable_filter_function_ft)(fmi2_import_variable_t *vl, void *data)`

Callback function typedef for the `fmiFilterVariables`.

The function should return 0 to prevent a variable from coming to the output list.

Definition at line 81 of file `fmi2_import_variable_list.h`.

6.7.3 Function Documentation

6.7.3.1 `fmi2_import_variable_list_t* fmi2_import_alloc_variable_list (fmi2_import_t * fmu, size_t size )`

Allocate an empty list.

6.7.3.2 `FMLIB_EXPORT void fmi2_import_free_variable_list (fmi2_import_variable_list_t * vl )`

Free a variable list. Note that variable lists are allocated dynamically and must be freed when not needed any longer.

Parameters

<code>vl</code>	A variable list.
-----------------	------------------

6.7.3.3 `FMLIB_EXPORT fmi2_import_variable_list_t* fmi2_import_clone_variable_list ( fmi2_import_variable_list_t * vl )`

Make a copy of the list.

Parameters

<i>vl</i>	A variable list.
-----------	------------------

6.7.3.4 FMILIB_EXPORT size_t fmi2_import_get_variable_list_size (fmi2_import_variable_list_t * vl)

Get number of variables in a list.

6.7.3.5 FMILIB_EXPORT const fmi2_value_reference_t* fmi2_import_get_value_referece_list (fmi2_import_variable_list_t * vl)

Get a pointer to the list of the value references for all the variables.

6.7.3.6 FMILIB_EXPORT fmi2_import_variable_t* fmi2_import_get_variable (fmi2_import_variable_list_t * vl, size_t index)

Get a single variable from the list.

6.7.3.7 FMILIB_EXPORT fmi2_import_variable_list_t* fmi2_import_get_sublist (fmi2_import_variable_list_t * vl, size_t fromIndex, size_t toIndex)

Select sub-lists.

Parameters

<i>vl</i>	A variable list.
<i>fromIndex</i>	Zero based start index, inclusive.
<i>toIndex</i>	Zero based end index, inclusive.

Returns

A sublist. NULL is returned if toIndex is less than fromIndex or is larger than the list size or if memory allocation failed.

6.7.3.8 FMILIB_EXPORT fmi2_import_variable_list_t* fmi2_import_filter_variables (fmi2_import_variable_list_t * vl, fmi2_import_variable_filter_function_ft filter, void * context)

Call the provided 'filter' function on every variable in the list and create a new list.

Parameters

<i>vl</i>	A variable list.
<i>filter</i>	A filter function according to fmi2_import_variable_filter_function_ft .
<i>context</i>	A parameter to be forwarded to the filter function.

Returns

A sub-list with the variables for which filter returned non-zero value.

6.7.3.9 FMILIB_EXPORT fmi2_import_variable_list_t* fmi2_import_join_var_list (fmi2_import_variable_list_t * a, fmi2_import_variable_list_t * b)

Create a new variable list by concatenating two lists.

Parameters

<i>a</i>	A variable list.
<i>b</i>	A variable list.

6.7.3.10 FMILIB_EXPORT fmi2_import_variable_list_t* fmi2_import_append_to_var_list (fmi2_import_variable_list_t * vl, fmi2_import_variable_t * v)

Append a variable to the variable list.

Parameters

<i>vl</i>	A variable list.
<i>v</i>	A variable.

6.7.3.11 FMILIB_EXPORT fmi2_import_variable_list_t* fmi2_import_prepend_to_var_list (fmi2_import_variable_list_t * vl, fmi2_import_variable_t * v)

Prepend a variable to the variable list.

Parameters

<i>vl</i>	A variable list.
<i>v</i>	A variable.

6.7.3.12 FMILIB_EXPORT jm_status_enu_t fmi2_import_var_list_push_back (fmi2_import_variable_list_t * vl, fmi2_import_variable_t * v)

Add a variable to a variable list.

Parameters

<i>vl</i>	A variable list.
<i>v</i>	A variable.

6.8 Functions and types supporting FMI 1.0 processing.

Data Structures

- struct [fmi1_me_callback_functions_t](#)
- struct [fmi1_callback_functions_t](#)
- struct [fmi1_event_info_t](#)

Modules

- [Enum types used with FMI 1.0 libs](#)

Typedefs

- typedef void(* [fmi1_callback_logger_ft](#))(fmi1_component_t c, fmi1_string_t instanceName, [fmi1_status_t](#) status, fmi1_string_t category, fmi1_string_t message,...)
- typedef void (* [fmi1_callback_allocate_memory_ft](#))(size_t nobj, size_t size)
- typedef void(* [fmi1_callback_free_memory_ft](#))(void *obj)
- typedef void(* [fmi1_step_finished_ft](#))(fmi1_component_t c, [fmi1_status_t](#) status)
- typedef const char *(* [fmi1_get_version_ft](#))(void)
- typedef [fmi1_status_t](#)(* [fmi1_set_debug_logging_ft](#))(fmi1_component_t c, fmi1_boolean_t loggingOn)
- typedef [fmi1_status_t](#)(* [fmi1_set_real_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_real_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_set_integer_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_set_boolean_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_boolean_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_set_string_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_string_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_real_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, fmi1_real_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_integer_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, fmi1_integer_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_boolean_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, fmi1_boolean_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_string_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, fmi1_string_t value[])
- typedef const char *(* [fmi1_get_model_typesPlatform_ft](#))(void)
- typedef fmi1_component_t(* [fmi1_instantiate_model_ft](#))(fmi1_string_t instanceName, fmi1_string_t GUID, [fmi1_me_callback_functions_t](#) functions, fmi1_boolean_t loggingOn)
- typedef void(* [fmi1_free_model_instance_ft](#))(fmi1_component_t c)
- typedef [fmi1_status_t](#)(* [fmi1_set_time_ft](#))(fmi1_component_t c, fmi1_real_t time)

- typedef [fmi1_status_t](#)(* [fmi1_set_continuous_states_ft](#))(fmi1_component_t c, const fmi1_real_t x[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_completed_integrator_step_ft](#))(fmi1_component_t c, fmi1_boolean_t *callEventUpdate)
- typedef [fmi1_status_t](#)(* [fmi1_initialize_ft](#))(fmi1_component_t c, fmi1_boolean_t toleranceControlled, fmi1_real_t relativeTolerance, [fmi1_event_info_t](#) *eventInfo)
- typedef [fmi1_status_t](#)(* [fmi1_get_derivatives_ft](#))(fmi1_component_t c, fmi1_real_t derivatives[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_get_event_indicators_ft](#))(fmi1_component_t c, fmi1_real_t eventIndicators[], [size_t](#) ni)
- typedef [fmi1_status_t](#)(* [fmi1_event_update_ft](#))(fmi1_component_t c, fmi1_boolean_t intermediateResults, [fmi1_event_info_t](#) *eventInfo)
- typedef [fmi1_status_t](#)(* [fmi1_get_continuous_states_ft](#))(fmi1_component_t c, fmi1_real_t states[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_get_nominal_continuousStates_ft](#))(fmi1_component_t c, fmi1_real_t x_nominal[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_get_state_valueReferences_ft](#))(fmi1_component_t c, fmi1_value_reference_t vrx[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_terminate_ft](#))(fmi1_component_t c)
- typedef const [char](#) *(* [fmi1_get_types_platform_ft](#))(void)
- typedef fmi1_component_t(* [fmi1_instantiate_slave_ft](#))(fmi1_string_t instanceName, fmi1_string_t fmuGUID, fmi1_string_t fmuLocation, fmi1_string_t mimeType, fmi1_real_t timeout, fmi1_boolean_t visible, fmi1_boolean_t interactive, [fmi1_callback_functions_t](#) functions, fmi1_boolean_t loggingOn)
- typedef [fmi1_status_t](#)(* [fmi1_initialize_slave_ft](#))(fmi1_component_t c, fmi1_real_t tStart, fmi1_boolean_t StopTimeDefined, fmi1_real_t tStop)
- typedef [fmi1_status_t](#)(* [fmi1_terminate_slave_ft](#))(fmi1_component_t c)
- typedef [fmi1_status_t](#)(* [fmi1_reset_slave_ft](#))(fmi1_component_t c)
- typedef void(* [fmi1_free_slave_instance_ft](#))(fmi1_component_t c)
- typedef [fmi1_status_t](#)(* [fmi1_set_real_inputDerivatives_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], [size_t](#) nvr, const fmi1_integer_t order[], const fmi1_real_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_real_outputDerivatives_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], [size_t](#) nvr, const fmi1_integer_t order[], fmi1_real_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_cancel_step_ft](#))(fmi1_component_t c)
- typedef [fmi1_status_t](#)(* [fmi1_do_step_ft](#))(fmi1_component_t c, fmi1_real_t currentCommunicationPoint, fmi1_real_t communicationStepSize, fmi1_boolean_t newStep)
- typedef [fmi1_status_t](#)(* [fmi1_get_status_ft](#))(fmi1_component_t c, const [fmi1_status_kind_t](#) s, [fmi1_status_t](#) *value)
- typedef [fmi1_status_t](#)(* [fmi1_get_real_status_ft](#))(fmi1_component_t c, const [fmi1_status_kind_t](#) s, fmi1_real_t *value)
- typedef [fmi1_status_t](#)(* [fmi1_get_integer_status_ft](#))(fmi1_component_t c, const [fmi1_status_kind_t](#) s, fmi1_integer_t *value)
- typedef [fmi1_status_t](#)(* [fmi1_get_boolean_status_ft](#))(fmi1_component_t c, const [fmi1_status_kind_t](#) s, fmi1_boolean_t *value)

- typedef `fmi1_status_t`(* `fmi1_get_string_status_ft`)(`fmi1_component_t` c, const `fmi1_status_kind_t` s, `fmi1_string_t` *value)
- typedef enum `fmi1_value_reference_enu_t` `fmi1_value_reference_enu_t`

Enumerations

- enum `fmi1_status_t` { `fmi1_status_ok`, `fmi1_status_warning`, `fmi1_status_discard`, `fmi1_status_error`, `fmi1_status_fatal`, `fmi1_status_pending` }
- enum `fmi1_status_kind_t` { `fmi1_do_step_status`, `fmi1_pending_status`, `fmi1_last_successful_time` }
- enum `fmi1_boolean_enu_t` { `fmi1_true` = `fmiTrue`, `fmi1_false` = `fmiFalse` }
- enum `fmi1_value_reference_enu_t` { `fmi1_undefined_value_reference` = (int)`fmiUndefinedValueReference` }

Functions

- `FMLIB_EXPORT` const char * `fmi1_status_to_string` (`fmi1_status_t` status)
- static const char * `fmi1_get_platform` (void)

Renaming of typedefs

- #define `fmiComponent` `fmi1_component_t`
- #define `fmiValueReference` `fmi1_value_reference_t`
- #define `fmiReal` `fmi1_real_t`
- #define `fmiInteger` `fmi1_integer_t`
- #define `fmiBoolean` `fmi1_boolean_t`
- #define `fmiString` `fmi1_string_t`

6.8.1 Define Documentation

6.8.1.1 #define `fmiComponent` `fmi1_component_t`

Definition at line 33 of file `fmi1_types.h`.

6.8.1.2 #define `fmiValueReference` `fmi1_value_reference_t`

Definition at line 34 of file `fmi1_types.h`.

6.8.1.3 #define `fmiReal` `fmi1_real_t`

Definition at line 35 of file `fmi1_types.h`.

6.8.1.4 `#define fmiInteger fmi1_integer_t`

Definition at line 36 of file fmi1_types.h.

6.8.1.5 `#define fmiBoolean fmi1_boolean_t`

Definition at line 37 of file fmi1_types.h.

6.8.1.6 `#define fmiString fmi1_string_t`

Definition at line 38 of file fmi1_types.h.

6.8.2 Typedef Documentation

6.8.2.1 `typedef void(* fmi1_callback_logger_ft)(fmi1_component_t c, fmi1_string_t
instanceName, fmi1_status_t status, fmi1_string_t category, fmi1_string_t message,...)`

FMI 1.0 logger function type

Definition at line 44 of file fmi1_functions.h.

6.8.2.2 `typedef void(* fmi1_callback_allocate_memory_ft)(size_t nobj, size_t size)`

FMI 1.0 allocate memory function type

Definition at line 46 of file fmi1_functions.h.

6.8.2.3 `typedef void(* fmi1_callback_free_memory_ft)(void *obj)`

FMI 1.0 free memory function type

Definition at line 48 of file fmi1_functions.h.

6.8.2.4 `typedef void(* fmi1_step_finished_ft)(fmi1_component_t c, fmi1_status_t status)`

FMI 1.0 step finished callback function type

Definition at line 50 of file fmi1_functions.h.

6.8.2.5 `typedef const char(* fmi1_get_version_ft)(void)`

Definition at line 85 of file fmi1_functions.h.

6.8.2.6 `typedef fmi1_status_t(* fmi1_set_debug_logging_ft)(fmi1_component_t c, fmi1_boolean_t loggingOn)`

Definition at line 86 of file fmi1_functions.h.

6.8.2.7 `typedef fmi1_status_t(* fmi1_set_real_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_real_t value[])`

Definition at line 87 of file fmi1_functions.h.

6.8.2.8 `typedef fmi1_status_t(* fmi1_set_integer_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t value[])`

Definition at line 88 of file fmi1_functions.h.

6.8.2.9 `typedef fmi1_status_t(* fmi1_set_boolean_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_boolean_t value[])`

Definition at line 89 of file fmi1_functions.h.

6.8.2.10 `typedef fmi1_status_t(* fmi1_set_string_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_string_t value[])`

Definition at line 90 of file fmi1_functions.h.

6.8.2.11 `typedef fmi1_status_t(* fmi1_get_real_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, fmi1_real_t value[])`

Definition at line 91 of file fmi1_functions.h.

6.8.2.12 `typedef fmi1_status_t(* fmi1_get_integer_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, fmi1_integer_t value[])`

Definition at line 92 of file fmi1_functions.h.

6.8.2.13 `typedef fmi1_status_t(* fmi1_get_boolean_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, fmi1_boolean_t value[])`

Definition at line 93 of file fmi1_functions.h.

6.8.2.14 `typedef fmi1_status_t(* fmi1_get_string_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, fmi1_string_t value[])`

Definition at line 94 of file fmi1_functions.h.

6.8.2.15 `typedef const char*(* fmi1_get_model_typesPlatform_ft)(void)`

Definition at line 97 of file fmi1_functions.h.

6.8.2.16 `typedef fmi1_component_t(* fmi1_instantiate_model_ft)(fmi1_string_t
instanceName, fmi1_string_t GUID, fmi1_me_callback_functions_t functions,
fmi1_boolean_t loggingOn)`

Definition at line 98 of file fmi1_functions.h.

6.8.2.17 `typedef void(* fmi1_free_model_instance_ft)(fmi1_component_t c)`

Definition at line 99 of file fmi1_functions.h.

6.8.2.18 `typedef fmi1_status_t(* fmi1_set_time_ft)(fmi1_component_t c, fmi1_real_t time)`

Definition at line 100 of file fmi1_functions.h.

6.8.2.19 `typedef fmi1_status_t(* fmi1_set_continuous_states_ft)(fmi1_component_t c,
const fmi1_real_t x[], size_t nx)`

Definition at line 101 of file fmi1_functions.h.

6.8.2.20 `typedef fmi1_status_t(* fmi1_completed_integrator_step_ft)(fmi1_
component_t c, fmi1_boolean_t *callEventUpdate)`

Definition at line 102 of file fmi1_functions.h.

6.8.2.21 `typedef fmi1_status_t(* fmi1_initialize_ft)(fmi1_component_t c, fmi1_boolean_t
toleranceControlled, fmi1_real_t relativeTolerance, fmi1_event_info_t *eventInfo)`

Definition at line 103 of file fmi1_functions.h.

6.8.2.22 `typedef fmi1_status_t(* fmi1_get_derivatives_ft)(fmi1_component_t c,
fmi1_real_t derivatives[], size_t nx)`

Definition at line 104 of file fmi1_functions.h.

6.8.2.23 `typedef fmi1_status_t(* fmi1_get_event_indicators_ft)(fmi1_component_t c,
fmi1_real_t eventIndicators[], size_t ni)`

Definition at line 105 of file fmi1_functions.h.

6.8.2.24 `typedef fmi1_status_t(* fmi1_event_update_ft)(fmi1_component_t c,
fmi1_boolean_t intermediateResults, fmi1_event_info_t *eventInfo)`

Definition at line 106 of file fmi1_functions.h.

6.8.2.25 `typedef fmi1_status_t(* fmi1_get_continuous_states_ft)(fmi1_component_t c,
fmi1_real_t states[], size_t nx)`

Definition at line 107 of file fmi1_functions.h.

6.8.2.26 `typedef fmi1_status_t(* fmi1_get_nominal_continuous-
States_ft)(fmi1_component_t c, fmi1_real_t x_nominal[], size_t
nx)`

Definition at line 108 of file fmi1_functions.h.

6.8.2.27 `typedef fmi1_status_t(* fmi1_get_state_valueReferences-
_ft)(fmi1_component_t c, fmi1_value_reference_t vrx[], size_t
nx)`

Definition at line 109 of file fmi1_functions.h.

6.8.2.28 `typedef fmi1_status_t(* fmi1_terminate_ft)(fmi1_component_t c)`

Definition at line 110 of file fmi1_functions.h.

6.8.2.29 `typedef const char*( * fmi1_get_types_platform_ft)(void)`

Definition at line 114 of file fmi1_functions.h.

6.8.2.30 `typedef fmi1_component_t(* fmi1_instantiate_slave_ft)(fmi1_string_t
instanceName, fmi1_string_t fmuGUID, fmi1_string_t fmuLocation, fmi1_string_t
mimeType, fmi1_real_t timeout, fmi1_boolean_t visible, fmi1_boolean_t interactive,
fmi1_callback_functions_t functions, fmi1_boolean_t loggingOn)`

Definition at line 115 of file fmi1_functions.h.

6.8.2.31 `typedef fmi1_status_t(* fmi1_initialize_slave_ft)(fmi1_component_t c, fmi1_real_t
tStart, fmi1_boolean_t StopTimeDefined, fmi1_real_t tStop)`

Definition at line 118 of file fmi1_functions.h.

6.8.2.32 `typedef fmi1_status_t(* fmi1_terminate_slave_ft)(fmi1_component_t c)`

Definition at line 119 of file `fmi1_functions.h`.

6.8.2.33 `typedef fmi1_status_t(* fmi1_reset_slave_ft)(fmi1_component_t c)`

Definition at line 120 of file `fmi1_functions.h`.

6.8.2.34 `typedef void(* fmi1_free_slave_instance_ft)(fmi1_component_t c)`

Definition at line 121 of file `fmi1_functions.h`.

6.8.2.35 `typedef fmi1_status_t(* fmi1_set_real_inputDerivatives_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t order[], const fmi1_real_t value[])`

Definition at line 122 of file `fmi1_functions.h`.

6.8.2.36 `typedef fmi1_status_t(* fmi1_get_real_outputDerivatives_ft)(fmi1_component_t c, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t order[], fmi1_real_t value[])`

Definition at line 123 of file `fmi1_functions.h`.

6.8.2.37 `typedef fmi1_status_t(* fmi1_cancel_step_ft)(fmi1_component_t c)`

Definition at line 124 of file `fmi1_functions.h`.

6.8.2.38 `typedef fmi1_status_t(* fmi1_do_step_ft)(fmi1_component_t c, fmi1_real_t currentCommunicationPoint, fmi1_real_t communicationStepSize, fmi1_boolean_t newStep)`

Definition at line 125 of file `fmi1_functions.h`.

6.8.2.39 `typedef fmi1_status_t(* fmi1_get_status_ft)(fmi1_component_t c, const fmi1_status_kind_t s, fmi1_status_t *value)`

Definition at line 127 of file `fmi1_functions.h`.

6.8.2.40 `typedef fmi1_status_t(* fmi1_get_real_status_ft)(fmi1_component_t c, const fmi1_status_kind_t s, fmi1_real_t *value)`

Definition at line 128 of file `fmi1_functions.h`.

6.8.2.41 `typedef fmi1_status_t(* fmi1_get_integer_status_ft)(fmi1_component_t c, const fmi1_status_kind_t s, fmi1_integer_t *value)`

Definition at line 129 of file fmi1_functions.h.

6.8.2.42 `typedef fmi1_status_t(* fmi1_get_boolean_status_ft)(fmi1_component_t c, const fmi1_status_kind_t s, fmi1_boolean_t *value)`

Definition at line 130 of file fmi1_functions.h.

6.8.2.43 `typedef fmi1_status_t(* fmi1_get_string_status_ft)(fmi1_component_t c, const fmi1_status_kind_t s, fmi1_string_t *value)`

Definition at line 131 of file fmi1_functions.h.

6.8.2.44 `typedef enum fmi1_value_reference_enumeration_t fmi1_value_reference_enumeration_t`

Undefined value for fmiValueReference (largest unsigned int value)

6.8.3 Enumeration Type Documentation

6.8.3.1 `enum fmi1_status_t`

FMI 1.0 status codes

Enumerator:

fmi1_status_ok
fmi1_status_warning
fmi1_status_discard
fmi1_status_error
fmi1_status_fatal
fmi1_status_pending

Definition at line 31 of file fmi1_functions.h.

6.8.3.2 `enum fmi1_status_kind_t`

FMI 1.0 asynchronous co-simulation status

Enumerator:

fmi1_do_step_status
fmi1_pending_status
fmi1_last_successful_time

Definition at line 78 of file fmi1_functions.h.

6.8.3.3 enum fmi1_boolean_enumeration

FMI boolean constants.

Enumerator:

fmi1_true
fmi1_false

Definition at line 55 of file fmi1_types.h.

6.8.3.4 enum fmi1_value_reference_enumeration

Undefined value for fmiValueReference (largest unsigned int value)

Enumerator:

fmi1_undefined_value_reference

Definition at line 64 of file fmi1_types.h.

6.8.4 Function Documentation

6.8.4.1 FMILIB_EXPORT const char* fmi1_status_to_string (fmi1_status_t status)

Convert [fmi1_status_t](#) variable to string

6.8.4.2 static const char* fmi1_get_platform (void) [static]

FMI platform name constant string.

Definition at line 48 of file fmi1_types.h.

6.9 Enum types used with FMI 1.0 libs

Typedefs

- typedef enum `fmi1_variable_naming_convension_enumeration_t` `fmi1_variable_naming_convension_enumeration_t`
Naming convention for the variables in XML file.
- typedef enum `fmi1_fmu_kind_enumeration_t` `fmi1_fmu_kind_enumeration_t`
FMU 1.0 kinds.
- typedef enum `fmi1_variability_enumeration_t` `fmi1_variability_enumeration_t`
Variability property for variables.
- typedef enum `fmi1_causality_enumeration_t` `fmi1_causality_enumeration_t`
Causality property for variables.
- typedef enum `fmi1_variable_alias_kind_enumeration_t` `fmi1_variable_alias_kind_enumeration_t`
Alias property for variables.
- typedef enum `fmi1_base_type_enumeration_t` `fmi1_base_type_enumeration_t`
Base types used in type definitions.

Enumerations

- enum `fmi1_variable_naming_convension_enumeration_t` { `fmi1_naming_enumeration_flat`, `fmi1_naming_enumeration_structured`, `fmi1_naming_enumeration_unknown` }
Naming convention for the variables in XML file.
- enum `fmi1_fmu_kind_enumeration_t` { `fmi1_fmu_kind_enumeration_me` = 0, `fmi1_fmu_kind_enumeration_cs_standalone`, `fmi1_fmu_kind_enumeration_cs_tool`, `fmi1_fmu_kind_enumeration_unknown` }
FMU 1.0 kinds.
- enum `fmi1_variability_enumeration_t` { `fmi1_variability_enumeration_constant`, `fmi1_variability_enumeration_parameter`, `fmi1_variability_enumeration_discrete`, `fmi1_variability_enumeration_continuous`, `fmi1_variability_enumeration_unknown` }
Variability property for variables.
- enum `fmi1_causality_enumeration_t` { `fmi1_causality_enumeration_input`, `fmi1_causality_enumeration_output`, `fmi1_causality_enumeration_internal`, `fmi1_causality_enumeration_none`, `fmi1_causality_enumeration_unknown` }
Causality property for variables.
- enum `fmi1_variable_alias_kind_enumeration_t` { `fmi1_variable_is_negated_alias` = -1, `fmi1_variable_is_not_alias` = 0, `fmi1_variable_is_alias` = 1 }
Alias property for variables.
- enum `fmi1_base_type_enumeration_t` { `fmi1_base_type_real`, `fmi1_base_type_int`, `fmi1_base_type_bool`, `fmi1_base_type_str`, `fmi1_base_type_enum` }
Base types used in type definitions.

Functions

- `FMILIB_EXPORT const char * fmi1_naming_convention_to_string (fmi1_variable_naming_convension_enu_t convention)`
Convert a *fmi1_variable_naming_convension_enu_t* constant into string.
- `FMILIB_EXPORT const char * fmi1_fmu_kind_to_string (fmi1_fmu_kind_enu_t kind)`
Convert a *fmi1_fmu_kind_enu_t* constant into string.
- `FMILIB_EXPORT const char * fmi1_variability_to_string (fmi1_variability_enu_t v)`
Convert a *fmi1_variability_enu_t* constant into string.
- `FMILIB_EXPORT const char * fmi1_causality_to_string (fmi1_causality_enu_t c)`
Convert a *fmi1_causality_enu_t* constant into string.
- `FMILIB_EXPORT const char * fmi1_base_type_to_string (fmi1_base_type_enu_t bt)`
Convert base type constant to string.

6.9.1 Typedef Documentation

6.9.1.1 typedef enum fmi1_variable_naming_convension_enu_t
fmi1_variable_naming_convension_enu_t

Naming convention for the variables in XML file.

6.9.1.2 typedef enum fmi1_fmu_kind_enu_t fmi1_fmu_kind_enu_t

FMU 1.0 kinds.

6.9.1.3 typedef enum fmi1_variability_enu_t fmi1_variability_enu_t

Variability property for variables.

6.9.1.4 typedef enum fmi1_causality_enu_t fmi1_causality_enu_t

Causality property for variables.

6.9.1.5 typedef enum fmi1_variable_alias_kind_enu_t
fmi1_variable_alias_kind_enu_t

Alias property for variables.

6.9.1.6 typedef enum fmi1_base_type_enu_t fmi1_base_type_enu_t

Base types used in type definitions.

6.9.2 Enumeration Type Documentation

6.9.2.1 enum fmi1_variable_naming_convension_enu_t

Naming convention for the variables in XML file.

Enumerator:

fmi1_naming_enu_flat
fmi1_naming_enu_structured
fmi1_naming_enu_unknown

Definition at line 38 of file fmi1_enums.h.

6.9.2.2 enum fmi1_fmu_kind_enu_t

FMU 1.0 kinds.

Enumerator:

fmi1_fmu_kind_enu_me
fmi1_fmu_kind_enu_cs_standalone
fmi1_fmu_kind_enu_cs_tool
fmi1_fmu_kind_enu_unknown

Definition at line 49 of file fmi1_enums.h.

6.9.2.3 enum fmi1_variability_enu_t

Variability property for variables.

Enumerator:

fmi1_variability_enu_constant
fmi1_variability_enu_parameter
fmi1_variability_enu_discrete
fmi1_variability_enu_continuous
fmi1_variability_enu_unknown

Definition at line 61 of file fmi1_enums.h.

6.9.2.4 enum fmi1_causality_enu_t

Causality property for variables.

Enumerator:

```
fmi1_causality_enu_input  
fmi1_causality_enu_output  
fmi1_causality_enu_internal  
fmi1_causality_enu_none  
fmi1_causality_enu_unknown
```

Definition at line 73 of file fmi1_enums.h.

6.9.2.5 enum fmi1_variable_alias_kind_enu_t

Alias property for variables.

Enumerator:

```
fmi1_variable_is_negated_alias  
fmi1_variable_is_not_alias  
fmi1_variable_is_alias
```

Definition at line 85 of file fmi1_enums.h.

6.9.2.6 enum fmi1_base_type_enu_t

Base types used in type definitions.

Enumerator:

```
fmi1_base_type_real  
fmi1_base_type_int  
fmi1_base_type_bool  
fmi1_base_type_str  
fmi1_base_type_enum
```

Definition at line 92 of file fmi1_enums.h.

6.9.3 Function Documentation

6.9.3.1 FMILIB_EXPORT const char* fmi1_naming_convention_to_string (fmi1_variable_naming_convension_enu_t convention)

Convert a [fmi1_variable_naming_convension_enu_t](#) constant into string.

6.9.3.2 FMILIB_EXPORT const char* fmi1_fmu_kind_to_string (
fmi1_fmu_kind_enumeration_t *kind*)

Convert a [fmi1_fmu_kind_enumeration_t](#) constant into string.

6.9.3.3 FMILIB_EXPORT const char* fmi1_variability_to_string (
fmi1_variability_enumeration_t *v*)

Convert a [fmi1_variability_enumeration_t](#) constant into string.

6.9.3.4 FMILIB_EXPORT const char* fmi1_causality_to_string (
fmi1_causality_enumeration_t *c*)

Convert a [fmi1_causality_enumeration_t](#) constant into string.

6.9.3.5 FMILIB_EXPORT const char* fmi1_base_type_to_string (
fmi1_base_type_enumeration_t *bt*)

Convert base type constant to string.

Parameters

<i>bt</i>	Base type identifier.
-----------	-----------------------

Returns

Corresponding base type name.

6.10 Functions and types supporting FMI 2.0 processing.

Data Structures

- struct [fmi2_callback_functions_t](#)
- struct [fmi2_event_info_t](#)

Modules

- Enum types used with FMI 2.0 libs
- Definition of XML callbacks struct

Typedefs

- typedef void(* [fmi2_callback_logger_ft](#))(fmi2_component_environment_t env, fmi2_string_t instanceName, [fmi2_status_t](#) status, fmi2_string_t category, fmi2_string_t message,...)
- typedef void(* [fmi2_callback_allocate_memory_ft](#))(size_t nobj, size_t size)
- typedef void(* [fmi2_callback_free_memory_ft](#))(void *obj)
- typedef void(* [fmi2_step_finished_ft](#))(fmi2_component_environment_t env, [fmi2_status_t](#) status)
- typedef const char *(* [fmi2_get_types_platform_ft](#))()
- typedef const char *(* [fmi2_get_version_ft](#))()
- typedef [fmi2_status_t](#)(* [fmi2_set_debug_logging_ft](#))(fmi2_component_t, fmi2_boolean_t, size_t nCategories, const fmi2_string_t categories[])
- typedef fmi2_component_t(* [fmi2_instantiate_ft](#))(fmi2_string_t, [fmi2_type_t](#), fmi2_string_t, fmi2_string_t, const [fmi2_callback_functions_t](#) *, fmi2_boolean_t, fmi2_boolean_t)
- typedef void(* [fmi2_free_instance_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_setup_experiment_ft](#))(fmi2_component_t, fmi2_boolean_t, fmi2_real_t, fmi2_real_t, fmi2_boolean_t, fmi2_real_t)
- typedef [fmi2_status_t](#)(* [fmi2_enter_initialization_mode_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_exit_initialization_mode_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_terminate_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_reset_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_get_real_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, fmi2_real_t[])
- typedef [fmi2_status_t](#)(* [fmi2_get_integer_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, fmi2_integer_t[])
- typedef [fmi2_status_t](#)(* [fmi2_get_boolean_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, fmi2_boolean_t[])
- typedef [fmi2_status_t](#)(* [fmi2_get_string_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, fmi2_string_t[])
- typedef [fmi2_status_t](#)(* [fmi2_set_real_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, const fmi2_real_t[])

- typedef [fmi2_status_t](#)(* [fmi2_set_integer_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], [size_t](#), const fmi2_integer_t[])
- typedef [fmi2_status_t](#)(* [fmi2_set_boolean_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], [size_t](#), const fmi2_boolean_t[])
- typedef [fmi2_status_t](#)(* [fmi2_set_string_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], [size_t](#), const fmi2_string_t[])
- typedef [fmi2_status_t](#)(* [fmi2_get_fmu_state_ft](#))(fmi2_component_t, fmi2_FMU_state_t *)
- typedef [fmi2_status_t](#)(* [fmi2_set_fmu_state_ft](#))(fmi2_component_t, fmi2_FMU_state_t)
- typedef [fmi2_status_t](#)(* [fmi2_free_fmu_state_ft](#))(fmi2_component_t, fmi2_FMU_state_t *)
- typedef [fmi2_status_t](#)(* [fmi2_serialized_fmu_state_size_ft](#))(fmi2_component_t, fmi2_FMU_state_t, [size_t](#) *)
- typedef [fmi2_status_t](#)(* [fmi2_serialize_fmu_state_ft](#))(fmi2_component_t, fmi2_FMU_state_t, fmi2_byte_t[], [size_t](#))
- typedef [fmi2_status_t](#)(* [fmi2_de_serialize_fmu_state_ft](#))(fmi2_component_t, const fmi2_byte_t[], [size_t](#), fmi2_FMU_state_t *)
- typedef [fmi2_status_t](#)(* [fmi2_get_directional_derivative_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], [size_t](#), const fmi2_value_reference_t[], [size_t](#), const fmi2_real_t[], fmi2_real_t[])
- typedef [fmi2_status_t](#)(* [fmi2_enter_event_mode_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_new_discrete_states_ft](#))(fmi2_component_t, [fmi2_event_info_t](#) *)
- typedef [fmi2_status_t](#)(* [fmi2_enter_continuous_time_mode_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_completed_integrator_step_ft](#))(fmi2_component_t, fmi2_boolean_t, fmi2_boolean_t *, fmi2_boolean_t *)
- typedef [fmi2_status_t](#)(* [fmi2_set_time_ft](#))(fmi2_component_t, fmi2_real_t)
- typedef [fmi2_status_t](#)(* [fmi2_set_continuous_states_ft](#))(fmi2_component_t, const fmi2_real_t[], [size_t](#))
- typedef [fmi2_status_t](#)(* [fmi2_get_derivatives_ft](#))(fmi2_component_t, fmi2_real_t[], [size_t](#))
- typedef [fmi2_status_t](#)(* [fmi2_get_event_indicators_ft](#))(fmi2_component_t, fmi2_real_t[], [size_t](#))
- typedef [fmi2_status_t](#)(* [fmi2_get_continuous_states_ft](#))(fmi2_component_t, fmi2_real_t[], [size_t](#))
- typedef [fmi2_status_t](#)(* [fmi2_get_nominals_of_continuous_states_ft](#))(fmi2_component_t, fmi2_real_t[], [size_t](#))
- typedef [fmi2_status_t](#)(* [fmi2_set_real_input_derivatives_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], [size_t](#), const fmi2_integer_t[], const fmi2_real_t[])
- typedef [fmi2_status_t](#)(* [fmi2_get_real_output_derivatives_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], [size_t](#), const fmi2_integer_t[], fmi2_real_t[])
- typedef [fmi2_status_t](#)(* [fmi2_do_step_ft](#))(fmi2_component_t, fmi2_real_t, fmi2_real_t, fmi2_boolean_t)
- typedef [fmi2_status_t](#)(* [fmi2_cancel_step_ft](#))(fmi2_component_t)

- `typedef fmi2_status_t(* fmi2_get_status_ft)(fmi2_component_t, const fmi2_status_kind_t, fmi2_status_t *)`
- `typedef fmi2_status_t(* fmi2_get_real_status_ft)(fmi2_component_t, const fmi2_status_kind_t, fmi2_real_t *)`
- `typedef fmi2_status_t(* fmi2_get_integer_status_ft)(fmi2_component_t, const fmi2_status_kind_t, fmi2_integer_t *)`
- `typedef fmi2_status_t(* fmi2_get_boolean_status_ft)(fmi2_component_t, const fmi2_status_kind_t, fmi2_boolean_t *)`
- `typedef fmi2_status_t(* fmi2_get_string_status_ft)(fmi2_component_t, const fmi2_status_kind_t, fmi2_string_t *)`

Enumerations

- `enum fmi2_status_t { fmi2_status_ok, fmi2_status_warning, fmi2_status_discard, fmi2_status_error, fmi2_status_fatal, fmi2_status_pending }`
- `enum fmi2_type_t { fmi2_model_exchange, fmi2_cosimulation }`
- `enum fmi2_status_kind_t { fmi2_do_step_status, fmi2_pending_status, fmi2_last_successful_time, fmi2_terminated }`
- `enum fmi2_boolean_enumeration_t { fmi2_true = fmi2True, fmi2_false = fmi2False }`

Functions

- `FMI2_EXPORT const char * fmi2_status_to_string(fmi2_status_t status)`
- `static const char * fmi2_get_types_platform(void)`

Renaming of typedefs

- `#define fmi2Component fmi2_component_t`
- `#define fmi2ComponentEnvironment fmi2_component_environment_t`
- `#define fmi2FMUState fmi2_FMU_state_t`
- `#define fmi2ValueReference fmi2_value_reference_t`
- `#define fmi2Real fmi2_real_t`
- `#define fmi2Integer fmi2_integer_t`
- `#define fmi2Boolean fmi2_boolean_t`
- `#define fmi2Char fmi2_char_t`
- `#define fmi2String fmi2_string_t`
- `#define fmi2Byte fmi2_byte_t`

6.10.1 Define Documentation

6.10.1.1 `#define fmi2Component fmi2_component_t`

Definition at line 33 of file `fmi2_types.h`.

6.10.1.2 `#define fmi2ComponentEnvironment fmi2_component_environment_t`

Definition at line 34 of file `fmi2_types.h`.

6.10.1.3 `#define fmi2FMUstate fmi2_FMU_state_t`

Definition at line 35 of file `fmi2_types.h`.

6.10.1.4 `#define fmi2ValueReference fmi2_value_reference_t`

Definition at line 36 of file `fmi2_types.h`.

6.10.1.5 `#define fmi2Real fmi2_real_t`

Definition at line 37 of file `fmi2_types.h`.

6.10.1.6 `#define fmi2Integer fmi2_integer_t`

Definition at line 38 of file `fmi2_types.h`.

6.10.1.7 `#define fmi2Boolean fmi2_boolean_t`

Definition at line 39 of file `fmi2_types.h`.

6.10.1.8 `#define fmi2Char fmi2_char_t`

Definition at line 40 of file `fmi2_types.h`.

6.10.1.9 `#define fmi2String fmi2_string_t`

Definition at line 41 of file `fmi2_types.h`.

6.10.1.10 `#define fmi2Byte fmi2_byte_t`

Definition at line 42 of file `fmi2_types.h`.

6.10.2 Typedef Documentation

6.10.2.1 `typedef void(* fmi2_callback_logger_ft)(fmi2_component_environment_t env, fmi2_string_t instanceName, fmi2_status_t status, fmi2_string_t category, fmi2_string_t message,...)`

FMI 2.0 logger function type

Definition at line 61 of file fmi2_functions.h.

6.10.2.2 `typedef void>(* fmi2_callback_allocate_memory_ft)(size_t nobj, size_t size)`

FMI 2.0 allocate memory function type

Definition at line 63 of file fmi2_functions.h.

6.10.2.3 `typedef void(* fmi2_callback_free_memory_ft)(void *obj)`

FMI 2.0 free memory function type

Definition at line 65 of file fmi2_functions.h.

6.10.2.4 `typedef void(* fmi2_step_finished_ft)(fmi2_component_environment_t env,
fmi2_status_t status)`

FMI 2.0 step finished callback function type

Definition at line 67 of file fmi2_functions.h.

6.10.2.5 `typedef const char>(* fmi2_get_types_platform_ft)()`

Definition at line 112 of file fmi2_functions.h.

6.10.2.6 `typedef const char(* fmi2_get_version_ft)()`

Definition at line 113 of file fmi2_functions.h.

6.10.2.7 `typedef fmi2_status_t(* fmi2_set_debug_logging_ft)(fmi2_component_t,
fmi2_boolean_t, size_t nCategories, const fmi2_string_t categories[])`

Definition at line 114 of file fmi2_functions.h.

6.10.2.8 `typedef fmi2_component_t(* fmi2_instantiate_ft)(fmi2_string_t, fmi2_type_t,
fmi2_string_t, fmi2_string_t, const fmi2_callback_functions_t *, fmi2_boolean_t,
fmi2_boolean_t)`

Definition at line 117 of file fmi2_functions.h.

6.10.2.9 `typedef void(* fmi2_free_instance_ft)(fmi2_component_t)`

Definition at line 118 of file fmi2_functions.h.

6.10.2.10 `typedef fmi2_status_t(* fmi2_setup_experiment_ft)(fmi2_component_t,
fmi2_boolean_t, fmi2_real_t, fmi2_real_t, fmi2_boolean_t, fmi2_real_t)`

Definition at line 121 of file fmi2_functions.h.

6.10.2.11 `typedef fmi2_status_t(* fmi2_enter_initialization_mode_ft)(fmi2_component-
_t)`

Definition at line 122 of file fmi2_functions.h.

6.10.2.12 `typedef fmi2_status_t(* fmi2_exit_initialization_mode_ft)(fmi2_component_t)`

Definition at line 123 of file fmi2_functions.h.

6.10.2.13 `typedef fmi2_status_t(* fmi2_terminate_ft)(fmi2_component_t)`

Definition at line 124 of file fmi2_functions.h.

6.10.2.14 `typedef fmi2_status_t(* fmi2_reset_ft)(fmi2_component_t)`

Definition at line 125 of file fmi2_functions.h.

6.10.2.15 `typedef fmi2_status_t(* fmi2_get_real_ft)(fmi2_component_t, const
fmi2_value_reference_t[], size_t, fmi2_real_t[])`

Definition at line 128 of file fmi2_functions.h.

6.10.2.16 `typedef fmi2_status_t(* fmi2_get_integer_ft)(fmi2_component_t, const
fmi2_value_reference_t[], size_t, fmi2_integer_t[])`

Definition at line 129 of file fmi2_functions.h.

6.10.2.17 `typedef fmi2_status_t(* fmi2_get_boolean_ft)(fmi2_component_t, const
fmi2_value_reference_t[], size_t, fmi2_boolean_t[])`

Definition at line 130 of file fmi2_functions.h.

6.10.2.18 `typedef fmi2_status_t(* fmi2_get_string_ft)(fmi2_component_t, const
fmi2_value_reference_t[], size_t, fmi2_string_t[])`

Definition at line 131 of file fmi2_functions.h.

6.10.2.19 `typedef fmi2_status_t(* fmi2_set_real_ft)(fmi2_component_t, const
fmi2_value_reference_t[], size_t, const fmi2_real_t[])`

Definition at line 133 of file fmi2_functions.h.

6.10.2.20 `typedef fmi2_status_t(* fmi2_set_integer_ft)(fmi2_component_t, const
fmi2_value_reference_t[], size_t, const fmi2_integer_t[])`

Definition at line 134 of file fmi2_functions.h.

6.10.2.21 `typedef fmi2_status_t(* fmi2_set_boolean_ft)(fmi2_component_t, const
fmi2_value_reference_t[], size_t, const fmi2_boolean_t[])`

Definition at line 135 of file fmi2_functions.h.

6.10.2.22 `typedef fmi2_status_t(* fmi2_set_string_ft)(fmi2_component_t, const
fmi2_value_reference_t[], size_t, const fmi2_string_t[])`

Definition at line 136 of file fmi2_functions.h.

6.10.2.23 `typedef fmi2_status_t(* fmi2_get_fmu_state_ft)(fmi2_component_t,
fmi2_FMU_state_t *)`

Definition at line 139 of file fmi2_functions.h.

6.10.2.24 `typedef fmi2_status_t(* fmi2_set_fmu_state_ft)(fmi2_component_t,
fmi2_FMU_state_t)`

Definition at line 140 of file fmi2_functions.h.

6.10.2.25 `typedef fmi2_status_t(* fmi2_free_fmu_state_ft)(fmi2_component_t,
fmi2_FMU_state_t *)`

Definition at line 141 of file fmi2_functions.h.

6.10.2.26 `typedef fmi2_status_t(* fmi2_serialized_fmu_state-
_size_ft)(fmi2_component_t, fmi2_FMU_state_t, size_t
*)`

Definition at line 142 of file fmi2_functions.h.

6.10.2.27 `typedef fmi2_status_t(* fmi2_serialize_fmu_state_ft)(fmi2_component_t, fmi2_FMU_state_t, fmi2_byte_t[], size_t)`

Definition at line 143 of file `fmi2_functions.h`.

6.10.2.28 `typedef fmi2_status_t(* fmi2_de_serialize_fmu_state_ft)(fmi2_component_t, const fmi2_byte_t[], size_t, fmi2_FMU_state_t *)`

Definition at line 144 of file `fmi2_functions.h`.

6.10.2.29 `typedef fmi2_status_t(* fmi2_get_directional_derivative_ft)(fmi2_component_t, const fmi2_value_reference_t[], size_t, const fmi2_value_reference_t[], size_t, const fmi2_real_t[], fmi2_real_t[])`

Definition at line 147 of file `fmi2_functions.h`.

6.10.2.30 `typedef fmi2_status_t(* fmi2_enter_event_mode_ft)(fmi2_component_t)`

Definition at line 156 of file `fmi2_functions.h`.

6.10.2.31 `typedef fmi2_status_t(* fmi2_new_discrete_states_ft)(fmi2_component_t, fmi2_event_info_t *)`

Definition at line 157 of file `fmi2_functions.h`.

6.10.2.32 `typedef fmi2_status_t(* fmi2_enter_continuous_time_mode_ft)(fmi2_component_t)`

Definition at line 158 of file `fmi2_functions.h`.

6.10.2.33 `typedef fmi2_status_t(* fmi2_completed_integrator_step_ft)(fmi2_component_t, fmi2_boolean_t, fmi2_boolean_t *, fmi2_boolean_t *)`

Definition at line 159 of file `fmi2_functions.h`.

6.10.2.34 `typedef fmi2_status_t(* fmi2_set_time_ft)(fmi2_component_t, fmi2_real_t)`

Definition at line 162 of file `fmi2_functions.h`.

6.10.2.35 `typedef fmi2_status_t(* fmi2_set_continuous_states_ft)(fmi2_component_t, const fmi2_real_t[], size_t)`

Definition at line 163 of file `fmi2_functions.h`.

6.10.2.36 `typedef fmi2_status_t(* fmi2_get_derivatives_ft)(fmi2_component_t,
fmi2_real_t[], size_t)`

Definition at line 166 of file fmi2_functions.h.

6.10.2.37 `typedef fmi2_status_t(* fmi2_get_event_indicators_ft)(fmi2_component_t,
fmi2_real_t[], size_t)`

Definition at line 167 of file fmi2_functions.h.

6.10.2.38 `typedef fmi2_status_t(* fmi2_get_continuous_states_ft)(fmi2_component_t,
fmi2_real_t[], size_t)`

Definition at line 168 of file fmi2_functions.h.

6.10.2.39 `typedef fmi2_status_t(* fmi2_get_nominals_of_-
continuous_states_ft)(fmi2_component_t, fmi2_real_t[],
size_t)`

Definition at line 169 of file fmi2_functions.h.

6.10.2.40 `typedef fmi2_status_t(* fmi2_set_real_input_derivatives_ft)(fmi2_-
component_t, const fmi2_value_reference_t[], size_t, const fmi2_integer_t[], const
fmi2_real_t[])`

Definition at line 177 of file fmi2_functions.h.

6.10.2.41 `typedef fmi2_status_t(* fmi2_get_real_output_derivatives_ft)(fmi2_-
component_t, const fmi2_value_reference_t[], size_t, const fmi2_integer_t[],
fmi2_real_t[])`

Definition at line 178 of file fmi2_functions.h.

6.10.2.42 `typedef fmi2_status_t(* fmi2_do_step_ft)(fmi2_component_t, fmi2_real_t,
fmi2_real_t, fmi2_boolean_t)`

Definition at line 180 of file fmi2_functions.h.

6.10.2.43 `typedef fmi2_status_t(* fmi2_cancel_step_ft)(fmi2_component_t)`

Definition at line 181 of file fmi2_functions.h.

6.10.2.44 `typedef fmi2_status_t(* fmi2_get_status_ft)(fmi2_component_t, const
fmi2_status_kind_t, fmi2_status_t *)`

Definition at line 184 of file `fmi2_functions.h`.

6.10.2.45 `typedef fmi2_status_t(* fmi2_get_real_status_ft)(fmi2_component_t, const
fmi2_status_kind_t, fmi2_real_t *)`

Definition at line 185 of file `fmi2_functions.h`.

6.10.2.46 `typedef fmi2_status_t(* fmi2_get_integer_status_ft)(fmi2_component_t, const
fmi2_status_kind_t, fmi2_integer_t *)`

Definition at line 186 of file `fmi2_functions.h`.

6.10.2.47 `typedef fmi2_status_t(* fmi2_get_boolean_status_ft)(fmi2_component_t,
const fmi2_status_kind_t, fmi2_boolean_t *)`

Definition at line 187 of file `fmi2_functions.h`.

6.10.2.48 `typedef fmi2_status_t(* fmi2_get_string_status_ft)(fmi2_component_t, const
fmi2_status_kind_t, fmi2_string_t *)`

Definition at line 188 of file `fmi2_functions.h`.

6.10.3 Enumeration Type Documentation

6.10.3.1 `enum fmi2_status_t`

FMI 2.0 status codes

Enumerator:

fmi2_status_ok
fmi2_status_warning
fmi2_status_discard
fmi2_status_error
fmi2_status_fatal
fmi2_status_pending

Definition at line 40 of file `fmi2_functions.h`.

6.10.3.2 enum `fmi2_type_t`

Enumerator:

fmi2_model_exchange

fmi2_cosimulation

Definition at line 49 of file `fmi2_functions.h`.

6.10.3.3 enum `fmi2_status_kind_t`

Co-simulation status for FMI 2.0 CS

Enumerator:

fmi2_do_step_status

fmi2_pending_status

fmi2_last_successful_time

fmi2_terminated

Definition at line 91 of file `fmi2_functions.h`.

6.10.3.4 enum `fmi2_boolean_enumeration_t`

FMI boolean constants.

Enumerator:

fmi2_true

fmi2_false

Definition at line 60 of file `fmi2_types.h`.

6.10.4 Function Documentation

6.10.4.1 `FMILIB_EXPORT const char* fmi2_status_to_string ( fmi2_status_t status )`

Convert [fmi2_status_t](#) variable to string

6.10.4.2 `static const char* fmi2_get_types_platform ( void ) [static]`

FMI platform name constant string.

Definition at line 53 of file `fmi2_types.h`.

6.11 Enum types used with FMI 2.0 libs

Defines

- `#define FMI2_ME_CAPABILITIES(H)`
List of capability flags for ModelExchange.
- `#define FMI2_CS_CAPABILITIES(H)`
List of capability flags for CoSimulation.
- `#define FMI2_SI_BASE_UNITS(H) H(kg) H(m) H(s) H(A) H(K) H(mol) H(cd) - H(rad)`
List of SI base units used in Unit definitions.

Typedefs

- `typedef enum fmi2_variable_naming_convension_enu_t fmi2_variable_naming_convension_enu_t`
Naming convention for the variables in XML file.
- `typedef enum fmi2_fmu_kind_enu_t fmi2_fmu_kind_enu_t`
FMU 2.0 kinds.
- `typedef enum fmi2_variability_enu_t fmi2_variability_enu_t`
Variability property for variables.
- `typedef enum fmi2_causality_enu_t fmi2_causality_enu_t`
Causality property for variables.
- `typedef enum fmi2_initial_enu_t fmi2_initial_enu_t`
Initial property for variables.
- `typedef enum fmi2_variable_alias_kind_enu_t fmi2_variable_alias_kind_enu_t`
Alias property for variables.
- `typedef enum fmi2_base_type_enu_t fmi2_base_type_enu_t`
Base types used in type definitions.
- `typedef enum fmi2_capabilities_enu_t fmi2_capabilities_enu_t`
Capability flags for ModelExchange and CoSimulation.
- `typedef enum fmi2_SI_base_units_enu_t fmi2_SI_base_units_enu_t`
SI base units used in Unit definitions.
- `typedef enum fmi2_dependency_factor_kind_enu_t fmi2_dependency_factor_kind_enu_t`
Dependency factor kinds are used as part of ModelStructure definition.

Enumerations

- `enum fmi2_variable_naming_convension_enu_t { fmi2_naming_enu_flat, fmi2_naming_enu_structured, fmi2_naming_enu_unknown }`
Naming convention for the variables in XML file.
- `enum fmi2_fmu_kind_enu_t { fmi2_fmu_kind_unknown = 0, fmi2_fmu_kind_me = 1, fmi2_fmu_kind_cs = 2, fmi2_fmu_kind_me_and_cs = 3 }`

FMU 2.0 kinds.

- enum `fmi2_variability_enu_t` { `fmi2_variability_enu_constant` = 0, `fmi2_variability_enu_fixed` = 1, `fmi2_variability_enu_tunable` = 2, `fmi2_variability_enu_discrete` = 3, `fmi2_variability_enu_continuous` = 4, `fmi2_variability_enu_unknown` = 5 }

Variability property for variables.

- enum `fmi2_causality_enu_t` { `fmi2_causality_enu_parameter` = 0, `fmi2_causality_enu_calculated_parameter` = 1, `fmi2_causality_enu_input` = 2, `fmi2_causality_enu_output` = 3, `fmi2_causality_enu_local` = 4, `fmi2_causality_enu_independent` = 5, `fmi2_causality_enu_unknown` = 6 }

Causality property for variables.

- enum `fmi2_initial_enu_t` { `fmi2_initial_enu_exact`, `fmi2_initial_enu_approx`, `fmi2_initial_enu_calculated`, `fmi2_initial_enu_unknown` }

Initial property for variables.

- enum `fmi2_variable_alias_kind_enu_t` { `fmi2_variable_is_not_alias` = 0, `fmi2_variable_is_alias` = 1 }

Alias property for variables.

- enum `fmi2_base_type_enu_t` { `fmi2_base_type_real`, `fmi2_base_type_int`, `fmi2_base_type_bool`, `fmi2_base_type_str`, `fmi2_base_type_enum` }

Base types used in type definitions.

- enum `fmi2_capabilities_enu_t`

Capability flags for ModelExchange and CoSimulation.

- enum `fmi2_SI_base_units_enu_t`

SI base units used in Unit definitions.

- enum `fmi2_dependency_factor_kind_enu_t` { `fmi2_dependency_factor_kind_dependent` = 0, `fmi2_dependency_factor_kind_constant`, `fmi2_dependency_factor_kind_fixed`, `fmi2_dependency_factor_kind_tunable`, `fmi2_dependency_factor_kind_discrete`, `fmi2_dependency_factor_kind_num` }

Dependency factor kinds are used as part of ModelStructure definition.

Functions

- `FMILIB_EXPORT` const char * `fmi2_naming_convention_to_string` (`fmi2_variable_naming_convention_enu_t` convention)

Convert a `fmi2_variable_naming_convention_enu_t` constant into string.

- `FMILIB_EXPORT` const char * `fmi2_fmu_kind_to_string` (`fmi2_fmu_kind_enu_t` kind)

Convert a `fmi2_fmu_kind_enu_t` constant into string.

- `FMILIB_EXPORT` const char * `fmi2_variability_to_string` (`fmi2_variability_enu_t` v)

Convert a `fmi2_variability_enu_t` constant into string.

- `FMILIB_EXPORT` const char * `fmi2_causality_to_string` (`fmi2_causality_enu_t` c)

Convert a `fmi2_causality_enu_t` constant into string.

- `FMILIB_EXPORT` const char * `fmi2_initial_to_string` (`fmi2_initial_enu_t` c)

Convert a `fmi2_initial_enu_t` constant into string.

- `FMLIB_EXPORT fmi2_initial_enu_t fmi2_get_default_initial (fmi2_variability_enu_t v, fmi2_causality_enu_t c)`
Get default initial attribute value for the given variability and causality combination.
- `FMLIB_EXPORT fmi2_initial_enu_t fmi2_get_valid_initial (fmi2_variability_enu_t v, fmi2_causality_enu_t c, fmi2_initial_enu_t i)`
Check if the combination of variability, causality and initial is valid.
- `FMLIB_EXPORT const char * fmi2_base_type_to_string (fmi2_base_type_enu_t bt)`
Convert base type constant to string.
- `FMLIB_EXPORT const char * fmi2_capability_to_string (fmi2_capabilities_enu_t id)`
Convert capability flag to a string.
- `FMLIB_EXPORT const char * fmi2_SI_base_unit_to_string (fmi2_SI_base_units_enu_t id)`
Convert SI base unit ID a string.
- `FMLIB_EXPORT size_t fmi2_SI_base_unit_exp_to_string (const int exp[fmi2_SI_base_units_Num], size_t bufSize, char buf[])`
Convert a list of SI base unit exponents (corresponding to the IDs from `fmi2_SI_base_units_enu_t`) to a string of the form $kg \cdot m^2 / s^2$. Prints '-' if all the exponents are zero.
- `FMLIB_EXPORT const char * fmi2_dependency_factor_kind_to_string (fmi2_dependency_factor_kind_enu_t fc)`
Convert dependency factor kind constant to string.

6.11.1 Define Documentation

6.11.1.1 #define FMI2_ME_CAPABILITIES(H)

Value:

```
H(needsExecutionTool) \
  H(completedIntegratorStepNotNeeded) \
  H(canBeInstantiatedOnlyOncePerProcess) \
  H(canNotUseMemoryManagementFunctions) \
  H(canGetAndSetFMUstate) \
  H(canSerializeFMUstate) \
  H(providesDirectionalDerivatives) \
  H(completedEventIterationIsProvided)
```

List of capability flags for ModelExchange.

Definition at line 134 of file `fmi2_enums.h`.

6.11.1.2 #define FMI2_CS_CAPABILITIES(H)

Value:

```

H(needsExecutionTool) \
  H(canHandleVariableCommunicationStepSize) \
  H(canInterpolateInputs) \
  H(maxOutputDerivativeOrder) \
  H(canRunAsynchronously) \
  H(canBeInstantiatedOnlyOncePerProcess) \
  H(canNotUseMemoryManagementFunctions) \
  H(canGetAndSetFMUstate) \
  H(canSerializeFMUstate) \
  H(providesDirectionalDerivatives)

```

List of capability flags for CoSimulation.

Definition at line 145 of file fmi2_enums.h.

6.11.1.3 `#define FMI2_SI_BASE_UNITS( H ) H(kg) H(m) H(s) H(A) H(K) H(mol) H(cd) H(rad)`

List of SI base units used in Unit definitions.

Definition at line 173 of file fmi2_enums.h.

6.11.2 Typedef Documentation

6.11.2.1 `typedef enum fmi2_variable_naming_convension_enu_t
fmi2_variable_naming_convension_enu_t`

Naming convention for the variables in XML file.

6.11.2.2 `typedef enum fmi2_fmu_kind_enu_t fmi2_fmu_kind_enu_t`

FMU 2.0 kinds.

6.11.2.3 `typedef enum fmi2_variability_enu_t fmi2_variability_enu_t`

Variability property for variables.

6.11.2.4 `typedef enum fmi2_causality_enu_t fmi2_causality_enu_t`

Causality property for variables.

6.11.2.5 `typedef enum fmi2_initial_enu_t fmi2_initial_enu_t`

Initial property for variables.

6.11.2.6 `typedef enum fmi2_variable_alias_kind_enu_t`
`fmi2_variable_alias_kind_enu_t`

Alias property for variables.

6.11.2.7 `typedef enum fmi2_base_type_enu_t fmi2_base_type_enu_t`

Base types used in type definitions.

6.11.2.8 `typedef enum fmi2_capabilities_enu_t fmi2_capabilities_enu_t`

Capability flags for ModelExchange and CoSimulation.

6.11.2.9 `typedef enum fmi2_SI_base_units_enu_t fmi2_SI_base_units_enu_t`

SI base units used in Unit definitions.

6.11.2.10 `typedef enum fmi2_dependency_factor_kind_enu_t`
`fmi2_dependency_factor_kind_enu_t`

Dependency factor kinds are used as part of ModelStructure definition.

6.11.3 Enumeration Type Documentation

6.11.3.1 `enum fmi2_variable_naming_convension_enu_t`

Naming convention for the variables in XML file.

Enumerator:

fmi2_naming_enu_flat
fmi2_naming_enu_structured
fmi2_naming_enu_unknown

Definition at line 38 of file fmi2_enums.h.

6.11.3.2 `enum fmi2_fmu_kind_enu_t`

FMU 2.0 kinds.

Enumerator:

fmi2_fmu_kind_unknown
fmi2_fmu_kind_me

fmi2_fmu_kind_cs
fmi2_fmu_kind_me_and_cs

Definition at line 49 of file fmi2_enums.h.

6.11.3.3 enum fmi2_variability_enu_t

Variability property for variables.

Enumerator:

fmi2_variability_enu_constant
fmi2_variability_enu_fixed
fmi2_variability_enu_tunable
fmi2_variability_enu_discrete
fmi2_variability_enu_continuous
fmi2_variability_enu_unknown

Definition at line 61 of file fmi2_enums.h.

6.11.3.4 enum fmi2_causality_enu_t

Causality property for variables.

Enumerator:

fmi2_causality_enu_parameter
fmi2_causality_enu_calculated_parameter
fmi2_causality_enu_input
fmi2_causality_enu_output
fmi2_causality_enu_local
fmi2_causality_enu_independent
fmi2_causality_enu_unknown

Definition at line 74 of file fmi2_enums.h.

6.11.3.5 enum fmi2_initial_enu_t

Initial property for variables.

Enumerator:

fmi2_initial_enu_exact
fmi2_initial_enu_approx
fmi2_initial_enu_calculated
fmi2_initial_enu_unknown

Definition at line 88 of file fmi2_enums.h.

6.11.3.6 enum `fmi2_variable_alias_kind_enu_t`

Alias property for variables.

Enumerator:

fmi2_variable_is_not_alias
fmi2_variable_is_alias

Definition at line 112 of file `fmi2_enums.h`.

6.11.3.7 enum `fmi2_base_type_enu_t`

Base types used in type definitions.

Enumerator:

fmi2_base_type_real
fmi2_base_type_int
fmi2_base_type_bool
fmi2_base_type_str
fmi2_base_type_enum

Definition at line 118 of file `fmi2_enums.h`.

6.11.3.8 enum `fmi2_capabilities_enu_t`

Capability flags for ModelExchange and CoSimulation.

Definition at line 158 of file `fmi2_enums.h`.

6.11.3.9 enum `fmi2_SI_base_units_enu_t`

SI base units used in Unit definitions.

Definition at line 177 of file `fmi2_enums.h`.

6.11.3.10 enum `fmi2_dependency_factor_kind_enu_t`

Dependency factor kinds are used as part of ModelStructure definition.

Enumerator:

fmi2_dependency_factor_kind_dependent
fmi2_dependency_factor_kind_constant
fmi2_dependency_factor_kind_fixed

fmi2_dependency_factor_kind_tunable
fmi2_dependency_factor_kind_discrete
fmi2_dependency_factor_kind_num

Definition at line 201 of file fmi2_enums.h.

6.11.4 Function Documentation

6.11.4.1 **FMILIB_EXPORT** const char* fmi2_naming_convention_to_string (fmi2_variable_naming_convension_enu_t convention)

Convert a [fmi2_variable_naming_convension_enu_t](#) constant into string.

6.11.4.2 **FMILIB_EXPORT** const char* fmi2_fmu_kind_to_string (fmi2_fmu_kind_enu_t kind)

Convert a [fmi2_fmu_kind_enu_t](#) constant into string.

6.11.4.3 **FMILIB_EXPORT** const char* fmi2_variability_to_string (fmi2_variability_enu_t v)

Convert a [fmi2_variability_enu_t](#) constant into string.

6.11.4.4 **FMILIB_EXPORT** const char* fmi2_causality_to_string (fmi2_causality_enu_t c)

Convert a [fmi2_causality_enu_t](#) constant into string.

6.11.4.5 **FMILIB_EXPORT** const char* fmi2_initial_to_string (fmi2_initial_enu_t c)

Convert a [fmi2_initial_enu_t](#) constant into string.

6.11.4.6 **FMILIB_EXPORT** fmi2_initial_enu_t fmi2_get_default_initial (fmi2_variability_enu_t v, fmi2_causality_enu_t c)

Get default initial attribute value for the given variability and causality combination.

Returns

The default initial attribute or fmi2_initial_enu_unknown if combination of causality and variability is not valid.

6.11.4.7 **FMILIB_EXPORT** fmi2_initial_enu_t fmi2_get_valid_initial (fmi2_variability_enu_t *v*, fmi2_causality_enu_t *c*, fmi2_initial_enu_t *i*)

Check if the combination of variability, causality and initial is valid.

Returns

Same initial as submitted if the combination is valid. Otherwise, same as fmi2_get_default_initial.

6.11.4.8 **FMILIB_EXPORT** const char* fmi2_base_type_to_string (fmi2_base_type_enu_t *bt*)

Convert base type constant to string.

Parameters

<i>bt</i>	Base type identifier.
-----------	-----------------------

Returns

Corresponding base type name.

6.11.4.9 **FMILIB_EXPORT** const char* fmi2_capability_to_string (fmi2_capabilities_enu_t *id*)

Convert capability flag to a string.

Parameters

<i>id</i>	Capability flag ID.
-----------	---------------------

Returns

Name of the flag or Unknown if the id is out of range.

6.11.4.10 **FMILIB_EXPORT** const char* fmi2_SI_base_unit_to_string (fmi2_SI_base_units_enu_t *id*)

Convert SI base unit ID a string.

Parameters

<i>id</i>	SI base unit ID.
-----------	------------------

Returns

Name of the base unit or "unknown" if the id is out of range.

6.11.4.11 FMILIB_EXPORT `size_t fmi2_SI_base_unit_exp_to_string ( const int exp[fmi2_SI_base_units_Num], size_t bufSize, char buf[] )`

Convert a list of SI base unit exponents (corresponding to the IDs from `fmi2_SI_base_units_enu_t`) to a string of the form `kg*m^2/s^2`. Prints '-' if all the exponents are zero.

Parameters

<i>exp</i>	An array of SI base units exponents.
<i>bufSize</i>	Size of the buffer to store the string.
<i>buf</i>	Buffer to store the string

Returns

Required size of the buffer to store the string. This most likely be under `[8*fmi2_SI_base_units_Num]`. If the return value is larger or equal than `bufSize` than the string could not be fitted in the buffer.

6.11.4.12 FMILIB_EXPORT `const char* fmi2_dependency_factor_kind_to_string ( fmi2_dependency_factor_kind_enu_t fc )`

Convert dependency factor kind constant to string.

Parameters

<i>fc</i>	Dependency factor kind identifier.
-----------	------------------------------------

Returns

Corresponding factor kind as string.

6.12 Definition of XML callbacks struct

Data Structures

- struct [fmi2_xml_callbacks_t](#)

XML callbacks are used to process parts of XML that are not handled by the library.

Typedefs

- typedef struct [fmi2_xml_callbacks_t](#) [fmi2_xml_callbacks_t](#)

Variables

- [fmi2_xml_element_start_handle_ft](#) [fmi2_xml_callbacks_t::startHandle](#)
- [fmi2_xml_element_data_handle_ft](#) [fmi2_xml_callbacks_t::dataHandle](#)
Handle start of an XML element within tool annotation in a SAX parser.
- [fmi2_xml_element_end_handle_ft](#) [fmi2_xml_callbacks_t::endHandle](#)
Handle data of an XML element within tool annotation in a SAX parser.
- void * [fmi2_xml_callbacks_t::context](#)
Handle end of an XML element within tool annotation in a SAX parser.

XML handling callbacks

- typedef int(* [fmi2_xml_element_start_handle_ft](#))(void *context, const char *tool-Name, void *parent, const char *elm, const char **attr)
Handle start of an XML element within tool annotation in a SAX parser.
- typedef int(* [fmi2_xml_element_data_handle_ft](#))(void *context, const char *s, int len)
Handle data of an XML element within tool annotation in a SAX parser.
- typedef int(* [fmi2_xml_element_end_handle_ft](#))(void *context, const char *elm)
Handle end of an XML element within tool annotation in a SAX parser.

6.12.1 Typedef Documentation

6.12.1.1 typedef struct [fmi2_xml_callbacks_t](#) [fmi2_xml_callbacks_t](#)

Definition at line 33 of file [fmi2_xml_callbacks.h](#).

6.12.1.2 `typedef int(* fmi2_xml_element_start_handle_ft)(void *context, const char *toolName, void *parent, const char *elm, const char **attr)`

Handle start of an XML element within tool annotation in a SAX parser.

Parameters

<i>context</i>	as specified when setting up the callbacks,
<i>parentName</i>	- tool name as given by name attribute to the Tool element,
<i>parent</i>	- NULL for model level annotations; <code>fmi2_import_variable_t * variable</code> pointer for variable annotations.
<i>elm</i>	- name of the element,
<i>attr</i>	- attributes (names and values). The function should return 0 on success or error code on exit (in which case parsing will be aborted).

Definition at line 47 of file `fmi2_xml_callbacks.h`.

6.12.1.3 `typedef int(* fmi2_xml_element_data_handle_ft)(void *context, const char *s, int len)`

Handle data of an XML element within tool annotation in a SAX parser.

Parameters

<i>context</i>	as specified when setting up the callbacks
<i>s</i>	- data string
<i>len</i>	- length of the data. The function should return 0 on success or error code on exit (in which case parsing will be aborted).

Definition at line 56 of file `fmi2_xml_callbacks.h`.

6.12.1.4 `typedef int(* fmi2_xml_element_end_handle_ft)(void *context, const char *elm)`

Handle end of an XML element within tool annotation in a SAX parser.

Parameters

<i>context</i>	as specified when setting up the callbacks
<i>elm</i>	- name of the element. The function should return 0 on success or error code on exit (in which case parsing will be aborted).

Definition at line 64 of file `fmi2_xml_callbacks.h`.

6.12.2 Variable Documentation

6.12.2.1 fmi2_xml_element_start_handle_ft fmi2_xml_callbacks_t::startHandle

Definition at line 68 of file fmi2_xml_callbacks.h.

6.12.2.2 fmi2_xml_element_data_handle_ft fmi2_xml_callbacks_t::dataHandle

Handle start of an XML element within tool annotation in a SAX parser.

Definition at line 69 of file fmi2_xml_callbacks.h.

6.12.2.3 fmi2_xml_element_end_handle_ft fmi2_xml_callbacks_t::endHandle

Handle data of an XML element within tool annotation in a SAX parser.

Definition at line 70 of file fmi2_xml_callbacks.h.

6.12.2.4 void* fmi2_xml_callbacks_t::context

Handle end of an XML element within tool annotation in a SAX parser.

Definition at line 71 of file fmi2_xml_callbacks.h.

6.13 Definition of callbacks struct and supporting functions

Data Structures

- struct [jm_callbacks](#)

The callbacks struct is sent to all the modules in the library.

Defines

- #define [JM_MAX_ERROR_MESSAGE_SIZE](#) 2000

Maximum message size that can be stored in the [jm_callbacks](#) struct.

Typedefs

- typedef struct [jm_callbacks](#) [jm_callbacks](#)
- typedef void(* [jm_logger_f](#))([jm_callbacks](#) *c, [jm_string](#) module, [jm_log_level_enu_t](#) log_level, [jm_string](#) message)

Logger callback type.

Functions

- static [jm_string](#) [jm_get_last_error](#) ([jm_callbacks](#) *cb)
Get the last log message produced by the library.
- static void [jm_clear_last_error](#) ([jm_callbacks](#) *cb)
Clear the last generated log message.
- [FMILIB_EXPORT](#) void [jm_set_default_callbacks](#) ([jm_callbacks](#) *c)
Set the structure to be returned by [jm_get_default_callbacks\(\)](#).
- [FMILIB_EXPORT](#) [jm_callbacks](#) * [jm_get_default_callbacks](#) (void)
Get default callbacks. The function never returns NULL.
- [FMILIB_EXPORT](#) void [jm_default_logger](#) ([jm_callbacks](#) *c, [jm_string](#) module, [jm_log_level_enu_t](#) log_level, [jm_string](#) message)
The default logger implementation prints messages to stderr.
- [FMILIB_EXPORT](#) void [jm_log](#) ([jm_callbacks](#) *cb, const [char](#) *module, [jm_log_level_enu_t](#) log_level, const [char](#) *fmt,...)
Send a message to the logger function.
- [FMILIB_EXPORT](#) void [jm_log_v](#) ([jm_callbacks](#) *cb, const [char](#) *module, [jm_log_level_enu_t](#) log_level, const [char](#) *fmt, va_list ap)
Send a message to the logger function.
- [FMILIB_EXPORT](#) void [jm_log_fatal_v](#) ([jm_callbacks](#) *cb, const [char](#) *module, const [char](#) *fmt, va_list ap)
Send a fatal error message to the logger function. See [jm_log\(\)](#) for details.
- [FMILIB_EXPORT](#) void [jm_log_fatal](#) ([jm_callbacks](#) *cb, const [char](#) *module, const [char](#) *fmt,...)
Send a fatal error message to the logger function. See [jm_log\(\)](#) for details.

- **FMILIB_EXPORT** void `jm_log_error_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)
Send a error message to the logger function. See [jm_log\(\)](#) for details.
- **FMILIB_EXPORT** void `jm_log_error` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)
Send a error message to the logger function. See [jm_log\(\)](#) for details.
- **FMILIB_EXPORT** void `jm_log_warning_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)
Send a warning message to the logger function. See [jm_log\(\)](#) for details.
- **FMILIB_EXPORT** void `jm_log_warning` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)
Send a warning message to the logger function. See [jm_log\(\)](#) for details.
- **FMILIB_EXPORT** void `jm_log_info_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)
Send an info message to the logger function. See [jm_log\(\)](#) for details.
- **FMILIB_EXPORT** void `jm_log_info` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)
Send an info message to the logger function. See [jm_log\(\)](#) for details.
- **FMILIB_EXPORT** void `jm_log_verbose_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)
Send a verbose message to the logger function. See [jm_log\(\)](#) for details.
- **FMILIB_EXPORT** void `jm_log_verbose` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)
Send a verbose message to the logger function. See [jm_log\(\)](#) for details.
- static void `jm_log_debug_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)
Send a debug message to the logger function. See [jm_log\(\)](#) for details.
- static void `jm_log_debug` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)
Send a debug message to the logger function. See [jm_log\(\)](#) for details.

Memory management callbacks

`jm_malloc_f`, `jm_realloc_f`, `jm_calloc_f`, `jm_free_f` function types correspond to the standard C memory management functions

- typedef `jm_voidp`(* `jm_malloc_f`)(`size_t` size)
Allocation function type.
- typedef `jm_voidp`(* `jm_realloc_f`)(void *ptr, `size_t` size)
Re-allocation function type.
- typedef `jm_voidp`(* `jm_calloc_f`)(`size_t` numitems, `size_t` itemsize)
Zero-initialized allocation function type.
- typedef void(* `jm_free_f`)(`jm_voidp` p)
Free memory function type.

6.13.1 Define Documentation

6.13.1.1 `#define JM_MAX_ERROR_MESSAGE_SIZE 2000`

Maximum message size that can be stored in the `jm_callbacks` struct.

Definition at line 70 of file `jm_callbacks.h`.

6.13.2 Typedef Documentation

6.13.2.1 `typedef struct jm_callbacks jm_callbacks`

Definition at line 37 of file `jm_callbacks.h`.

6.13.2.2 `typedef jm_voidp(* jm_malloc_f)(size_t size)`

Allocation function type.

Definition at line 45 of file `jm_callbacks.h`.

6.13.2.3 `typedef jm_voidp(* jm_realloc_f)(void *ptr, size_t size)`

Re-allocation function type.

Definition at line 48 of file `jm_callbacks.h`.

6.13.2.4 `typedef jm_voidp(* jm_calloc_f)(size_t numitems, size_t itemsize)`

Zero-initialized allocation function type.

Definition at line 51 of file `jm_callbacks.h`.

6.13.2.5 `typedef void(* jm_free_f)(jm_voidp p)`

Free memory function type.

Definition at line 54 of file `jm_callbacks.h`.

6.13.2.6 `typedef void(* jm_logger_f)(jm_callbacks *c, jm_string module, jm_log_level_enumer_t log_level, jm_string message)`

Logger callback type.

The logger callback is used to report errors. Note that this function is by default only used in FMI standard independent code (e.g., `fmi_import_get_fmi_version()`). Since logging functions are different between different standard versions separate logging functions are necessary for each fmi implementation.

Defaults are provided for each standard.

Definition at line 67 of file `jm_callbacks.h`.

6.13.3 Function Documentation

6.13.3.1 `static jm_string jm_get_last_error ( jm_callbacks * cb ) [static]`

Get the last log message produced by the library.

An alternative way to get error information is to use `jm_get_last_error()`. This is only meaningful if logger function is not present.

Definition at line 98 of file `jm_callbacks.h`.

6.13.3.2 `static void jm_clear_last_error ( jm_callbacks * cb ) [static]`

Clear the last generated log message.

Definition at line 103 of file `jm_callbacks.h`.

6.13.3.3 `FMILIB_EXPORT void jm_set_default_callbacks ( jm_callbacks * c )`

Set the structure to be returned by `jm_get_default_callbacks()`.

Parameters

<code>c</code>	- a pointer to initialized struct to be used as default later on. If this is NULL library default implementation will be used.
----------------	--

6.13.3.4 `FMILIB_EXPORT jm_callbacks* jm_get_default_callbacks ( void )`

Get default callbacks. The function never returns NULL.

Returns

Default `jm_callbacks` struture. Either the one supplied by the library of the one set with `jm_set_default_callbacks()`.

6.13.3.5 `FMILIB_EXPORT void jm_default_logger ( jm_callbacks * c, jm_string module, jm_log_level_enu_t log_level, jm_string message )`

The default logger implementation prints messages to stderr.

6.13.3.6 FMILIB_EXPORT void jm_log (jm_callbacks * *cb*, const char * *module*, jm_log_level_enu_t *log_level*, const char * *fmt*, ...)

Send a message to the logger function.

Parameters

<i>cb</i>	- callbacks to be used for reporting;
<i>module</i>	- a name of reporting module;
<i>log_level</i>	- message kind;
<i>fmt</i>	- "printf" type of format followed by the arguments.

6.13.3.7 FMILIB_EXPORT void jm_log_v (jm_callbacks * *cb*, const char * *module*, jm_log_level_enu_t *log_level*, const char * *fmt*, va_list *ap*)

Send a message to the logger function.

Parameters

<i>cb</i>	- callbacks to be used for reporting;
<i>module</i>	- a name of reporting module;
<i>log_level</i>	- message kind;
<i>fmt</i>	- "printf" type of format followed by the arguments.

Parameters

<i>ap</i>	- variable size argument list.
-----------	--------------------------------

6.13.3.8 FMILIB_EXPORT void jm_log_fatal_v (jm_callbacks * *cb*, const char * *module*, const char * *fmt*, va_list *ap*)

Send a fatal error message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.9 FMILIB_EXPORT void jm_log_fatal (jm_callbacks * *cb*, const char * *module*, const char * *fmt*, ...)

Send a fatal error message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.10 FMILIB_EXPORT void jm_log_error_v (jm_callbacks * *cb*, const char * *module*, const char * *fmt*, va_list *ap*)

Send a error message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.11 FMILIB_EXPORT void jm_log_error (jm_callbacks * cb, const char * module, const char * fmt, ...)

Send a error message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.12 FMILIB_EXPORT void jm_log_warning_v (jm_callbacks * cb, const char * module, const char * fmt, va_list ap)

Send a warning message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.13 FMILIB_EXPORT void jm_log_warning (jm_callbacks * cb, const char * module, const char * fmt, ...)

Send a warning message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.14 FMILIB_EXPORT void jm_log_info_v (jm_callbacks * cb, const char * module, const char * fmt, va_list ap)

Send an info message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.15 FMILIB_EXPORT void jm_log_info (jm_callbacks * cb, const char * module, const char * fmt, ...)

Send an info message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.16 FMILIB_EXPORT void jm_log_verbose_v (jm_callbacks * cb, const char * module, const char * fmt, va_list ap)

Send a verbose message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.17 FMILIB_EXPORT void jm_log_verbose (jm_callbacks * cb, const char * module, const char * fmt, ...)

Send a verbose message to the logger function. See [jm_log\(\)](#) for details.

6.13.3.18 static void jm_log_debug_v (jm_callbacks * cb, const char * module, const char * fmt, va_list ap) [static]

Send a debug message to the logger function. See [jm_log\(\)](#) for details.

Note that the function is only active if the library is configure with FMILIB_ENABLE_LOG_LEVEL_DEBUG=ON

Definition at line 208 of file jm_callbacks.h.

6.13.3.19 `static void jm_log_debug ( jm_callbacks * cb, const char * module, const char * fmt, ... ) [static]`

Send a debug message to the logger function. See [jm_log\(\)](#) for details.

Note that the function is only active if the library is configure with `FMILIB_ENABLE_LOG_LEVEL_DEBUG=ON`

Definition at line 213 of file `jm_callbacks.h`.

6.14 Named objects

Data Structures

- struct `jm_named_ptr`
Name and object pointer pair.

Typedefs

- typedef struct `jm_named_ptr` `jm_named_ptr`
Name and object pointer pair.

Functions

- `jm_named_ptr jm_named_alloc` (`jm_string` name, `size_t` size, `size_t` nameoffset, `jm_callbacks` *c)
Allocate memory for the object and the name string and sets pointer to it packed together with the name pointer.
- `jm_named_ptr jm_named_alloc_v` (`jm_vector(char)*`name, `size_t` size, `size_t` nameoffset, `jm_callbacks` *c)
Same as `jm_named_alloc()` but name is given as a `jm_vector(char)` pointer.
- static void `jm_named_free` (`jm_named_ptr` np, `jm_callbacks` *c)
Free the memory allocated for the object pointed by `jm_named_ptr`.
- `jm_vector_declare_template` (`jm_named_ptr`) `jm_define_comp_f`(`jm_compare_named`)
Helper to construct comparison operation.
- static `jm_diff_named` void `jm_named_vector_free_data` (`jm_vector(jm_named_ptr)*`v)
Release the data allocated by the items in a vector and then clears the memory used by the vector as well.
- static void `jm_named_vector_free` (`jm_vector(jm_named_ptr)*`v)
Release the data allocated by the items in a vector and then clears the memory used by the vector as well.

6.14.1 Typedef Documentation

6.14.1.1 `jm_named_ptr`

Name and object pointer pair.

Definition at line 36 of file `jm_named_ptr.h`.

6.14.2 Function Documentation

6.14.2.1 `jm_named_ptr jm_named_alloc ( jm_string name, size_t size, size_t nameoffset, jm_callbacks * c )`

Allocate memory for the object and the name string and sets pointer to it packed together with the name pointer.

Parameters

<i>name</i>	Name for the object.
<i>size</i>	Size of the data structure.
<i>nameoffset</i>	Offset of the name field within the data structure.
<i>c</i>	Callbacks to be used for memory allocation.

The function `jm_named_alloc()` is intended for types defined as:

```
struct T {
    < some data fields>
    char name[1];
}
```

The "name" is copied into the allocated memory.

6.14.2.2 `jm_named_ptr jm_named_alloc_v ( jm_vector(char)* name, size_t size, size_t nameoffset, jm_callbacks * c )`

Same as `jm_named_alloc()` but name is given as a `jm_vector(char)` pointer.

6.14.2.3 `static void jm_named_free ( jm_named_ptr np, jm_callbacks * c )` [static]

Free the memory allocated for the object pointed by `jm_named_ptr`.

Definition at line 66 of file `jm_named_ptr.h`.

6.14.2.4 `jm_vector_declare_template ( jm_named_ptr )`

Helper to construct comparison operation.

6.14.2.5 `static jm_diff_named void jm_named_vector_free_data ( jm_vector(jm_named_ptr)* v )` [static]

Release the data allocated by the items in a vector and then clears the memory used by the vector as well.

This should be used for vectors initialized with `jm_vector_init`.

Definition at line 80 of file `jm_named_ptr.h`.

6.14.2.6 `static void jm_named_vector_free ( jm_vector(jm_named_ptr)* v )`
`[static]`

Release the data allocated by the items in a vector and then clears the memory used by the vector as well.

This should be used for vectors created with `jm_vector_alloc`.

Definition at line 90 of file `jm_named_ptr.h`.

6.15 Handling platform specific defines and functions

Typedefs

- `typedef void * jm_dll_function_ptr`
A function pointer as returned when DLL symbol is loaded.

Functions

- `DLL_HANDLE jm_portability_load_dll_handle` (const `char` *dll_file_path)
Load a dll/so library into the process and return a handle.
- `jm_status_enu_t jm_portability_free_dll_handle` (DLL_HANDLE dll_handle)
Unload a Dll and release the handle.
- `jm_status_enu_t jm_portability_load_dll_function` (DLL_HANDLE dll_handle, `char` *dll_function_name, `jm_dll_function_ptr` *dll_function_ptrptr)
Find a function in the Dll and return a function pointer.
- `char * jm_portability_get_last_dll_error` (void)
Return error associated with Dll handling.
- `jm_status_enu_t jm_portability_get_current_working_directory` (`char` *buffer, `size_t` len)
Get current working directory name.
- `jm_status_enu_t jm_portability_set_current_working_directory` (const `char` *cwd)
Set current working directory.
- `const char * jm_get_system_temp_dir` ()
Get system-wide temporary directory.
- `char * jm_mktemp` (`char` *tmpl)
- `char * jm_get_dir_abspath` (jm_callbacks *cb, const `char` *dir, `char` *outPath, `size_t` len)
Create a unique file name.
- `char * jm_get_dir_abspath` (jm_callbacks *cb, const `char` *dir, `char` *outPath, `size_t` len)
Get absolute path to an existing directory.
- `char * jm_mk_temp_dir` (jm_callbacks *cb, const `char` *systemTempDir, const `char` *tempPrefix)
Create a unique temporary directory.
- `char * jm_create_URL_from_abs_path` (jm_callbacks *cb, const `char` *absPath)
Create a file:// URL from absolute path.
- `jm_status_enu_t jm_mkdir` (jm_callbacks *cb, const `char` *dir)
Make a directory.
- `jm_status_enu_t jm_rmdir` (jm_callbacks *cb, const `char` *dir)
Remove directory and all it contents.
- `FMLIB_EXPORT int jm_vsnprintf` (`char` *str, `size_t` size, const `char` *fmt, va_list al)
C89 compatible implementation of C99 vsnprintf.
- `FMLIB_EXPORT int jm_snprintf` (`char` *str, `size_t` size, const `char` *fmt,...)
C89 compatible implementation of C99 snprintf.

6.15.1 Typedef Documentation

6.15.1.1 `typedef void* jm_dll_function_ptr`

A function pointer as returned when DLL symbol is loaded.

Definition at line 52 of file `jm_portability.h`.

6.15.2 Function Documentation

6.15.2.1 `DLL_HANDLE jm_portability_load_dll_handle ( const char * dll_file_path )`

Load a dll/so library into the process and return a handle.

6.15.2.2 `jm_status_enu_t jm_portability_free_dll_handle ( DLL_HANDLE dll_handle )`

Unload a DLL and release the handle.

6.15.2.3 `jm_status_enu_t jm_portability_load_dll_function ( DLL_HANDLE dll_handle, char * dll_function_name, jm_dll_function_ptr * dll_function_ptrptr )`

Find a function in the DLL and return a function pointer.

6.15.2.4 `char* jm_portability_get_last_dll_error ( void )`

Return error associated with DLL handling.

6.15.2.5 `jm_status_enu_t jm_portability_get_current_working_directory ( char * buffer, size_t len )`

Get current working directory name.

6.15.2.6 `jm_status_enu_t jm_portability_set_current_working_directory ( const char * cwd )`

Set current working directory.

6.15.2.7 `const char* jm_get_system_temp_dir ( )`

Get system-wide temporary directory.

6.15.2.8 `char* jm_mktemp ( char * tmpl )`

Create a unique file name.

Parameters

<i>tmpl</i>	File name template ending with XXXXXX.
-------------	--

Returns

A pointer to the modified template. The function returns NULL if template is badly formed or no more unique names can be created from the given template.

6.15.2.9 `char* jm_get_dir_abspath ( jm_callbacks * cb, const char * dir, char * outPath, size_t len )`

Get absolute path to an existing directory.

Parameters

<i>cb</i>	- callbacks for memory allocation and logging. Default callbacks are used if this parameter is NULL.
<i>dir</i>	- path to a directory (relative or absolute).
<i>outPath</i>	- buffer for storing the directory
<i>len</i>	- of the buffer (if size is larger than FILENAME_MAX + 1 then the path will always fit in)

Returns

Pointer to *outPath* on success, 0 - on error in which case a message is send to the logger.

6.15.2.10 `char* jm_mk_temp_dir ( jm_callbacks * cb, const char * systemTempDir, const char * tempPrefix )`

Create a unique temporary directory.

Parameters

<i>cb</i>	- callbacks for memory allocation and logging. Default callbacks are used if this parameter is NULL.
<i>systemTempDir</i>	- directory where the temp dir should be located both absolute and relative path are accepted. System-wide directory is used if this parameter is NULL.
<i>tempPrefix</i>	- File name template prefix used when creating temporary directories. "jm" is used if this is NULL.

Returns

A pointer to the temporary directory name (absolute path, no terminating '/'). Caller is responsible for freeing the memory. The function returns NULL if there were errors in which case a message is send to the logger.

6.15.2.11 `char* jm_create_URL_from_abs_path ( jm_callbacks * cb, const char * absPath )`

Create a `file://` URL from absolute path.

Parameters

<i>cb</i>	- callbacks for memory allocation and logging. Default callbacks are used if this parameter is NULL.
<i>absPath</i>	- absolute path to be converted into the URL

Returns

A pointer to the URL. Caller is responsible for freeing the memory. The function returns NULL if there were errors in which case a message is send to the logger.

6.15.2.12 `jm_status_enu_t jm_mkdir ( jm_callbacks * cb, const char * dir )`

Make a directory.

6.15.2.13 `jm_status_enu_t jm_rmdir ( jm_callbacks * cb, const char * dir )`

Remove directory and all it contents.

6.15.2.14 `FMILIB_EXPORT int jm_vsnprintf ( char * str, size_t size, const char * fmt, va_list al )`

C89 compatible implementation of C99 vsnprintf.

6.15.2.15 `FMILIB_EXPORT int jm_snprintf ( char * str, size_t size, const char * fmt, ... )`

C89 compatible implementation of C99 snprintf.

6.16 A set of strings

Typedefs

- typedef struct jm_vector_jm_string jm_string_set
Set of string is based on a vector.

Functions

- static jm_string jm_string_set_find (jm_string_set *s, jm_string str)
Find a string in a set.
- static jm_string jm_string_set_put (jm_string_set *s, jm_string str)
Put an element in the set if it is not there yet.

6.16.1 Typedef Documentation

6.16.1.1 typedef struct jm_vector_jm_string jm_string_set

Set of string is based on a vector.

Definition at line 42 of file jm_string_set.h.

6.16.2 Function Documentation

6.16.2.1 static jm_string jm_string_set_find (jm_string_set * s, jm_string str) [static]

Find a string in a set.

Parameters

<i>s</i>	A string set.
<i>str</i>	Search string.

Returns

If found returns a pointer to the string saved in the set. If not found returns NULL.

Definition at line 51 of file jm_string_set.h.

6.16.2.2 static jm_string jm_string_set_put (jm_string_set * s, jm_string str) [static]

Put an element in the set if it is not there yet.

Parameters

<i>s</i>	A string set.
<i>str</i>	String to put.

Returns

A pointer to the inserted (or found) element or zero pointer if failed.

Definition at line 64 of file jm_string_set.h.

6.17 A vector of items (dynamic array)

Defines

- #define `jm_mangle_ex`(name, type) name##_ ##type
jm_mangle macro is used to construct names for the template instances Extra level (jm_mangle_ex) is needed to force argument expansion (pre-scan)
- #define `jm_mangle`(name, type) `jm_mangle_ex`(name,type)
- #define `jm_vector`(T) `jm_mangle`(jm_vector, T)
jm_vector(T) is the type name (i.e., to be used as `jm_vector(int) vi;`)
- #define `JM_VECTOR_MINIMAL_CAPACITY` 16
- #define `JM_VECTOR_MAX_MEMORY_CHUNK` 1024
- #define `jm_vector_declare_template`(T)

Functions

- `jm_vector_declare_template` (char) static `jm_string` `jm_vector_char2string`(jm_vector(char)*v)
- `jm_vector_declare_template` (int) `jm_vector_declare_template`(double) `jm_vector_declare_template`(jm_voidp) `jm_vector_declare_template`(size_t) `jm_vector_declare_template`(jm_string) `jm_vector_declare_template`(jm_name_ID_map_t) `jm_define_comp_f`(jm_compare_voidp)
- int `jm_diff` `jm_define_comp_f` (jm_compare_int, int, `jm_diff`) `jm_define_comp_f`(jm_compare_char)
- int `jm_diff` `jm_diff` `jm_define_comp_f` (jm_compare_double, double, `jm_diff`) `jm_define_comp_f`(jm_compare_size_t)
- int `jm_diff` `jm_diff` `jm_diff` `jm_define_comp_f` (jm_compare_string, `jm_string`, strcmp) `jm_define_comp_f`(jm_compare_name)
- `jm_vector` (JM_TEMPLATE_INSTANCE_TYPE)*`jm_vector_alloc`(JM_TEMPLATE_INSTANCE_TYPE)(size_t size)

Variables

- int `jm_diff` char
- int `jm_diff` `jm_diff` size_t
- int `jm_diff` `jm_diff` `jm_diff` `jm_name_ID_map_t`
- int `jm_diff` `jm_diff` `jm_diff` `jm_diff_name`
- size_t capacity

Vector handling functions.

Allocates a vector on heap with the specified size and specified number of preallocated items (can be larger than size).

extern `jm_vector`(T)* `jm_vector_alloc`(T)(size_t size, size_t capacity, jm_callbacks*c);
 Note that there is no need to call `jm_vector_init` for a vector allocated with this function.

Parameters

<i>size</i>	- initial size of the vector, can be 0
<i>capacity</i>	- initial capacity of the vector, can be 0. At least <i>initSize</i> elements are allocated.
<i>c</i>	- jm_callbacks callbacks, can be zero

Returns

Newly allocated vector

- `typedef int(* jm\_compare\_ft )(const void *, const void *)`
Function type for item comparison. Can be generated with [jm_define_comp_f](#).
- `#define jm\_vector\_alloc(T) jm\_mangle(jm\_vector\_alloc, T)`
- `#define jm\_vector\_free(T) jm\_mangle(jm\_vector\_free, T)`
- `#define jm\_vector\_init(T) jm\_mangle(jm\_vector\_init, T)`
[jm_vector_init](#) initializes a vector allocated on stack.
- `#define jm\_vector\_free\_data(T) jm\_mangle(jm\_vector\_free\_data, T)`
- `#define jm\_vector\_get\_size(T) jm\_mangle(jm\_vector\_get\_size, T)`
- `#define jm\_vector\_get\_item(T) jm\_mangle(jm\_vector\_get\_item, T)`
- `#define jm\_vector\_get\_itemp(T) jm\_mangle(jm\_vector\_get\_itemp, T)`
- `#define jm\_vector\_get\_last(T) jm\_mangle(jm\_vector\_get\_last, T)`
- `#define jm\_vector\_get\_lastp(T) jm\_mangle(jm\_vector\_get\_lastp, T)`
- `#define jm\_define\_comp\_f(F, T, COMPAR_OP)`
A convenience macro for comparison function definition.
- `#define jm\_diff(first, second) (int)(first-second)`
- `#define jm\_vector\_find(T) jm\_mangle(jm\_vector\_find, T)`
[jm_vector_find](#) functions use linear search to find items in a vector. [JM_COMPAR_OP](#) is used for comparison.
- `#define jm\_vector\_find\_index(T) jm\_mangle(jm\_vector\_find\_index, T)`
- `#define jm\_vector\_qsort(T) jm\_mangle(jm\_vector\_qsort, T)`
- `#define jm\_vector\_bsearch(T) jm\_mangle(jm\_vector\_bsearch, T)`
- `#define jm\_vector\_bsearch\_index(T) jm\_mangle(jm\_vector\_bsearch\_index, T)`
- `#define jm\_vector\_set\_item(T) jm\_mangle(jm\_vector\_set\_item, T)`
- `#define jm\_vector\_zero(T) jm\_mangle(jm\_vector\_zero, T)`
- `#define jm\_vector\_resize(T) jm\_mangle(jm\_vector\_resize, T)`
- `#define jm\_vector\_reserve(T) jm\_mangle(jm\_vector\_reserve, T)`
- `#define jm\_vector\_copy(T) jm\_mangle(jm\_vector\_copy, T)`
- `#define jm\_vector\_clone(T) jm\_mangle(jm\_vector\_clone, T)`
- `#define jm\_vector\_append(T) jm\_mangle(jm\_vector\_append, T)`
- `#define jm\_vector\_insert(T) jm\_mangle(jm\_vector\_insert, T)`
- `#define jm\_vector\_remove\_item(T) jm\_mangle(jm\_vector\_remove\_item, T)`
- `#define jm\_vector\_resize1(T) jm\_mangle(jm\_vector\_resize1, T)`
- `#define jm\_vector\_push\_back(T) jm\_mangle(jm\_vector\_push\_back, T)`
- `#define jm\_vector\_foreach(T) jm\_mangle(jm\_vector\_foreach, T)`
- `#define jm\_vector\_foreach\_c(T) jm\_mangle(jm\_vector\_foreach\_c, T)`

6.17.1 Define Documentation

6.17.1.1 `#define jm_mangle_ex( name, type ) name##_##type`

`jm_mange` macro is used to construct names for the template instances Extra level (`jm_mangle_ex`) is needed to force argument expansion (pre-scan)

Definition at line 44 of file `jm_vector.h`.

6.17.1.2 `#define jm_mangle( name, type ) jm_mangle_ex(name,type)`

Definition at line 45 of file `jm_vector.h`.

6.17.1.3 `#define jm_vector( T ) jm_mangle(jm_vector, T)`

`jm_vector(T)` is the type name (i.e., to be used as `jm_vector(int) vi`;

Definition at line 48 of file `jm_vector.h`.

6.17.1.4 `#define jm_vector_alloc( T ) jm_mangle(jm_vector_alloc, T)`

Definition at line 62 of file `jm_vector.h`.

6.17.1.5 `#define jm_vector_free( T ) jm_mangle(jm_vector_free, T)`

`jm_vector_free` releases the memory allocated by `jm_vector_alloc`. `extern void jm_vector_free(T)(jm_vector(T)* a);`

Definition at line 68 of file `jm_vector.h`.

6.17.1.6 `#define jm_vector_init( T ) jm_mangle(jm_vector_init, T)`

`jm_vector_init` initializes a vector allocated on stack.

Input: `a` - pointer to the vector to be initialized; `size` - initial size of the vector, can be 0 `c` - `jm_callbacks` callbacks, can be zero Returns: size of the vector (can be zero for non-zero size if memory allocation failed) Note that for `initSize < JM_VECTOR_MINIMAL_CAPACITY` no heap memory allocation is needed `size_t jm_vector_init(T)(jm_vector(T)* a, size_t initSize, jm_callbacks* c)`

Definition at line 82 of file `jm_vector.h`.

6.17.1.7 `#define jm_vector_free_data( T ) jm_mangle(jm_vector_free_data, T)`

`jm_vector_free_data` releases memory allocated for vector data This only needs to be called for stack allocated vectors (`jm_vector_free` does the job for heap vectors automatically) inline void `jm_vector_free_data(T)(jm_vector(T)* a)`

Definition at line 90 of file jm_vector.h.

6.17.1.8 #define jm_vector_get_size(T) jm_mangle(jm_vector_get_size, T)

jm_vector_get_size get the vector size inline size_t jm_vector_get_size(T)(jm_vector(-T)* a)

Definition at line 96 of file jm_vector.h.

6.17.1.9 #define jm_vector_get_item(T) jm_mangle(jm_vector_get_item, T)

jm_vector_get_item returns the specified item. Range checking is done with an assert. inline T jm_vector_get_item(jm_vector(T)* a, size_t index)

Definition at line 102 of file jm_vector.h.

6.17.1.10 #define jm_vector_get_itemp(T) jm_mangle(jm_vector_get_itemp, T)

jm_vector_get_itemp returns a pointer to the specified item. Range checking is done with an assert. inline T* jm_vector_get_itemp(jm_vector(T)* a, size_t index)

Definition at line 108 of file jm_vector.h.

6.17.1.11 #define jm_vector_get_last(T) jm_mangle(jm_vector_get_last, T)

jm_vector_get_lastp returns a pointer to the last item in the vector. It is an error to call this if size=0 inline T jm_vector_get_last(jm_vector(T)* a)

Definition at line 115 of file jm_vector.h.

6.17.1.12 #define jm_vector_get_lastp(T) jm_mangle(jm_vector_get_lastp, T)

jm_vector_get_lastp returns a pointer to the last item in the vector. Zero pointer is returned if size=0 inline T* jm_vector_get_lastp(jm_vector(T)* a)

Definition at line 121 of file jm_vector.h.

6.17.1.13 #define jm_define_comp_f(F, T, COMPAR.OP)

Value:

```
static int F (const void* first, const void* second) { \
    return COMPAR_OP( (*(T*)first), (*(T*)second)); \
}
```

A convenience macro for comparison function definition.

#define jm_define_comp_f(F, T, COMPAR_OP) is a convenience macro for comparison function definition to be used in sort/search operations. Here F - is the defined function

name; T - type of the argument; COMPAR_OP(A,B) is a macro that returns an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second. If two members compare as equal, their order in the sorted array is undefined. Default definition below is jm_diff and is implemented as (int)(first-second)

Definition at line 140 of file jm_vector.h.

6.17.1.14 `#define jm_diff( first, second ) (int)(first-second)`

Definition at line 145 of file jm_vector.h.

6.17.1.15 `#define jm_vector_find( T ) jm_mangle(jm_vector_find, T)`

jm_vector_find functions use linear search to find items in a vector. JM_COMPAR_OP is used for comparison.

T* [jm_vector_find\(T\)\(jm_vector\(T\)* a, T item, jm_compare_ft f\)](#)

size_t [jm_vector_find_index\(T\)\(jm_vector\(T\)* a, T item, jm_compare_ft f\)](#)

Parameters

<i>a</i>	- the vector;
<i>item</i>	- the searched item;

Return: T* [jm_vector_find\(T\)\(jm_vector\(T\)* a, T item, jm_compare_ft f\)](#) returns a pointer to the found item or NULL if not found size_t [jm_vector_find_index\(T\)\(jm_vector\(T\)* a, T item, jm_compare_ft f\)](#) return the index of the found item or size of the vector if not found.

Definition at line 161 of file jm_vector.h.

6.17.1.16 `#define jm_vector_find_index( T ) jm_mangle(jm_vector_find_index, T)`

Definition at line 162 of file jm_vector.h.

6.17.1.17 `#define jm_vector_qsort( T ) jm_mangle(jm_vector_qsort, T)`

Definition at line 170 of file jm_vector.h.

6.17.1.18 `#define jm_vector_bsearch( T ) jm_mangle(jm_vector_bsearch, T)`

jm_vector_bsearch uses standard binary search (bsearch) to find elements in a sorted vector. It returns the index of an item in the vector or vector's size if not found. JM_COMPAR_OP is used for comparison.

T* [jm_vector_bsearch\(T\)\(jm_vector\(T\)* v, T* key, jm_compare_ft f\)](#) size_t [jm_vector_bsearch_index\(T\)\(jm_vector\(T\)* v, T* key, jm_compare_ft f\)](#)

Definition at line 180 of file jm_vector.h.

```
6.17.1.19 #define jm_vector_bsearch_index( T ) jm_mangle(jm_vector_bsearch_index, T)
```

Definition at line 181 of file jm_vector.h.

```
6.17.1.20 #define jm_vector_set_item( T ) jm_mangle(jm_vector_set_item, T)
```

jm_vector_set_item sets the specified item. Range checking is done with an assert.
void [jm_vector_set_item\(T\)\(jm_vector\(T\)* a, size_t index, T item\)](#)

Definition at line 187 of file jm_vector.h.

```
6.17.1.21 #define jm_vector_zero( T ) jm_mangle(jm_vector_zero, T)
```

jm_vector_zero sets all elements in the vector to zero void [jm_vector_zero\(T\)\(jm_vector\(T\)* a\)](#);

Definition at line 193 of file jm_vector.h.

```
6.17.1.22 #define jm_vector_resize( T ) jm_mangle(jm_vector_resize, T)
```

jm_vector_resize resizes the vector Input: a - the vector size - new size Return: size of the vector after operation. Can be less than size if memory allocation failed. Note: resizing to smaller vector does not release memory. size_t [jm_vector_resize\(T\)\(jm_vector\(T\)* a, size_t size\)](#)

Definition at line 205 of file jm_vector.h.

```
6.17.1.23 #define jm_vector_reserve( T ) jm_mangle(jm_vector_reserve, T)
```

jm_vector_reserve preallocates memory for the vector (to speed up consequent push_back) Returns: the actually reserved space. Can be smaller than requested "capacity" if memory allocation failed. Can be larger than "capacity" if more memory was previously allocated. size_t [jm_vector_reserve\(T\)\(jm_vector\(T\)* a, size_t capacity\)](#)

Definition at line 213 of file jm_vector.h.

```
6.17.1.24 #define jm_vector_copy( T ) jm_mangle(jm_vector_copy, T)
```

jm_vector_copy copies source vector into destination. Returns the number of elements actually copied (may be less than the source size if allocation failed). size_t [jm_vector_copy\(T\)\(jm_vector\(T\)* destination, jm_vector\(T\)* source\)](#)

Definition at line 220 of file jm_vector.h.

6.17.1.25 `#define jm_vector_clone( T ) jm_mangle(jm_vector_clone, T)`

`jm_vector_clone` creates a copy of the provided vector on heap and returns it. Allocated capacity matches the size of the given vector. Returns the vector pointer or zero if memory allocation failed. `jm_vector(T)* jm_vector_clone(T)(jm_vector(T)* source)`

Definition at line 227 of file `jm_vector.h`.

6.17.1.26 `#define jm_vector_append( T ) jm_mangle(jm_vector_append, T)`

`jm_vector_append` appends source vector into destination. Returns the number of elements actually appended (may be less than the source size if allocation failed). `size_t jm_vector_append(T)(jm_vector(T)* destination, jm_vector(T)* source)`

Definition at line 234 of file `jm_vector.h`.

6.17.1.27 `#define jm_vector_insert( T ) jm_mangle(jm_vector_insert, T)`

`jm_vector_insert` inserts an element at a given location. Returns a pointer to the inserted element or zero pointer if failed `T* jm_vector_insert(T)(jm_vector(T)* a, size_t index, T item)`

Definition at line 241 of file `jm_vector.h`.

6.17.1.28 `#define jm_vector_remove_item( T ) jm_mangle(jm_vector_remove_item, T)`

`jm_vector_remove_item` removes an item from the vector. Vector size is reduced by 1. Supplying index > size gives assertion fault. `void jm_vector_remove_item(T)(jm_vector(T)* v, size_t index)`

Definition at line 248 of file `jm_vector.h`.

6.17.1.29 `#define jm_vector_resize1( T ) jm_mangle(jm_vector_resize1, T)`

`T* jm_vector_resize1(jm_vector(T)* a)` Increase the size of the vector by 1 and return a pointer to the last item. Return 0 if memory allocation failed.

Definition at line 255 of file `jm_vector.h`.

6.17.1.30 `#define jm_vector_push_back( T ) jm_mangle(jm_vector_push_back, T)`

`jm_vector_push_back` Returns a pointer to the inserted element or zero pointer if failed. `T* jm_vector_push_back(jm_vector(T)* a, T item)`

Definition at line 262 of file `jm_vector.h`.

6.17.1.31 `#define jm_vector_foreach( T ) jm_mangle(jm_vector_foreach, T)`

`jm_vector_foreach` calls `f` for each element in the vector. "context" parameter is passed directly to the function as the second argument for the second version. `void jm_vector_foreach(T)(jm_vector(T)* a, void (*f)(T)) void jm_vector_foreach_c(T)(jm_vector(T)* a, void (*f)(T, void*), void * context)`

Definition at line 270 of file `jm_vector.h`.

6.17.1.32 `#define jm_vector_foreach_c( T ) jm_mangle(jm_vector_foreach_c, T)`

Definition at line 271 of file `jm_vector.h`.

6.17.1.33 `#define JM_VECTOR_MINIMAL_CAPACITY 16`

number of items always allocated on the stack

Definition at line 276 of file `jm_vector.h`.

6.17.1.34 `#define JM_VECTOR_MAX_MEMORY_CHUNK 1024`

maximum memory chunk (in items) to be allocated in `push_back`.

Definition at line 279 of file `jm_vector.h`.

6.17.1.35 `#define jm_vector_declare_template( T )`

Declare the struct and functions for the specified type.

Definition at line 282 of file `jm_vector.h`.

6.17.2 Typedef Documentation

6.17.2.1 `typedef int(* jm_compare_ft)(const void *, const void *)`

Function type for item comparison. Can be generated with `jm_define_comp_f`.

Definition at line 127 of file `jm_vector.h`.

6.17.3 Function Documentation

6.17.3.1 `jm_vector_declare_template( char )`

Definition at line 352 of file `jm_vector.h`.

6.17.3.2 `jm_vector_declare_template ( int )`

6.17.3.3 `int jm_diff jm_define_comp_f ( jm_compare_int , int , jm_diff )`

6.17.3.4 `int jm_diff jm_diff jm_define_comp_f ( jm_compare_double , double , jm_diff )`

6.17.3.5 `int jm_diff jm_diff jm_diff jm_define_comp_f ( jm_compare_string , jm_string , strcmp )`

6.17.3.6 `jm_vector ( JM_TEMPLATE_INSTANCE_TYPE )`

6.17.4 Variable Documentation

6.17.4.1 `int jm_diff char`

Definition at line 369 of file `jm_vector.h`.

6.17.4.2 `int jm_diff jm_diff size_t`

Definition at line 371 of file `jm_vector.h`.

6.17.4.3 `int jm_diff jm_diff jm_diff jm_name_ID_map_t`

Definition at line 375 of file `jm_vector.h`.

6.17.4.4 `int jm_diff jm_diff jm_diff jm_diff_name`

Definition at line 375 of file `jm_vector.h`.

6.17.4.5 `size_t capacity`

Definition at line 37 of file `jm_vector_template.h`.

6.18 Library initialization

Typedefs

- typedef struct fmi_xml_context_t fmi_import_context_t
FMI version independent library context. Opaque struct returned from [fmi_import_allocate_context\(\)](#)
- typedef struct fmi1_import_t fmi1_import_t
FMU version 1.0 object.
- typedef struct fmi2_import_t fmi2_import_t
FMU version 2.0 object.

Functions

- FMILIB_EXPORT fmi_import_context_t * fmi_import_allocate_context (jm_callbacks *callbacks)
Create fmi_import_context_t structure.
- FMILIB_EXPORT void fmi_import_free_context (fmi_import_context_t *c)
Free memory allocated for the library context.
- FMILIB_EXPORT fmi_version_enumeration_t fmi_import_get_fmi_version (fmi_import_context_t *c, const char *fileName, const char *dirName)
Unzip an FMU specified by the fileName into directory dirName and parse XML to get FMI standard version.
- FMILIB_EXPORT fmi1_import_t * fmi1_import_parse_xml (fmi_import_context_t *c, const char *dirName)
Parse FMI 1.0 XML file found in the directory dirName.
- FMILIB_EXPORT fmi2_import_t * fmi2_import_parse_xml (fmi_import_context_t *context, const char *dirPath, fmi2_xml_callbacks_t *xml_callbacks)
Create fmi2_import_t structure and parse the FMI 2.0 XML file found in the directory dirName.

6.18.1 Detailed Description

Interaction with an FMU by means of the FMI Library starts with allocation of an [fmi_import_context_t](#) structure. This is done with a call to [fmi_import_allocate_context\(\)](#). The next step is detection of FMI standard used in the specific FMU. This is achieved by calling [fmi_import_get_fmi_version\(\)](#) function. When the standard is known a standard specific function for processing model description XML should be called to create an opaque FMU structure. This is done by calling either [fmi1_import_parse_xml\(\)](#) or [fmi2_import_parse_xml\(\)](#). With the FMU structure available one can proceed by loading the FMU binary ([fmi1_import_create_dllfmu\(\)](#) or [fmi2_import_create_dllfmu\(\)](#)). After that the code is able to interact with the FMU by means of the methods presented in [Interface to the standard FMI 1.0 "C" API](#) and [Interface to the standard FMI 2.0 "C" API](#).

6.18.2 Typedef Documentation

6.18.2.1 typedef struct fmi_xml_context_t fmi_import_context_t

FMI version independent library context. Opaque struct returned from [fmi_import_allocate_context\(\)](#)

Definition at line 65 of file fmi_import_context.h.

6.18.2.2 typedef struct fmi1_import_t fmi1_import_t

FMU version 1.0 object.

Definition at line 90 of file fmi_import_context.h.

6.18.2.3 typedef struct fmi2_import_t fmi2_import_t

FMU version 2.0 object.

Definition at line 95 of file fmi_import_context.h.

6.18.3 Function Documentation

6.18.3.1 FMILIB_EXPORT fmi_import_context_t* fmi_import_allocate_context (jm_callbacks * callbacks)

Create fmi_import_context_t structure.

Parameters

<i>callbacks</i>	- a pointer to the library callbacks for memory management and logging. May be NULL if defaults are utilized.
------------------	---

Returns

A new structure if memory allocation was successful.

6.18.3.2 FMILIB_EXPORT void fmi_import_free_context (fmi_import_context_t * c)

Free memory allocated for the library context.

Parameters

<i>c</i>	- library context allocated by fmi_import_allocate_context()
----------	--

6.18.3.3 FMILIB_EXPORT fmi_version_enu_t fmi_import_get_fmi_version (fmi_import_context_t * c, const char * fileName, const char * dirName)

Unzip an FMU specified by the fileName into directory dirName and parse XML to get FMI standard version.

Parameters

<i>c</i>	- library context.
<i>fileName</i>	- an FMU file name.
<i>dirName</i>	- a directory name where the FMU should be unpacked

6.18.3.4 FMILIB_EXPORT fmi1_import_t* fmi1_import_parse_xml (fmi_import_context_t * c, const char * dirName)

Parse FMI 1.0 XML file found in the directory dirName.

Parameters

<i>c</i>	- library context.
<i>dirName</i>	- a directory where the FMU was unpacked and XML file is present.

Returns

fmi1_import_t:: opaque object pointer

6.18.3.5 FMILIB_EXPORT fmi2_import_t* fmi2_import_parse_xml (fmi_import_context_t * context, const char * dirPath, fmi2_xml_callbacks_t * xml_callbacks)

Create [fmi2_import_t](#) structure and parse the FMI 2.0 XML file found in the directory dirName.

Parameters

<i>context</i>	- library context.
<i>dirPath</i>	- a directory where the FMU was unpacked and XML file is present.
<i>xml_callbacks</i>	Callbacks to use for processing of annotations (may be NULL).

Returns

fmi2_import_t:: opaque object pointer

6.19 Utility functions supporting interactions with the library

Functions

- **FMILIB_EXPORT** `char * fmi_import_mk_temp_dir (jm_callbacks *cb, const char *systemTempDir, const char *tempPrefix)`
Create a unique temporary directory.
- **FMILIB_EXPORT** `jm_status_enu_t fmi_import_rmdir (jm_callbacks *cb, const char *dir)`
Remove directory and all its contents.
- **FMILIB_EXPORT** `char * fmi_import_create_URL_from_abs_path (jm_callbacks *cb, const char *absPath)`
Create a `file://` URL from absolute path.
- **FMILIB_EXPORT** `char * fmi_import_get_dll_path (const char *fmu_unzipped_path, const char *model_identifier, jm_callbacks *callBackFunctions)`
- **FMILIB_EXPORT** `char * fmi_import_get_model_description_path (const char *fmu_unzipped_path, jm_callbacks *callBackFunctions)`

6.19.1 Function Documentation

6.19.1.1 **FMILIB_EXPORT** `char* fmi_import_mk_temp_dir ( jm_callbacks * cb, const char * systemTempDir, const char * tempPrefix )`

Create a unique temporary directory.

Parameters

<i>cb</i>	- callbacks for memory allocation and logging. Default callbacks are used if this parameter is NULL.
<i>system-TempDir</i>	- directory where the temp dir should be located both absolute and relative path are accepted. System-wide directory is used if this parameter is NULL.
<i>tempPrefix</i>	- File name template prefix used when creating temporary directories. "fmi" is used if this is NULL.

Returns

A pointer to the temporary directory name (absolute path, no terminating '/'). Caller is responsible for freeing the memory. The function returns NULL if there were errors in which case a message is sent to the logger.

6.19.1.2 **FMILIB_EXPORT** `jm_status_enu_t fmi_import_rmdir ( jm_callbacks * cb, const char * dir )`

Remove directory and all its contents.

Parameters

<i>cb</i>	- callbacks for memory allocation and logging. Default callbacks are used if this parameter is NULL.
<i>dir</i>	- path to be removed.

Returns

Statuc success or error.

6.19.1.3 FMILIB_EXPORT char* fmi_import_create_URL_from_abs_path (jm_callbacks * *cb*, const char * *absPath*)

Create a [file://](#) URL from absolute path.

Parameters

<i>cb</i>	- callbacks for memory allocation and logging. Default callbacks are used if this parameter is NULL.
<i>absPath</i>	- absolute path to be converted into the URL

Returns

A pointer to the URL. Caller is responsible for freeing the memory. The function returns NULL if there were errors in which case a message is send to the logger.

6.19.1.4 FMILIB_EXPORT char* fmi_import_get_dll_path (const char * *fmu_unzipped_path*, const char * *model_identifier*, jm_callbacks * *callBackFunctions*)

Given directory name *fmu_unzipped_path* and model identifier consturct Dll/so name

Returns

Pointer to a string with the file name. Caller is responsible for freeing the memory.

6.19.1.5 FMILIB_EXPORT char* fmi_import_get_model_description_path (const char * *fmu_unzipped_path*, jm_callbacks * *callBackFunctions*)

Given directory name *fmu_unzipped_path* and model identifier consturct XML file name

Returns

Pointer to a string with the file name. Caller is responsible for freeing the memory.

6.20 Constuction, destruction and error handling

Functions

- **FMILIB_EXPORT** `fmi1_import_t *` `fmi1_import_parse_xml` (`fmi_import_context_t` `*context`, `const char *``dirPath`)
Create `fmi1_import_t` structure and parse the XML file.
- **FMILIB_EXPORT** `const char *` `fmi1_import_get_last_error` (`fmi1_import_t` `*fmu`)
Retrieve the last error message.
- **FMILIB_EXPORT** `int` `fmi1_import_clear_last_error` (`fmi1_import_t` `*fmu`)
Clear the error message.
- **FMILIB_EXPORT** `void` `fmi1_import_free` (`fmi1_import_t` `*fmu`)
Release the memory allocated.

6.20.1 Function Documentation

6.20.1.1 **FMILIB_EXPORT** `fmi1_import_t *` `fmi1_import_parse_xml` (`fmi_import_context_t` `* context`, `const char *` `dirPath`)

Create `fmi1_import_t` structure and parse the XML file.

Parameters

<i>context</i>	A context data strucutre is used to propagate the callbacks for memory handling and logging.
<i>dirPath</i>	A directory name (full path) of a directory where the FMU was unzipped.

Returns

The new structure if parsing was successfull. 0-pointer is returned on error.

6.20.1.2 **FMILIB_EXPORT** `const char *` `fmi1_import_get_last_error` (`fmi1_import_t` `* fmu`)

Retrieve the last error message.

Error handling:

Many functions in the library return pointers to struct. An error is indicated by returning NULL/0-pointer. If error is returned than `fmi1_import_get_last_error()` functions can be used to retrieve the error message. If logging callbacks were specified then the same information is reported via logger. Memory for the error string is allocated and deallocated in the module. Client code should not store the pointer to the string since it can become invalid.

Parameters

<i>fmu</i>	An FMU object as returned by fmi1_import_parse_xml() .
------------	--

Returns

NULL-terminated string with an error message.

6.20.1.3 FMILIB_EXPORT int fmi1_import_clear_last_error (fmi1_import_t * *fmu*)

Clear the error message.

Parameters

<i>fmu</i>	An FMU object as returned by fmi1_import_parse_xml() .
------------	--

Returns

0 if further processing is possible. If it returns 1 then the error was not recoverable.
The *fmu* object should then be freed and recreated.

6.20.1.4 FMILIB_EXPORT void fmi1_import_free (fmi1_import_t * *fmu*)

Release the memory allocated.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21 General information retrieval

Functions for retrieving general model information. Memory for the strings is allocated and deallocated in the module. All the functions take an FMU object as returned by [fmi1_import_parse_xml\(\)](#) as a parameter. The information is retrieved from the XML file.

Functions

- **FMILIB_EXPORT** const char * [fmi1_import_get_model_name](#) (fmi1_import_t *fmu)
Get model name.
- **FMILIB_EXPORT** const char * [fmi1_import_get_model_identifier](#) (fmi1_import_t *fmu)
Get model identifier.
- **FMILIB_EXPORT** const char * [fmi1_import_get_GUID](#) (fmi1_import_t *fmu)
Get FMU GUID.
- **FMILIB_EXPORT** const char * [fmi1_import_get_description](#) (fmi1_import_t *fmu)
Get FMU description.
- **FMILIB_EXPORT** const char * [fmi1_import_get_author](#) (fmi1_import_t *fmu)
Get FMU author.
- **FMILIB_EXPORT** const char * [fmi1_import_get_model_version](#) (fmi1_import_t *fmu)
Get FMU version.
- **FMILIB_EXPORT** const char * [fmi1_import_get_model_standard_version](#) (fmi1_import_t *fmu)
Get FMI standard version (always 1.0).
- **FMILIB_EXPORT** const char * [fmi1_import_get_generation_tool](#) (fmi1_import_t *fmu)
Get FMU generation tool.
- **FMILIB_EXPORT** const char * [fmi1_import_get_generation_date_and_time](#) (fmi1_import_t *fmu)
Get FMU generation date and time.
- **FMILIB_EXPORT** fmi1_variable_naming_convension_enumeration_t [fmi1_import_get_naming_convention](#) (fmi1_import_t *fmu)
Get variable naming convention used.
- **FMILIB_EXPORT** unsigned int [fmi1_import_get_number_of_continuous_states](#) (fmi1_import_t *fmu)
Get the number of continuous states.
- **FMILIB_EXPORT** unsigned int [fmi1_import_get_number_of_event_indicators](#) (fmi1_import_t *fmu)
Get the number of event indicators.
- **FMILIB_EXPORT** double [fmi1_import_get_default_experiment_start](#) (fmi1_import_t *fmu)

Get the start time for default experiment as specified in the XML file.

- `FMILIB_EXPORT double fmi1_import_get_default_experiment_stop (fmi1_import_t *fmu)`

Get the stop time for default experiment as specified in the XML file.

- `FMILIB_EXPORT double fmi1_import_get_default_experiment_tolerance (fmi1_import_t *fmu)`

Get the tolerance default experiment as specified in the XML file.

- `FMILIB_EXPORT fmi1_fmu_kind_enu_t fmi1_import_get_fmu_kind (fmi1_import_t *fmu)`

Get the type of the FMU (model exchange or co-simulation)

- `FMILIB_EXPORT fmi1_import_capabilities_t * fmi1_import_get_capabilities (fmi1_import_t *fmu)`

Get the structure with capability flags.

- `FMILIB_EXPORT fmi1_import_type_definitions_t * fmi1_import_get_type_definitions (fmi1_import_t *)`

Get the list of all the type definitions in the model.

- `FMILIB_EXPORT fmi1_import_unit_definitions_t * fmi1_import_get_unit_definitions (fmi1_import_t *fmu)`

Get a list of all the unit definitions in the model.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_direct_dependency (fmi1_import_t *fmu, fmi1_import_variable_t *)`

Get the direct dependency information.

- `FMILIB_EXPORT fmi1_import_variable_t * fmi1_import_get_variable_alias_base (fmi1_import_t *fmu, fmi1_import_variable_t *)`

Get the variable with the same value reference that is not an alias.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_variable_aliases (fmi1_import_t *fmu, fmi1_import_variable_t *)`
- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_variable_list (fmi1_import_t *fmu)`

Get the list of all the variables in the model.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_variable_list_alphabetical_order (fmi1_import_t *fmu)`

Get the list of all the variables in the model in alphabetical order.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_create_var_list (fmi1_import_t *fmu, fmi1_import_variable_t *v)`

Create a variable list with a single variable.

6.21.1 Detailed Description

Functions for retrieving general model information. Memory for the strings is allocated and deallocated in the module. All the functions take an FMU object as returned by `fmi1_import_parse_xml()` as a parameter. The information is retrieved from the XML file.

6.21.2 Function Documentation

6.21.2.1 **FMILIB_EXPORT** const char* fmi1_import_get_model_name (fmi1_import_t * *fmu*)

Get model name.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.2 **FMILIB_EXPORT** const char* fmi1_import_get_model_identifier (fmi1_import_t * *fmu*)

Get model identifier.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.3 **FMILIB_EXPORT** const char* fmi1_import_get_GUID (fmi1_import_t * *fmu*)

Get FMU GUID.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.4 **FMILIB_EXPORT** const char* fmi1_import_get_description (fmi1_import_t * *fmu*)

Get FMU description.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.5 **FMILIB_EXPORT** const char* fmi1_import_get_author (fmi1_import_t * *fmu*)

Get FMU author.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.6 FMILIB_EXPORT const char* fmi1_import_get_model_version (
fmi1_import_t * *fmu*)

Get FMU version.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.7 FMILIB_EXPORT const char* fmi1_import_get_model_standard_version (
fmi1_import_t * *fmu*)

Get FMI standard version (always 1.0).

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.8 FMILIB_EXPORT const char* fmi1_import_get_generation_tool (
fmi1_import_t * *fmu*)

Get FMU generation tool.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.9 FMILIB_EXPORT const char* fmi1_import_get_
generation_date_and_time (fmi1_import_t * *fmu*
)

Get FMU generation date and time.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.10 **FMILIB_EXPORT** fmi1_variable_naming_convension_enu_t
 fmi1_import_get_naming_convention (fmi1_import_t * fmu)

Get variable naming convention used.

Parameters

<i>fmu</i>	An fmu object as returned by fmi1_import_parse_xml() .
------------	--

6.21.2.11 **FMILIB_EXPORT** unsigned int fmi1_import_get -
 number_of_continuous_states (fmi1_import_t * fmu
)

Get the number of continuous states.

6.21.2.12 **FMILIB_EXPORT** unsigned int fmi1_import_get -
 number_of_event_indicators (fmi1_import_t * fmu
)

Get the number of event indicators.

6.21.2.13 **FMILIB_EXPORT** double fmi1_import_get_default_experiment_start (fmi1_import_t * fmu)

Get the start time for default experiment as specified in the XML file.

6.21.2.14 **FMILIB_EXPORT** double fmi1_import_get_default_experiment_stop (fmi1_import_t * fmu)

Get the stop time for default experiment as specified in the XML file.

6.21.2.15 **FMILIB_EXPORT** double fmi1_import_get_default_experiment_tolerance (fmi1_import_t * fmu)

Get the tolerance default experiment as specified in the XML file.

6.21.2.16 **FMILIB_EXPORT** fmi1_fmu_kind_enu_t fmi1_import_get_fmu_kind (fmi1_import_t * fmu)

Get the type of the FMU (model exchange or co-simulation)

6.21.2.17 FMILIB_EXPORT fmi1_import_capabilities_t*
fmi1_import_get_capabilities (fmi1_import_t * *fmu*)

Get the structure with capability flags.

Returns

A pointer to the `fmi1_import_capabilities_t` allocated within the library. Note that for model exchange FMUs the values of all the flags are always default.

6.21.2.18 FMILIB_EXPORT fmi1_import_type_definitions_t*
fmi1_import_get_type_definitions (fmi1_import_t *)

Get the list of all the type definitions in the model.

6.21.2.19 FMILIB_EXPORT fmi1_import_unit_definitions_t*
fmi1_import_get_unit_definitions (fmi1_import_t * *fmu*)

Get a list of all the unit definitions in the model.

6.21.2.20 FMILIB_EXPORT fmi1_import_variable_list_t* fmi1_import_get -
direct_dependency (fmi1_import_t * *fmu*, fmi1_import_variable_t *
)

Get the direct dependency information.

Returns

A variable list is returned for variables with causality Output. Null pointer for others.

6.21.2.21 FMILIB_EXPORT fmi1_import_variable_t* fmi1_import_get -
variable_alias_base (fmi1_import_t * *fmu*, fmi1_import_variable_t *
)

Get the variable with the same value reference that is not an alias.

6.21.2.22 FMILIB_EXPORT fmi1_import_variable_list_t* fmi1_import_get -
variable_aliases (fmi1_import_t * *fmu*, fmi1_import_variable_t *
)

Get the list of all the variables aliased to the given one (including the base one).

Note that the list is ordered: base variable, aliases, negated aliases.

6.21.2.23 **FMILIB_EXPORT** fmi1_import_variable_list_t*
fmi1_import_get_variable_list (fmi1_import_t * *fmu*)

Get the list of all the variables in the model.

Parameters

<i>fmu</i>	An FMU object as returned by fmi1_import_parse_xml() .
------------	--

Returns

a variable list with all the variables in the model.

Note that variable lists are allocated dynamically and must be freed when not needed any longer.

6.21.2.24 **FMILIB_EXPORT** fmi1_import_variable_list_t* fmi1_import-
_get_variable_list_alphabetical_order (fmi1_import_t * *fmu*
)

Get the list of all the variables in the model in alphabetical order.

Parameters

<i>fmu</i>	An FMU object as returned by fmi1_import_parse_xml() .
------------	--

Returns

a variable list with all the variables in the model sorted in alphabetical order.

Note that variable lists are allocated dynamically and must be freed when not needed any longer.

6.21.2.25 **FMILIB_EXPORT** fmi1_import_variable_list_t* fmi1_import-
_create_var_list (fmi1_import_t * *fmu*, fmi1_import_variable_t * *v*
)

Create a variable list with a single variable.

Parameters

<i>fmu</i>	An FMU object that this variable list will reference.
<i>v</i>	A variable.

6.22 Interface to the standard FMI 1.0 "C" API

Convenient functions for calling the FMI functions. This interface wraps the "C" API.

Modules

- [FMI 1.0 Constructor and Destructor](#)
Functions for instantiating and freeing the container of the struct that is responsible for the FMI functions.
- [FMI 1.0 \(ME\) Model Exchange functions](#)
List of Model Exchange wrapper functions. Common functions are not listed.
- [FMI 1.0 \(CS\) Co-Simulation functions](#)
List of Co-Simulation wrapper functions. Common functions are not listed.
- [FMI 1.0 \(ME & CS\) Common functions](#)
List of wrapper functions that are in common for both Model Exchange and Co-Simulation.

6.22.1 Detailed Description

Convenient functions for calling the FMI functions. This interface wraps the "C" API.

6.23 Functions to retrieve capability flags.

Functions

- `FMILIB_EXPORT int fmi1_import_get_canHandleVariableCommunicationStepSize (fmi1_import_capabilities_t *)`
Retrieve canHandleVariableCommunicationStepSize flag.
- `FMILIB_EXPORT int fmi1_import_get_canHandleEvents (fmi1_import_capabilities_t *)`
Retrieve canHandleEvents flag.
- `FMILIB_EXPORT int fmi1_import_get_canRejectSteps (fmi1_import_capabilities_t *)`
Retrieve canRejectSteps flag.
- `FMILIB_EXPORT int fmi1_import_get_canInterpolateInputs (fmi1_import_capabilities_t *)`
Retrieve canInterpolateInputs flag.
- `FMILIB_EXPORT unsigned int fmi1_import_get_maxOutputDerivativeOrder (fmi1_import_capabilities_t *)`
Retrieve maxOutputDerivativeOrder.
- `FMILIB_EXPORT int fmi1_import_get_canRunAsynchronously (fmi1_import_capabilities_t *)`
Retrieve canRunAsynchronously flag.
- `FMILIB_EXPORT int fmi1_import_get_canSignalEvents (fmi1_import_capabilities_t *)`
Retrieve canSignalEvents flag.
- `FMILIB_EXPORT int fmi1_import_get_canBeInstantiatedOnlyOncePerProcess (fmi1_import_capabilities_t *)`
Retrieve canBeInstantiatedOnlyOncePerProcess flag.
- `FMILIB_EXPORT int fmi1_import_get_canNotUseMemoryManagementFunctions (fmi1_import_capabilities_t *)`
Retrieve canNotUseMemoryManagementFunctions flag.

FMU capabilities flags

- `typedef struct fmi1_xml_capabilities_t fmi1_import_capabilities_t`
A container for all the capability flags.

6.23.1 Detailed Description

The functions accept a pointer to `fmi1_import_capabilities_t` returned by `fmi1_import_get_capabilities()`. They return the flags as specified by the FMI 1.0 standard. Default values are returned for model-exchange FMUs.

6.23.2 Typedef Documentation

6.23.2.1 typedef struct fmi1_xml_capabilities_t fmi1_import_capabilities_t

A container for all the capability flags.

Definition at line 42 of file fmi1_import_capabilities.h.

6.23.3 Function Documentation

6.23.3.1 FMILIB_EXPORT int fmi1_import_get_canHandleVariableCommunicationStepSize (fmi1_import_capabilities_t *

Retrieve canHandleVariableCommunicationStepSize flag.

6.23.3.2 FMILIB_EXPORT int fmi1_import_get_canHandleEvents (fmi1_import_capabilities_t *

Retrieve canHandleEvents flag.

6.23.3.3 FMILIB_EXPORT int fmi1_import_get_canRejectSteps (fmi1_import_capabilities_t *

Retrieve canRejectSteps flag.

6.23.3.4 FMILIB_EXPORT int fmi1_import_get_canInterpolateInputs (fmi1_import_capabilities_t *

Retrieve canInterpolateInputs flag.

6.23.3.5 FMILIB_EXPORT unsigned int fmi1_import_get_maxOutputDerivativeOrder (fmi1_import_capabilities_t *

Retrieve maxOutputDerivativeOrder.

6.23.3.6 FMILIB_EXPORT int fmi1_import_get_canRunAsynchronously (fmi1_import_capabilities_t *

Retrieve canRunAsynchronously flag.

6.23.3.7 FMILIB_EXPORT int fmi1_import_get_canSignalEvents (fmi1_import_capabilities_t *

Retrieve canSignalEvents flag.

**6.23.3.8 FMILIB_EXPORT int fmi1_import_get_canBeInstantiated-
OnlyOncePerProcess (fmi1_import_capabilities_t ***
)

Retrieve canBeInstantiatedOnlyOncePerProcess flag.

**6.23.3.9 FMILIB_EXPORT int fmi1_import_get_canNotUseMemory-
ManagementFunctions (fmi1_import_capabilities_t ***
)

Retrieve canNotUseMemoryManagementFunctions flag.

6.24 FMI 1.0 Constructor and Destructor

Functions for instantiating and freeing the container of the struct that is responsible for the FMI functions.

Functions

- **FMILIB_EXPORT** `jm_status_enu_t fmi1_import_create_dllfmu (fmi1_import_t *fmu, fmi1_callback_functions_t callBackFunctions, int registerGlobally)`
Create a C-API struct. The C-API struct is a placeholder for the FMI DLL functions.
- **FMILIB_EXPORT** `void fmi1_import_destroy_dllfmu (fmi1_import_t *fmu)`
Free a C-API struct. All memory allocated since the struct was created is freed.
- **FMILIB_EXPORT** `void fmi1_import_set_debug_mode (fmi1_import_t *fmu, int mode)`
Set CAPI debug mode flag. Setting to non-zero prevents DLL unloading in fmi1_import_destroy_dllfmu while all the memory is deallocated. This is to support valgrind debugging.

6.24.1 Detailed Description

Functions for instantiating and freeing the container of the struct that is responsible for the FMI functions. Before any of the FMI functions may be called, the construction function must instantiate a `fmi_import_t` module. After the `fmi_import_t` module has been successfully instantiated, all the FMI functions can be called. To unload the FMI functions, the destroy functions shall be called.

6.24.2 Function Documentation

6.24.2.1 FMILIB_EXPORT `jm_status_enu_t fmi1_import_create_dllfmu ( fmi1_import_t * fmu, fmi1_callback_functions_t callBackFunctions, int registerGlobally )`

Create a C-API struct. The C-API struct is a placeholder for the FMI DLL functions.

This function may only be called once if it returned successfully. `fmi1_import_destroy_dllfmu` must be called before this function can be called again.

Parameters

<i>fmu</i>	A model description object returned by <code>fmi1_import_parse_xml()</code> .
<i>callBack-Functions</i>	Callback functions used by the FMI functions internally.
<i>register-Globally</i>	Register the FMU globally to enable use of <code>fmi1_log_forwarding()</code> . If this parameter is non-zero the code becomes non-thread safe.

Returns

Error status. If the function returns with an error, it is not allowed to call any of the other C-API functions.

6.24.2.2 FMILIB_EXPORT void fmi1_import_destroy_dllfmu (fmi1_import_t * *fmu*)

Free a C-API struct. All memory allocated since the struct was created is freed.

Parameters

<i>fmu</i>	A model description object returned from fmi1_import_parse_xml() .
------------	--

6.24.2.3 FMILIB_EXPORT void fmi1_import_set_debug_mode (fmi1_import_t * *fmu*, int *mode*)

Set CAPI debug mode flag. Setting to non-zero prevents DLL unloading in `fmi1_import_destroy_dllfmu` while all the memory is deallocated. This is to support valgrind debugging.

Parameters

<i>fmu</i>	C-API struct that has successfully loaded the FMI function.
<i>mode</i>	The debug mode to set.

6.25 FMI 1.0 (ME) Model Exchange functions

List of Model Exchange wrapper functions. Common functions are not listed.

Functions

- `FMILIB_EXPORT const char * fmi1_import_get_model_types_platform (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiGetModelTypesPlatform(...)
- `FMILIB_EXPORT jm_status_enu_t fmi1_import_instantiate_model (fmi1_import_t *fmu, fmi1_string_t instanceName)`
Wrapper for the FMI function fmiInstantiateModel(...)
- `FMILIB_EXPORT void fmi1_import_free_model_instance (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiFreeModelInstance(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_set_time (fmi1_import_t *fmu, fmi1_real_t time)`
Wrapper for the FMI function fmiSetTime(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_set_continuous_states (fmi1_import_t *fmu, const fmi1_real_t x[], size_t nx)`
Wrapper for the FMI function fmiSetContinuousStates(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_completed_integrator_step (fmi1_import_t *fmu, fmi1_boolean_t *callEventUpdate)`
Wrapper for the FMI function fmiCompletedIntegratorStep(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_initialize (fmi1_import_t *fmu, fmi1_boolean_t toleranceControlled, fmi1_real_t relativeTolerance, fmi1_event_info_t *eventInfo)`
Wrapper for the FMI function fmiInitialize(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_derivatives (fmi1_import_t *fmu, fmi1_real_t derivatives[], size_t nx)`
Wrapper for the FMI function fmiGetDerivatives(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_event_indicators (fmi1_import_t *fmu, fmi1_real_t eventIndicators[], size_t ni)`
Wrapper for the FMI function fmiGetEventIndicators(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_eventUpdate (fmi1_import_t *fmu, fmi1_boolean_t intermediateResults, fmi1_event_info_t *eventInfo)`
Wrapper for the FMI function fmiEventUpdate(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_continuous_states (fmi1_import_t *fmu, fmi1_real_t states[], size_t nx)`
Wrapper for the FMI function fmiGetContinuousStates(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_nominal_continuous_states (fmi1_import_t *fmu, fmi1_real_t x_nominal[], size_t nx)`
Wrapper for the FMI function fmiGetNominalContinuousStates(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_state_value_references (fmi1_import_t *fmu, fmi1_value_reference_t vrx[], size_t nx)`

Wrapper for the FMI function *fmiGetStateValueReferences(...)*

- **FMLIB_EXPORT** *fmi1_status_t fmi1_import_terminate* (*fmi1_import_t *fmu*)

Wrapper for the FMI function *fmiTerminate(...)*

6.25.1 Detailed Description

List of Model Exchange wrapper functions. Common functions are not listed.

6.25.2 Function Documentation

6.25.2.1 **FMLIB_EXPORT** *const char* fmi1_import_get_model_types_platform* (*fmi1_import_t *fmu*)

Wrapper for the FMI function *fmiGetModelTypesPlatform(...)*

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
------------	---

Returns

The platform the FMU was compiled for.

6.25.2.2 **FMLIB_EXPORT** *jm_status_enu_t fmi1_import_instantiate_model* (*fmi1_import_t *fmu*, *fmi1_string_t instanceName*)

Wrapper for the FMI function *fmiInstantiateModel(...)*

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>instance-Name</i>	The name of the instance.

Returns

Error status. Returns *jm_status_error* if *fmiInstantiateModel* returned NULL, otherwise *jm_status_success*.

6.25.2.3 **FMLIB_EXPORT** *void fmi1_import_free_model_instance* (*fmi1_import_t *fmu*)

Wrapper for the FMI function *fmiFreeModelInstance(...)*

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
------------	---

6.25.2.4 FMILIB_EXPORT fmi1_status_t fmi1_import_set_time (fmi1_import_t * *fmu*, fmi1_real_t *time*)

Wrapper for the FMI function fmiSetTime(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>time</i>	Set the current time.

Returns

FMI status.

6.25.2.5 FMILIB_EXPORT fmi1_status_t fmi1_import_set_continuous_states (fmi1_import_t * *fmu*, const fmi1_real_t *x*[], size_t *nx*)

Wrapper for the FMI function fmiSetContinuousStates(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>x</i>	Array of state values.
<i>nx</i>	Number of states.

Returns

FMI status.

6.25.2.6 FMILIB_EXPORT fmi1_status_t fmi1_import_completed_integrator_step (fmi1_import_t * *fmu*, fmi1_boolean_t * *callEventUpdate*)

Wrapper for the FMI function fmiCompletedIntegratorStep(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>callEventUpdate</i>	(Output) Call fmiEventUpdate indicator.

Returns

FMI status.

6.25.2.7 FMILIB_EXPORT fmi1_status_t fmi1_import_initialize (fmi1_import_t * *fmu*, fmi1_boolean_t *toleranceControlled*, fmi1_real_t *relativeTolerance*, fmi1_event_info_t * *eventInfo*)

Wrapper for the FMI function fmiInitialize(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>tolerance-Controlled</i>	Enable or disable the use of relativeTolerance in the FMU.
<i>relative-Tolerance</i>	A relative tolerance used in the FMU.
<i>eventInfo</i>	(Output) fmiEventInfo struct.

Returns

FMI status.

6.25.2.8 FMILIB_EXPORT fmi1_status_t fmi1_import_get_derivatives (fmi1_import_t * *fmu*, fmi1_real_t *derivatives*[], size_t *nx*)

Wrapper for the FMI function fmiGetDerivatives(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>derivatives</i>	(Output) Array of the derivatives.
<i>nx</i>	Number of derivatives.

Returns

FMI status.

6.25.2.9 FMILIB_EXPORT fmi1_status_t fmi1_import_get_event_indicators (fmi1_import_t * *fmu*, fmi1_real_t *eventIndicators*[], size_t *ni*)

Wrapper for the FMI function fmiGetEventIndicators(...)

Parameters

<i>fmU</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>event-Indicators</i>	(Output) The event indicators.
<i>ni</i>	Number of event indicators.

Returns

FMI status.

6.25.2.10 FMILIB_EXPORT fmi1_status_t fmi1_import_eventUpdate (
fmi1_import_t * fmU, fmi1_boolean_t intermediateResults, fmi1_event_info_t *
eventInfo)

Wrapper for the FMI function fmiEventUpdate(...)

Parameters

<i>fmU</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>intermediate-Results</i>	Indicate whether or not the fmiEventUpdate shall return after every internal event iteration.
<i>eventInfo</i>	(Output) An fmiEventInfo struct.

Returns

FMI status.

6.25.2.11 FMILIB_EXPORT fmi1_status_t fmi1_import_get_continuous_states (
fmi1_import_t * fmU, fmi1_real_t states[], size_t nx)

Wrapper for the FMI function fmiGetContinuousStates(...)

Parameters

<i>fmU</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>states</i>	(Output) Array of state values.
<i>nx</i>	Number of states.

Returns

FMI status.

6.25.2.12 **FMILIB_EXPORT** fmi1_status_t fmi1_import_get_nominal_continuous_states (fmi1_import_t * *fm**u*, fmi1_real_t *x_nominal*[], size_t *nx*)

Wrapper for the FMI function fmiGetNominalContinuousStates(...)

Parameters

<i>fm</i> <i>u</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfm <i>u</i> ()).
<i>x_nominal</i>	(Output) The nominal values.
<i>nx</i>	Number of nominal values.

Returns

FMI status.

6.25.2.13 **FMILIB_EXPORT** fmi1_status_t fmi1_import_get_state_value_references (fmi1_import_t * *fm**u*, fmi1_value_reference_t *vr**x*[], size_t *nx*)

Wrapper for the FMI function fmiGetStateValueReferences(...)

Parameters

<i>fm</i> <i>u</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfm <i>u</i> ()).
<i>vr</i> <i>x</i>	(Output) The value-references of the states.
<i>nx</i>	Number of value-references.

Returns

FMI status.

6.25.2.14 **FMILIB_EXPORT** fmi1_status_t fmi1_import_terminate (fmi1_import_t * *fm**u*)

Wrapper for the FMI function fmiTerminate(...)

Parameters

<i>fm</i> <i>u</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfm <i>u</i> ()).
--------------------	--

Returns

FMI status.

6.26 FMI 1.0 (CS) Co-Simulation functions

List of Co-Simulation wrapper functions. Common functions are not listed.

Functions

- **FMLIB_EXPORT** `const char * fmi1_import_get_types_platform (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiGetTypesPlatform(...)
- **FMLIB_EXPORT** `jm_status_enu_t fmi1_import_instantiate_slave (fmi1_import_t *fmu, fmi1_string_t instanceName, fmi1_string_t fmuLocation, fmi1_string_t mimeType, fmi1_real_t timeout, fmi1_boolean_t visible, fmi1_boolean_t interactive)`
Wrapper for the FMI function fmiInstantiateSlave(...)
- **FMLIB_EXPORT** `fmi1_status_t fmi1_import_initialize_slave (fmi1_import_t *fmu, fmi1_real_t tStart, fmi1_boolean_t StopTimeDefined, fmi1_real_t tStop)`
Wrapper for the FMI function fmiInitializeSlave(...)
- **FMLIB_EXPORT** `fmi1_status_t fmi1_import_terminate_slave (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiTerminateSlave(...)
- **FMLIB_EXPORT** `fmi1_status_t fmi1_import_reset_slave (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiResetSlave(...)
- **FMLIB_EXPORT** `void fmi1_import_free_slave_instance (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiFreeSlaveInstance(...)
- **FMLIB_EXPORT** `fmi1_status_t fmi1_import_set_real_input_derivatives (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t order[], const fmi1_real_t value[])`
Wrapper for the FMI function fmiSetRealInputDerivatives(...)
- **FMLIB_EXPORT** `fmi1_status_t fmi1_import_get_real_output_derivatives (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t order[], fmi1_real_t value[])`
Wrapper for the FMI function fmiGetOutputDerivatives(...)
- **FMLIB_EXPORT** `fmi1_status_t fmi1_import_cancel_step (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiCancelStep(...)
- **FMLIB_EXPORT** `fmi1_status_t fmi1_import_do_step (fmi1_import_t *fmu, fmi1_real_t currentCommunicationPoint, fmi1_real_t communicationStepSize, fmi1_boolean_t newStep)`
Wrapper for the FMI function fmiDoStep(...)
- **FMLIB_EXPORT** `fmi1_status_t fmi1_import_get_status (fmi1_import_t *fmu, const fmi1_status_kind_t s, fmi1_status_t *value)`
Wrapper for the FMI function fmiGetStatus(...)
- **FMLIB_EXPORT** `fmi1_status_t fmi1_import_get_real_status (fmi1_import_t *fmu, const fmi1_status_kind_t s, fmi1_real_t *value)`
Wrapper for the FMI function fmiGetRealStatus(...)

- [FMILIB_EXPORT fmi1_status_t fmi1_import_get_integer_status](#) ([fmi1_import_t *fmu](#), const [fmi1_status_kind_t](#) s, [fmi1_integer_t](#) *value)
Wrapper for the FMI function [fmiGetIntegerStatus\(...\)](#)
- [FMILIB_EXPORT fmi1_status_t fmi1_import_get_boolean_status](#) ([fmi1_import_t *fmu](#), const [fmi1_status_kind_t](#) s, [fmi1_boolean_t](#) *value)
Wrapper for the FMI function [fmiGetBooleanStatus\(...\)](#)
- [FMILIB_EXPORT fmi1_status_t fmi1_import_get_string_status](#) ([fmi1_import_t *fmu](#), const [fmi1_status_kind_t](#) s, [fmi1_string_t](#) *value)
Wrapper for the FMI function [fmiGetStringStatus\(...\)](#)

6.26.1 Detailed Description

List of Co-Simulation wrapper functions. Common functions are not listed.

6.26.2 Function Documentation

6.26.2.1 [FMILIB_EXPORT const char* fmi1_import_get_types_platform](#) ([fmi1_import_t * fmu](#))

Wrapper for the FMI function [fmiGetTypesPlatform\(...\)](#)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
------------	---

Returns

The platform the FMU was compiled for.

6.26.2.2 [FMILIB_EXPORT jm_status_enu_t fmi1_import_instantiate_slave](#) ([fmi1_import_t * fmu](#), [fmi1_string_t](#) instanceName, [fmi1_string_t](#) fmuLocation, [fmi1_string_t](#) mimeType, [fmi1_real_t](#) timeout, [fmi1_boolean_t](#) visible, [fmi1_boolean_t](#) interactive)

Wrapper for the FMI function [fmiInstantiateSlave\(...\)](#)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>instance-Name</i>	The name of the instance.
<i>fmuLocation</i>	Access path to the FMU archive. If null FMU will get the path to the directory where it was unpacked.
<i>mimeType</i>	MIME type. If NULL the FMU will get "application/x-fmu-sharedlibrary".

<i>timeout</i>	Communication timeout value in milli-seconds.
<i>visible</i>	Indicates whether or not the simulator application window should be visible.
<i>interactive</i>	Indicates whether the simulator application must be manually started by the user.

Returns

Error status. Returns `jm_status_error` if `fmiInstantiateSlave` returned `NULL`, otherwise `jm_status_success`.

6.26.2.3 FMILIB_EXPORT fmi1_status_t fmi1_import_initialize_slave (
fmi1_import_t * fmu, fmi1_real_t tStart, fmi1_boolean_t StopTimeDefined,
fmi1_real_t tStop)

Wrapper for the FMI function `fmiInitializeSlave(...)`

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>tStart</i>	Start time of the simulation
<i>StopTime-Defined</i>	Indicates whether or not the stop time is used.
<i>tStop</i>	The stop time of the simulation.

Returns

FMI status.

6.26.2.4 FMILIB_EXPORT fmi1_status_t fmi1_import_terminate_slave (
fmi1_import_t * fmu)

Wrapper for the FMI function `fmiTerminateSlave(...)`

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
------------	---

Returns

FMI status.

6.26.2.5 FMILIB_EXPORT fmi1_status_t fmi1_import_reset_slave (fmi1_import_t * *fmu*)

Wrapper for the FMI function fmiResetSlave(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
------------	---

Returns

FMI status.

6.26.2.6 FMILIB_EXPORT void fmi1_import_free_slave_instance (fmi1_import_t * *fmu*)

Wrapper for the FMI function fmiFreeSlaveInstance(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
------------	---

6.26.2.7 FMILIB_EXPORT fmi1_status_t fmi1_import_set_real_input_derivatives (fmi1_import_t * *fmu*, const fmi1_value_reference_t *vr*[], size_t *nvr*, const fmi1_integer_t *order*[], const fmi1_real_t *value*[])

Wrapper for the FMI function fmiSetRealInputDerivatives(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>order</i>	Array of derivative orders.
<i>value</i>	Array of variable values.

Returns

FMI status.

6.26.2.8 FMILIB_EXPORT fmi1_status_t fmi1_import_get_real_output_derivatives (fmi1_import_t * *fmu*, const fmi1_value_reference_t *vr*[], size_t *nvr*, const fmi1_integer_t *order*[], fmi1_real_t *value*[])

Wrapper for the FMI function fmiGetOutputDerivatives(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>order</i>	Array of derivative orders.
<i>value</i>	(Output) Array of variable values.

Returns

FMI status.

6.26.2.9 FMILIB_EXPORT fmi1_status_t fmi1_import_cancel_step (fmi1_import_t * *fmu*)

Wrapper for the FMI function fmiCancelStep(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
------------	---

Returns

FMI status.

6.26.2.10 FMILIB_EXPORT fmi1_status_t fmi1_import_do_step (fmi1_import_t * *fmu*, fmi1_real_t *currentCommunicationPoint*, fmi1_real_t *communicationStepSize*, fmi1_boolean_t *newStep*)

Wrapper for the FMI function fmiDoStep(...)

Parameters

<i>fm</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>current-Communication-Point</i>	Current communication point of the master.
<i>communication-StepSize</i>	Communication step size.
<i>newStep</i>	Indicates whether or not the last communication step was accepted by the master.

Returns

FMI status.

6.26.2.11 FMILIB_EXPORT fmi1_status_t fmi1_import_get_status (fmi1_import_t * *fm*, const fmi1_status_kind_t *s*, fmi1_status_t * *value*)

Wrapper for the FMI function `fmiGetStatus(...)`

Parameters

<i>fm</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI status value.

Returns

FMI status.

6.26.2.12 FMILIB_EXPORT fmi1_status_t fmi1_import_get_real_status (fmi1_import_t * *fm*, const fmi1_status_kind_t *s*, fmi1_real_t * *value*)

Wrapper for the FMI function `fmiGetRealStatus(...)`

Parameters

<i>fm</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI real value.

Returns

FMI status.

6.26.2.13 FMILIB_EXPORT fmi1_status_t fmi1_import_get_integer_status (
fmi1_import_t * *fm*, const fmi1_status_kind_t *s*, fmi1_integer_t * *value*)

Wrapper for the FMI function fmiGetIntegerStatus(...)

Parameters

<i>fm</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfm() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI integer value.

Returns

FMI status.

6.26.2.14 FMILIB_EXPORT fmi1_status_t fmi1_import_get_boolean_status (
fmi1_import_t * *fm*, const fmi1_status_kind_t *s*, fmi1_boolean_t * *value*)

Wrapper for the FMI function fmiGetBooleanStatus(...)

Parameters

<i>fm</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfm() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI boolean value.

Returns

FMI status.

6.26.2.15 FMILIB_EXPORT fmi1_status_t fmi1_import_get_string_status (
fmi1_import_t * *fm*, const fmi1_status_kind_t *s*, fmi1_string_t * *value*)

Wrapper for the FMI function fmiGetStringStatus(...)

Parameters

<i>fm</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfm() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI string value.

Returns

FMI status.

6.27 FMI 1.0 (ME & CS) Common functions

List of wrapper functions that are in common for both Model Exchange and Co-Simulation.

Functions

- **FMILIB_EXPORT** `const char * fmi1_import_get_version (fmi1_import_t *fmu)`
Wrapper for the FMI function `fmiGetVersion()`
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_debug_logging (fmi1_import_t *fmu, fmi1_boolean_t loggingOn)`
Wrapper for the FMI function `fmiSetDebugLogging(...)`
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_real (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_real_t value[])`
Wrapper for the FMI function `fmiSetReal(...)`
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_integer (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t value[])`
Wrapper for the FMI function `fmiSetInteger(...)`
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_boolean (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_boolean_t value[])`
Wrapper for the FMI function `fmiSetBoolean(...)`
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_string (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_string_t value[])`
Wrapper for the FMI function `fmiSetString(...)`
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_real (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, fmi1_real_t value[])`
Wrapper for the FMI function `fmiGetReal(...)`
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_integer (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, fmi1_integer_t value[])`
Wrapper for the FMI function `fmiGetInteger(...)`
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_boolean (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, fmi1_boolean_t value[])`
Wrapper for the FMI function `fmiGetBoolean(...)`
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_string (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, fmi1_string_t value[])`
Wrapper for the FMI function `fmiGetString(...)`

6.27.1 Detailed Description

List of wrapper functions that are in common for both Model Exchange and Co-Simulation.

6.27.2 Function Documentation

6.27.2.1 FMILIB_EXPORT const char* fmi1_import_get_version (fmi1_import_t * *fmu*)

Wrapper for the FMI function [fmiGetVersion\(\)](#)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
------------	---

Returns

FMI version.

6.27.2.2 FMILIB_EXPORT fmi1_status_t fmi1_import_set_debug_logging (fmi1_import_t * *fmu*, fmi1_boolean_t *loggingOn*)

Wrapper for the FMI function [fmiSetDebugLogging\(...\)](#)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>loggingOn</i>	Enable or disable the debug logger.

Returns

FMI status.

6.27.2.3 FMILIB_EXPORT fmi1_status_t fmi1_import_set_real (fmi1_import_t * *fmu*, const fmi1_value_reference_t *vr[]*, size_t *nvr*, const fmi1_real_t *value[]*)

Wrapper for the FMI function [fmiSetReal\(...\)](#)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	Array of variable values.

Returns

FMI status.

6.27.2.4 FMILIB_EXPORT fmi1_status_t fmi1_import_set_integer (fmi1_import_t * *fmu*, const fmi1_value_reference_t *vr*[], size_t *nvr*, const fmi1_integer_t *value*[])

Wrapper for the FMI function fmiSetInteger(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	Array of variable values.

Returns

FMI status.

6.27.2.5 FMILIB_EXPORT fmi1_status_t fmi1_import_set_boolean (fmi1_import_t * *fmu*, const fmi1_value_reference_t *vr*[], size_t *nvr*, const fmi1_boolean_t *value*[])

Wrapper for the FMI function fmiSetBoolean(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	Array of variable values.

Returns

FMI status.

6.27.2.6 FMILIB_EXPORT fmi1_status_t fmi1_import_set_string (fmi1_import_t * *fmu*, const fmi1_value_reference_t *vr*[], size_t *nvr*, const fmi1_string_t *value*[])

Wrapper for the FMI function fmiSetString(...)

Parameters

<i>fm</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	Array of variable values.

Returns

FMI status.

**6.27.2.7 FMILIB_EXPORT fmi1_status_t fmi1_import_get_real (fmi1_import_t *
fm, const fmi1_value_reference_t vr[], size_t nvr, fmi1_real_t value[])**

Wrapper for the FMI function fmiGetReal(...)

Parameters

<i>fm</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	(Output)Array of variable values.

Returns

FMI status.

**6.27.2.8 FMILIB_EXPORT fmi1_status_t fmi1_import_get_integer (fmi1_import_t
* fm, const fmi1_value_reference_t vr[], size_t nvr, fmi1_integer_t value[])**

Wrapper for the FMI function fmiGetInteger(...)

Parameters

<i>fm</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	(Output)Array of variable values.

Returns

FMI status.

6.27.2.9 FMILIB_EXPORT fmi1_status_t fmi1_import_get_boolean (
fmi1_import_t * *fmu*, const fmi1_value_reference_t *vr[]*, size_t *nvr*, fmi1_boolean_t
***value[]*)**

Wrapper for the FMI function fmiGetBoolean(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	(Output)Array of variable values.

Returns

FMI status.

6.27.2.10 FMILIB_EXPORT fmi1_status_t fmi1_import_get_string (fmi1_import_t
*** *fmu*, const fmi1_value_reference_t *vr[]*, size_t *nvr*, fmi1_string_t *value[]*)**

Wrapper for the FMI function fmiGetString(...)

Parameters

<i>fmu</i>	A model description object returned by fmi1_import_parse_xml() that has loaded the FMI functions, see fmi1_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	(Output)Array of variable values.

Returns

FMI status.

6.28 Convenience functions.

The functions in this module are provided for convenience. The functionality is already available via other lower level functions.

Data Structures

- struct [fmi1_import_model_counts_t](#)
Collection of counters providing model information.

Functions

- [FMILIB_EXPORT](#) void [fmi1_import_collect_model_counts](#) ([fmi1_import_t](#) *fmu, [fmi1_import_model_counts_t](#) *counts)
Collect model information by counting the number of variables with specific properties and fillin in [fmi1_import_model_counts_t](#) struct.
- [FMILIB_EXPORT](#) void [fmi1_import_expand_variable_references](#) ([fmi1_import_t](#) *fmu, const [char](#) *msgIn, [char](#) *msgOut, [size_t](#) maxMsgSize)
Print msgIn into msgOut by expanding variable references of the form #<Type><V-R># into variable names and replacing '##' with a single #.
- [FMILIB_EXPORT](#) void [fmi1_log_forwarding](#) ([fmi1_component_t](#) c, [fmi1_string_t](#) instanceName, [fmi1_status_t](#) status, [fmi1_string_t](#) category, [fmi1_string_t](#) message,...)
An implementation of FMI 1.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.
- [FMILIB_EXPORT](#) void [fmi1_log_forwarding_v](#) ([fmi1_component_t](#) c, [fmi1_string_t](#) instanceName, [fmi1_status_t](#) status, [fmi1_string_t](#) category, [fmi1_string_t](#) message, va_list args)
An implementation of FMI 1.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.
- [FMILIB_EXPORT](#) void [fmi1_default_callback_logger](#) ([fmi1_component_t](#) c, [fmi1_string_t](#) instanceName, [fmi1_status_t](#) status, [fmi1_string_t](#) category, [fmi1_string_t](#) message,...)
Default FMI 1.0 logger may be used when instantiating FMUs.
- [FMILIB_EXPORT](#) void [fmi1_import_init_logger](#) ([jm_callbacks](#) *cb, [fmi1_callback_functions_t](#) *fmiCallbacks)
Given [fmi1_callback_functions_t](#) logger ([fmi1_logger](#)), the [jm_callbacks](#) logger may be setup to redirect the messages to the [fmi1_logger](#).

6.28.1 Detailed Description

The functions in this module are provided for convenience. The functionality is already available via other lower level functions.

6.28.2 Function Documentation

6.28.2.1 FMILIB_EXPORT void fmi1_import_collect_model_counts (
fmi1_import_t * *fmu*, fmi1_import_model_counts_t * *counts*)

Collect model information by counting the number of variables with specific properties and fillin in [fmi1_import_model_counts_t](#) struct.

Parameters

<i>fmu</i>	- An fmu object as returned by fmi1_import_parse_xml() .
<i>counts</i>	- a pointer to a preallocated struct.

6.28.2.2 FMILIB_EXPORT void fmi1_import_expand_variable_references (
fmi1_import_t * *fmu*, const char * *msgIn*, char * *msgOut*, size_t *maxMsgSize*)

Print *msgIn* into *msgOut* by expanding variable references of the form #<Type><V-R># into variable names and replacing '##' with a single #.

Parameters

<i>fmu</i>	- An fmu object as returned by fmi1_import_parse_xml() .
<i>msgIn</i>	- Log message as produced by an FMU.
<i>msgOut</i>	- Output message buffer.
<i>maxMsgSize</i>	- maximum message size

6.28.2.3 FMILIB_EXPORT void fmi1_log_forwarding (fmi1_component_t *c*, fmi1_string_t *instanceName*, fmi1_status_t *status*, fmi1_string_t *category*, fmi1_string_t *message*, ...)

An implementation of FMI 1.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.

The function is using a global array of active FMUs to find out which FMU is sending the log message. It then forwards the message to the logger connected to the particular [fmi1_import_t](#) struct. The function is called by the FMU. The FMU must be loaded with non-zero registerGlobally parameter of [fmi1_import_create_dllfmu\(\)](#) in order to work. If no matching [fmi1_import_t](#) struct is found on the global list then [jm_get_default_callbacks\(\)](#) is used to get the default logger. Note that this function is not thread safe due to the use of the global list.

6.28.2.4 FMILIB_EXPORT void fmi1_log_forwarding_v (fmi1_component_t *c*, fmi1_string_t *instanceName*, fmi1_status_t *status*, fmi1_string_t *category*, fmi1_string_t *message*, va_list *args*)

An implementation of FMI 1.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.

See [fmi1_log_forwarding\(\)](#) for more information.

6.28.2.5 FMILIB_EXPORT void fmi1_default_callback_logger (fmi1_component_t c, fmi1_string_t instanceName, fmi1_status_t status, fmi1_string_t category, fmi1_string_t message, ...)

Default FMI 1.0 logger may be used when instantiating FMUs.

6.28.2.6 FMILIB_EXPORT void fmi1_import_init_logger (jm_callbacks * cb, fmi1_callback_functions_t * fmiCallbacks)

Given [fmi1_callback_functions_t](#) logger (fmi1_logger), the [jm_callbacks](#) logger may be setup to redirect the messages to the fmi1_logger.

The functions sets up the redirection. Note that the context field in [jm_callbacks](#) is set to point to the provided [fmi1_callback_functions_t](#).

Parameters

<i>cb</i>	FMI Library callbacks
<i>fmiCallbacks</i>	FMI 1.0 standard callbacks

6.29 Functions to retrieve co-simulation related information.

Functions

- **FMILIB_EXPORT** const char * fmi1_import_get_entry_point (fmi1_import_t *)
Get the entry point.
- **FMILIB_EXPORT** const char * fmi1_import_get_mime_type (fmi1_import_t *)
Get mime type.
- **FMILIB_EXPORT** int fmi1_import_get_manual_start (fmi1_import_t *)
Get manual start flag.
- **FMILIB_EXPORT** size_t fmi1_import_get_number_of_additional_models (fmi1_import_t *fmu)
Get the number of additional models specified.
- **FMILIB_EXPORT** const char * fmi1_import_get_additional_model_name (fmi1_import_t *fmu, size_t index)
Get the name of an additional model.

6.29.1 Function Documentation

6.29.1.1 **FMILIB_EXPORT** const char* fmi1_import_get_entry_point (fmi1_import_t *)

Get the entry point.

6.29.1.2 **FMILIB_EXPORT** const char* fmi1_import_get_mime_type (fmi1_import_t *)

Get mime type.

6.29.1.3 **FMILIB_EXPORT** int fmi1_import_get_manual_start (fmi1_import_t *)

Get manual start flag.

6.29.1.4 **FMILIB_EXPORT** size_t fmi1_import_get_number_of_additional_models (fmi1_import_t * fmu)

Get the number of additional models specified.

6.29.1.5 **FMILIB_EXPORT** const char* fmi1_import_get_additional_model_name (fmi1_import_t * fmu, size_t index)

Get the name of an additional model.

Parameters

<i>fmu</i>	- the FMU processed
<i>index</i>	- the index of an additional model (must be less than the number returned by fmi1_import_get_number_of_additional_models()).

6.30 Support for processing variable types

Functions

- `FMILIB_EXPORT size_t fmi1_import_get_type_definition_number (fmi1_import_type_definitions_t *td)`
Get the number of available type definitions.
- `FMILIB_EXPORT fmi1_import_variable_typedef_t * fmi1_import_get_typedef (fmi1_import_type_definitions_t *td, unsigned int index)`
Get a type definition specified by the index.
- `FMILIB_EXPORT fmi1_import_display_unit_t * fmi1_import_get_type_display_unit (fmi1_import_real_typedef_t *)`
Get associated display unit for a type definition if any.
- `FMILIB_EXPORT const char * fmi1_import_get_type_name (fmi1_import_variable_typedef_t *)`
Get the type name.
- `FMILIB_EXPORT const char * fmi1_import_get_type_description (fmi1_import_variable_typedef_t *)`
Get type description.
- `FMILIB_EXPORT fmi1_base_type_enumeration_t fmi1_import_get_base_type (fmi1_import_variable_typedef_t *)`
Get base type used for the type definition.
- `FMILIB_EXPORT fmi1_import_real_typedef_t * fmi1_import_get_type_as_real (fmi1_import_variable_typedef_t *)`
Cast the general type definition object to an object with a specific base type.
- `FMILIB_EXPORT fmi1_import_integer_typedef_t * fmi1_import_get_type_as_int (fmi1_import_variable_typedef_t *)`
Cast the general type definition object to an object with a specific base type.
- `FMILIB_EXPORT fmi1_import_enumeration_typedef_t * fmi1_import_get_type_as_enum (fmi1_import_variable_typedef_t *)`
Cast the general type definition object to an object with a specific base type.
- `FMILIB_EXPORT const char * fmi1_import_get_type_quantity (fmi1_import_variable_typedef_t *)`
Get the quantity associated with the type definition.
- `FMILIB_EXPORT double fmi1_import_get_real_type_min (fmi1_import_real_typedef_t *)`
Get minimal value for the type.
- `FMILIB_EXPORT double fmi1_import_get_real_type_max (fmi1_import_real_typedef_t *)`
Get maximum value for the type.
- `FMILIB_EXPORT double fmi1_import_get_real_type_nominal (fmi1_import_real_typedef_t *)`
Get the nominal value associated with the type definition.
- `FMILIB_EXPORT fmi1_import_unit_t * fmi1_import_get_real_type_unit (fmi1_import_real_typedef_t *)`

Get the unit object associated with the type definition if any.

- `FMILIB_EXPORT int fmi1_import_get_real_type_is_relative_quantity (fmi1_import_real_typedef_t *)`

Get the relativeQuantity flag.

- `FMILIB_EXPORT int fmi1_import_get_integer_type_min (fmi1_import_integer_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT int fmi1_import_get_integer_type_max (fmi1_import_integer_typedef_t *)`

Get maximum value for the type.

- `FMILIB_EXPORT unsigned int fmi1_import_get_enum_type_min (fmi1_import_enumeration_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT unsigned int fmi1_import_get_enum_type_max (fmi1_import_enumeration_typedef_t *)`

Get maximum value for the type.

- `FMILIB_EXPORT unsigned int fmi1_import_get_enum_type_size (fmi1_import_enumeration_typedef_t *)`

Get the number of elements in the enum.

- `FMILIB_EXPORT const char * fmi1_import_get_enum_type_item_name (fmi1_import_enumeration_typedef_t *, unsigned int item)`

Get an enumeration item name by index.

- `FMILIB_EXPORT const char * fmi1_import_get_enum_type_item_description (fmi1_import_enumeration_typedef_t *, unsigned int item)`

Get an enumeration item description by index.

Type definitions supporting structures

- `typedef struct fmi1_xml_real_typedef_t fmi1_import_real_typedef_t`

Opaque type definition object.

- `typedef struct fmi1_xml_integer_typedef_t fmi1_import_integer_typedef_t`

Opaque integer type definition object.

- `typedef struct fmi1_xml_enumeration_typedef_t fmi1_import_enumeration_typedef_t`

Opaque enumeration type definition object.

- `typedef struct fmi1_xml_variable_typedef_t fmi1_import_variable_typedef_t`

Opaque general variable type definition object.

- `typedef struct fmi1_xml_type_definitions_t fmi1_import_type_definitions_t`

Opaque list of the type definitions in the model.

6.30.1 Typedef Documentation

6.30.1.1 `typedef struct fmi1_xml_real_typedef_t fmi1_import_real_typedef_t`

Opaque type definition object.

Definition at line 42 of file `fmi1_import_type.h`.

6.30.1.2 `typedef struct fmi1_xml_integer_typedef_t fmi1_import_integer_typedef_t`

Opaque integer type definition object.

Definition at line 44 of file `fmi1_import_type.h`.

6.30.1.3 `typedef struct fmi1_xml_enumeration_typedef_t fmi1_import_enumeration_ - typedef_t`

Opaque enumeration type definition object.

Definition at line 46 of file `fmi1_import_type.h`.

6.30.1.4 `typedef struct fmi1_xml_variable_typedef_t fmi1_import_variable_typedef_t`

Opaque general variable type definition object.

Definition at line 48 of file `fmi1_import_type.h`.

6.30.1.5 `typedef struct fmi1_xml_type_definitions_t fmi1_import_type_definitions_t`

Opaque list of the type definitions in the model.

Definition at line 50 of file `fmi1_import_type.h`.

6.30.2 Function Documentation

6.30.2.1 `FMILIB_EXPORT size_t fmi1_import_get_type_definition_number ( fmi1_import_type_definitions_t * td )`

Get the number of available type definitions.

6.30.2.2 `FMILIB_EXPORT fmi1_import_variable_typedef_t* fmi1_import_get_typedef ( fmi1_import_type_definitions_t * td, unsigned int index )`

Get a type definition specified by the index.

Parameters

<i>td</i>	the type definition list object
<i>index</i>	the index of type definition. Must be less than the numbere returned by fmi1_import_get_type_definition_number()

Returns

A type definition object or NULL if index is out of range.

6.30.2.3 FMILIB_EXPORT fmi1_import_display_unit_t *
fmi1_import_get_type_display_unit (fmi1_import_real_typedef_t *)

Get associated display unit for a type defition if any.

Get display unit associated with a real type definition.

Returns

Display unit object of NULL if none was given.

6.30.2.4 FMILIB_EXPORT const char* fmi1_import_get_type_name (
fmi1_import_variable_typedef_t *)

Get the type name.

6.30.2.5 FMILIB_EXPORT const char* fmi1_import_get_type_description (
fmi1_import_variable_typedef_t *)

Get type description.

Note that an empty string is returned if the attribute is not present in the XML.

6.30.2.6 FMILIB_EXPORT fmi1_base_type_enumeration_t fmi1_import_get_base_type (
fmi1_import_variable_typedef_t *)

Get base type used for the type definition.

6.30.2.7 FMILIB_EXPORT fmi1_import_real_typedef_t*
fmi1_import_get_type_as_real (fmi1_import_variable_typedef_t *)

Cast the general type definition object to an object with a specific base type.

Returns

Pointer to the specific type object or NULL if base type does not match.

6.30.2.8 FMILIB_EXPORT fmi1_import_integer_typedef_t*
fmi1_import_get_type_as_int (fmi1_import_variable_typedef_t *)

Cast the general type definition object to an object with a specific base type.

Returns

Pointer to the specific type object or NULL if base type does not match.

6.30.2.9 FMILIB_EXPORT fmi1_import_enumeration_typedef_t*
fmi1_import_get_type_as_enum (fmi1_import_variable_typedef_t *)

Cast the general type definition object to an object with a specific base type.

Returns

Pointer to the specific type object or NULL if base type does not match.

6.30.2.10 FMILIB_EXPORT const char* fmi1_import_get_type_quantity (
fmi1_import_variable_typedef_t *)

Get the quantity associated with the type definition.

Returns

NULL-pointer is always returned for strings and booleans. Empty string is returned if attribute is not present for other types.

6.30.2.11 FMILIB_EXPORT double fmi1_import_get_real_type_min (
fmi1_import_real_typedef_t *)

Get minimal value for the type.

Returns

Either the value specified in the XML file or negated DBL_MAX as defined in <float.h>

6.30.2.12 FMILIB_EXPORT double fmi1_import_get_real_type_max (
fmi1_import_real_typedef_t *)

Get maximum value for the type.

Returns

Either the value specified in the XML file or DBL_MAX as defined in <float.h>

6.30.2.13 FMILIB_EXPORT double fmi1_import_get_real_type_nominal (fmi1_import_real_typedef_t *)

Get the nominal value associated with the type definition.

6.30.2.14 FMILIB_EXPORT fmi1_import_unit_t* fmi1_import_get_real_type_unit (fmi1_import_real_typedef_t *)

Get the unit object associated with the type definition if any.

6.30.2.15 FMILIB_EXPORT int fmi1_import_get_real_type_is_relative_quantity (fmi1_import_real_typedef_t *)

Get the relativeQuantity flag.

6.30.2.16 FMILIB_EXPORT int fmi1_import_get_integer_type_min (fmi1_import_integer_typedef_t *)

Get minimal value for the type.

Returns

Either the value specified in the XML file or INT_MIN as defined in <limits.h>

6.30.2.17 FMILIB_EXPORT int fmi1_import_get_integer_type_max (fmi1_import_integer_typedef_t *)

Get maximum value for the type.

Returns

Either the value specified in the XML file or INT_MAX as defined in <limits.h>

6.30.2.18 FMILIB_EXPORT unsigned int fmi1_import_get_enum_type_min (fmi1_import_enumeration_typedef_t *)

Get minimal value for the type.

Returns

Either the value specified in the XML file or 0

6.30.2.19 **FMILIB_EXPORT** unsigned int fmi1_import_get_enum_type_max (fmi1_import_enumeration_typedef_t *)

Get maximum value for the type.

Returns

Either the value specified in the XML file or INT_MAX as defined in <limits.h>

6.30.2.20 **FMILIB_EXPORT** unsigned int fmi1_import_get_enum_type_size (fmi1_import_enumeration_typedef_t *)

Get the number of elements in the enum.

6.30.2.21 **FMILIB_EXPORT** const char* fmi1_import_get_enum_type_item_name (fmi1_import_enumeration_typedef_t * , unsigned int *item*)

Get an enumeration item name by index.

6.30.2.22 **FMILIB_EXPORT** const char* fmi1_import_get_enum_type_item_description (fmi1_import_enumeration_typedef_t * , unsigned int *item*)

Get an enumeration item description by index.

6.31 Functions for handling unit definitions.

Functions

- `FMILIB_EXPORT unsigned int fmi1_import_get_unit_definitions_number (fmi1_import_unit_definitions_t *)`
Get the number of unit definitions.
- `FMILIB_EXPORT fmi1_import_unit_t * fmi1_import_get_unit (fmi1_import_unit_definitions_t *, unsigned int index)`
Get a unit definition.
- `FMILIB_EXPORT const char * fmi1_import_get_unit_name (fmi1_import_unit_t *)`
Get a unit name.
- `FMILIB_EXPORT unsigned int fmi1_import_get_unit_display_unit_number (fmi1_import_unit_t *)`
Get the number of display units associated with this unit.
- `FMILIB_EXPORT fmi1_import_display_unit_t * fmi1_import_get_unit_display_unit (fmi1_import_unit_t *, size_t index)`
Get a display unit object by index.
- `FMILIB_EXPORT fmi1_import_unit_t * fmi1_import_get_base_unit (fmi1_import_display_unit_t *)`
Get unit definition for a display unit.
- `FMILIB_EXPORT const char * fmi1_import_get_display_unit_name (fmi1_import_display_unit_t *)`
Get display unit name.
- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_display_unit_gain (fmi1_import_display_unit_t *)`
Get the "gain" associated with the display unit.
- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_display_unit_offset (fmi1_import_display_unit_t *)`
Get the "offset" associated with the display unit.
- `FMILIB_EXPORT fmi1_real_t fmi1_import_convert_to_display_unit (fmi1_real_t value, fmi1_import_display_unit_t *du, int isRelativeQuantity)`
Convert a value measured in "units" to a value measured with "display units".
- `FMILIB_EXPORT fmi1_real_t fmi1_import_convert_from_display_unit (fmi1_real_t value, fmi1_import_display_unit_t *du, int isRelativeQuantity)`
Convert a value measured in "display units" to a value measured with "units".

Structures encapsulating unit information

- `typedef struct fmi1_xml_unit_t fmi1_import_unit_t`
A variable unit defined with a unit definition.
- `typedef struct fmi1_xml_display_unit_t fmi1_import_display_unit_t`
A display unit.
- `typedef struct fmi1_xml_unit_definitions_t fmi1_import_unit_definitions_t`
The list of all the unit definitions in the model.

6.31.1 Typedef Documentation

6.31.1.1 `typedef struct fmi1_xml_unit_t fmi1_import_unit_t`

A variable unit defined with a unit definition.

Definition at line 41 of file `fmi1_import_unit.h`.

6.31.1.2 `typedef struct fmi1_xml_display_unit_t fmi1_import_display_unit_t`

A display unit.

Definition at line 43 of file `fmi1_import_unit.h`.

6.31.1.3 `typedef struct fmi1_xml_unit_definitions_t fmi1_import_unit_definitions_t`

The list of all the unit definitions in the model.

Definition at line 45 of file `fmi1_import_unit.h`.

6.31.2 Function Documentation

6.31.2.1 `FMILIB_EXPORT unsigned int fmi1_import_get_unit_definitions_number ( fmi1_import_unit_definitions_t * )`

Get the number of unit definitions.

6.31.2.2 `FMILIB_EXPORT fmi1_import_unit_t* fmi1_import_get_unit ( fmi1_import_unit_definitions_t *, unsigned int index )`

Get a unit definition.

6.31.2.3 `FMILIB_EXPORT const char* fmi1_import_get_unit_name ( fmi1_import_unit_t * )`

Get a unit name.

6.31.2.4 `FMILIB_EXPORT unsigned int fmi1_import_get_unit_display_unit_number ( fmi1_import_unit_t * )`

Get the number of display units associated with this unit.

6.31.2.5 **FMILIB_EXPORT** `fmi1_import_display_unit_t*` `fmi1_import_get_unit_display_unit ( fmi1_import_unit_t * , size_t index )`

Get a display unit object by index.

Parameters

<i>index</i>	The index of display unit to be returned. Must be less than the number returned by <code>fmi1_import_get_unit_display_unit_number()</code>
--------------	--

6.31.2.6 **FMILIB_EXPORT** `fmi1_import_unit_t*` `fmi1_import_get_base_unit ( fmi1_import_display_unit_t * )`

Get unit definition for a display unit.

6.31.2.7 **FMILIB_EXPORT** `const char*` `fmi1_import_get_display_unit_name ( fmi1_import_display_unit_t * )`

Get display unit name.

6.31.2.8 **FMILIB_EXPORT** `fmi1_real_t` `fmi1_import_get_display_unit_gain ( fmi1_import_display_unit_t * )`

Get the "gain" associated with the display unit.

6.31.2.9 **FMILIB_EXPORT** `fmi1_real_t` `fmi1_import_get_display_unit_offset ( fmi1_import_display_unit_t * )`

Get the "offset" associated with the display unit.

6.31.2.10 **FMILIB_EXPORT** `fmi1_real_t` `fmi1_import_convert_to_display_unit ( fmi1_real_t value, fmi1_import_display_unit_t * du, int isRelativeQuantity )`

Convert a value measured in "units" to a value measured with "display units".

Parameters

<i>value</i>	The value to be converted.
<i>du</i>	The display unit object
<i>isRelative-Quantity</i>	specifies if "offset" should be incorporated into conversion

6.31.2.11 **FMLIB_EXPORT** fmi1_real_t fmi1_import_convert_from_display_unit (fmi1_real_t *value*, fmi1_import_display_unit_t * *du*, int *isRelativeQuantity*)

Convert a value measured in "display units" to a value measured with "units".

Parameters

<i>value</i>	The value to be converted.
<i>du</i>	The display unit object
<i>isRelative-Quantity</i>	specifies if "offset" should be incorporated into conversion

6.32 Functions for handling variable definitions.

All the functions in this group take a pointer to `fmi1_import_variable_t` as a parameter. A variable pointer may be obtained via a [Handling of variable lists](#) module or via functions `fmi1_import_get_variable_by_name()` and `fmi1_import_get_variable_by_vr()`.

Functions

- `FMILIB_EXPORT fmi1_import_variable_t * fmi1_import_get_variable_by_name (fmi1_import_t *fmu, const char *name)`
Get variable by variable name.
- `FMILIB_EXPORT fmi1_import_variable_t * fmi1_import_get_variable_by_vr (fmi1_import_t *fmu, fmi1_base_type_enumer_t baseType, fmi1_value_reference_t vr)`
Get variable by value reference.
- `FMILIB_EXPORT const char * fmi1_import_get_variable_name (fmi1_import_variable_t *)`
Get the variable name.
- `FMILIB_EXPORT const char * fmi1_import_get_variable_description (fmi1_import_variable_t *)`
Get variable description.
- `FMILIB_EXPORT fmi1_value_reference_t fmi1_import_get_variable_vr (fmi1_import_variable_t *)`
Get variable value reference.
- `FMILIB_EXPORT fmi1_import_variable_typedef_t * fmi1_import_get_variable_declared_type (fmi1_import_variable_t *)`
For scalar variable gives the type definition is present.
- `FMILIB_EXPORT fmi1_base_type_enumer_t fmi1_import_get_variable_base_type (fmi1_import_variable_t *)`
Get variable base type.
- `FMILIB_EXPORT int fmi1_import_get_variable_has_start (fmi1_import_variable_t *)`
Check if the variable has "start" attribute.
- `FMILIB_EXPORT int fmi1_import_get_variable_is_fixed (fmi1_import_variable_t *)`
Get the variable "fixed" attribute.
- `FMILIB_EXPORT fmi1_variability_enumer_t fmi1_import_get_variability (fmi1_import_variable_t *)`
Get variability attribute.
- `FMILIB_EXPORT fmi1_causality_enumer_t fmi1_import_get_causality (fmi1_import_variable_t *)`
Get causality attribute.
- `FMILIB_EXPORT fmi1_import_real_variable_t * fmi1_import_get_variable_as_real (fmi1_import_variable_t *)`
Cast general variable to a one with the specific type.

- `FMILIB_EXPORT fmi1_import_integer_variable_t * fmi1_import_get_variable_as_integer (fmi1_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi1_import_enum_variable_t * fmi1_import_get_variable_as_enum (fmi1_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi1_import_string_variable_t * fmi1_import_get_variable_as_string (fmi1_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi1_import_bool_variable_t * fmi1_import_get_variable_as_boolean (fmi1_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_start (fmi1_import_real_variable_t *v)`
Get the variable start attribute.
- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_max (fmi1_import_real_variable_t *v)`
Get maximum value for the variable.
- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_min (fmi1_import_real_variable_t *v)`
Get minimal value for the variable.
- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_nominal (fmi1_import_real_variable_t *v)`
Get nominal value for the variable.
- `FMILIB_EXPORT fmi1_import_unit_t * fmi1_import_get_real_variable_unit (fmi1_import_real_variable_t *v)`
Get associated "unit" object if any.
- `FMILIB_EXPORT fmi1_import_display_unit_t * fmi1_import_get_real_variable_display_unit (fmi1_import_real_variable_t *v)`
Get associated "display unit" object if any.
- `FMILIB_EXPORT const char * fmi1_import_get_string_variable_start (fmi1_import_string_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT fmi1_boolean_t fmi1_import_get_boolean_variable_start (fmi1_import_bool_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT int fmi1_import_get_integer_variable_start (fmi1_import_integer_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT int fmi1_import_get_integer_variable_min (fmi1_import_integer_variable_t *v)`
Get minimal value for the variable.
- `FMILIB_EXPORT int fmi1_import_get_integer_variable_max (fmi1_import_integer_variable_t *v)`
Get max value for the variable.

- `FMILIB_EXPORT int fmi1_import_get_enum_variable_start (fmi1_import_enum_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT int fmi1_import_get_enum_variable_min (fmi1_import_enum_variable_t *v)`
Get minimal value for the variable.
- `FMILIB_EXPORT int fmi1_import_get_enum_variable_max (fmi1_import_enum_variable_t *v)`
Get max value for the variable.
- `FMILIB_EXPORT fmi1_variable_alias_kind_enumeration_t fmi1_import_get_variable_alias_kind (fmi1_import_variable_t *)`
Get the variable alias kind.
- `size_t fmi1_import_get_variable_original_order (fmi1_import_variable_t *v)`
Get the original index in xml of the variable.

Scalar variable types

- `typedef struct fmi1_xml_variable_t fmi1_import_variable_t`
General variable type.
- `typedef struct fmi1_xml_real_variable_t fmi1_import_real_variable_t`
Opaque real variable.
- `typedef struct fmi1_xml_integer_variable_t fmi1_import_integer_variable_t`
Opaque integer variable.
- `typedef struct fmi1_xml_string_variable_t fmi1_import_string_variable_t`
Opaque string variable.
- `typedef struct fmi1_xml_enum_variable_t fmi1_import_enum_variable_t`
Opaque enumeration variable.
- `typedef struct fmi1_xml_bool_variable_t fmi1_import_bool_variable_t`
Opaque boolean variable.
- `typedef struct fmi1_import_variable_list_t fmi1_import_variable_list_t`
List of variables.

6.32.1 Detailed Description

All the functions in this group take a pointer to `fmi1_import_variable_t` as a parameter. A variable pointer may be obtained via a [Handling of variable lists](#) module or via functions `fmi1_import_get_variable_by_name()` and `fmi1_import_get_variable_by_vr()`.

6.32.2 Typedef Documentation

6.32.2.1 `typedef struct fmi1_xml_variable_t fmi1_import_variable_t`

General variable type.

This type is convenient to unify all the variable list operations. However, typed variables are needed to support specific attributes.

Definition at line 51 of file `fmi1_import_variable.h`.

6.32.2.2 `typedef struct fmi1_xml_real_variable_t fmi1_import_real_variable_t`

Opaque real variable.

Definition at line 53 of file `fmi1_import_variable.h`.

6.32.2.3 `typedef struct fmi1_xml_integer_variable_t fmi1_import_integer_variable_t`

Opaque integer variable.

Definition at line 55 of file `fmi1_import_variable.h`.

6.32.2.4 `typedef struct fmi1_xml_string_variable_t fmi1_import_string_variable_t`

Opaque string variable.

Definition at line 57 of file `fmi1_import_variable.h`.

6.32.2.5 `typedef struct fmi1_xml_enum_variable_t fmi1_import_enum_variable_t`

Opaque enumeration variable.

Definition at line 59 of file `fmi1_import_variable.h`.

6.32.2.6 `typedef struct fmi1_xml_bool_variable_t fmi1_import_bool_variable_t`

Opaque boolean variable.

Definition at line 61 of file `fmi1_import_variable.h`.

6.32.2.7 `typedef struct fmi1_import_variable_list_t fmi1_import_variable_list_t`

List of variables.

Definition at line 63 of file `fmi1_import_variable.h`.

6.32.3 Function Documentation

6.32.3.1 `FMILIB_EXPORT fmi1_import_variable_t* fmi1_import_get_variable_by_name ( fmi1_import_t * fmu, const char * name )`

Get variable by variable name.

Parameters

<i>fmu</i>	- An fmu object as returned by fmi1_import_parse_xml() .
<i>name</i>	- variable name

Returns

variable pointer.

6.32.3.2 FMILIB_EXPORT `fmi1_import_variable_t* fmi1_import_get_variable_by_vr ( fmi1_import_t * fmu, fmi1_base_type_enu_t baseType, fmi1_value_reference_t vr )`

Get variable by value reference.

Parameters

<i>fmu</i>	- An fmu object as returned by fmi1_import_parse_xml() .
<i>baseType</i>	- basic data type
<i>vr</i>	- value reference

Returns

variable pointer.

6.32.3.3 FMILIB_EXPORT `const char* fmi1_import_get_variable_name ( fmi1_import_variable_t * )`

Get the variable name.

6.32.3.4 FMILIB_EXPORT `const char* fmi1_import_get_variable_description ( fmi1_import_variable_t * )`

Get variable description.

Returns

Description string or empty string ("") if no description in the XML file was given.

6.32.3.5 FMILIB_EXPORT `fmi1_value_reference_t fmi1_import_get_variable_vr ( fmi1_import_variable_t * )`

Get variable value reference.

6.32.3.6 **FMILIB_EXPORT** fmi1_import_variable_typedef_t*
 fmi1_import_get_variable_declared_type (fmi1_import_variable_t *)

For scalar variable gives the type definition is present.

Returns

Pointer of a type [fmi1_import_variable_typedef_t](#) object or NULL if not present.

6.32.3.7 **FMILIB_EXPORT** fmi1_base_type_enumeration_t fmi1_import-
 _get_variable_base_type (fmi1_import_variable_t *
)

Get variable base type.

6.32.3.8 **FMILIB_EXPORT** int fmi1_import_get_variable_has_start (fmi1_import_variable_t *)

Check if the variable has "start" attribute.

6.32.3.9 **FMILIB_EXPORT** int fmi1_import_get_variable_is_fixed (fmi1_import_variable_t *)

Get the variable "fixed" attribute.

6.32.3.10 **FMILIB_EXPORT** fmi1_variability_enumeration_t fmi1_import_get_variability (fmi1_import_variable_t *)

Get variability attribute.

6.32.3.11 **FMILIB_EXPORT** fmi1_causality_enumeration_t fmi1_import_get_causality (fmi1_import_variable_t *)

Get causality attribute.

6.32.3.12 **FMILIB_EXPORT** fmi1_import_real_variable_t*
 fmi1_import_get_variable_as_real (fmi1_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.32.3.13 FMILIB_EXPORT fmi1_import_integer_variable_t*
fmi1_import_get_variable_as_integer (fmi1_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.32.3.14 FMILIB_EXPORT fmi1_import_enum_variable_t*
fmi1_import_get_variable_as_enum (fmi1_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.32.3.15 FMILIB_EXPORT fmi1_import_string_variable_t*
fmi1_import_get_variable_as_string (fmi1_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.32.3.16 FMILIB_EXPORT fmi1_import_bool_variable_t*
fmi1_import_get_variable_as_boolean (fmi1_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.32.3.17 FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_start (
fmi1_import_real_variable_t * v)

Get the variable start attribute.

Returns

The "start" attriburte as specified in the XML file or variable nominal value.

6.32.3.18 FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_max (fmi1_import_real_variable_t * v)

Get maximum value for the variable.

Returns

Either the value specified in the XML file or DBL_MAX as defined in <float.h>

6.32.3.19 FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_min (fmi1_import_real_variable_t * v)

Get minimal value for the variable.

Returns

Either the value specified in the XML file or negated DBL_MAX as defined in <float.h>

6.32.3.20 FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_nominal (fmi1_import_real_variable_t * v)

Get nominal value for the variable.

6.32.3.21 FMILIB_EXPORT fmi1_import_unit_t* fmi1_import_get_real_variable_unit (fmi1_import_real_variable_t * v)

Get associated "unit" object if any.

6.32.3.22 FMILIB_EXPORT fmi1_import_display_unit_t* fmi1_import_get_real_variable_display_unit (fmi1_import_real_variable_t * v)

Get associated "display unit" object if any.

6.32.3.23 FMILIB_EXPORT const char* fmi1_import_get_string_variable_start (fmi1_import_string_variable_t * v)

Get start value for the variable.

6.32.3.24 FMILIB_EXPORT fmi1_boolean_t fmi1_import_get_boolean_variable_start (fmi1_import_bool_variable_t * v)

Get start value for the variable.

6.32.3.25 FMILIB_EXPORT int fmi1_import_get_integer_variable_start (
fmi1_import_integer_variable_t * v)

Get start value for the variable.

6.32.3.26 FMILIB_EXPORT int fmi1_import_get_integer_variable_min (
fmi1_import_integer_variable_t * v)

Get minimal value for the variable.

6.32.3.27 FMILIB_EXPORT int fmi1_import_get_integer_variable_max (
fmi1_import_integer_variable_t * v)

Get max value for the variable.

6.32.3.28 FMILIB_EXPORT int fmi1_import_get_enum_variable_start (
fmi1_import_enum_variable_t * v)

Get start value for the variable.

6.32.3.29 FMILIB_EXPORT int fmi1_import_get_enum_variable_min (
fmi1_import_enum_variable_t * v)

Get minimal value for the variable.

6.32.3.30 FMILIB_EXPORT int fmi1_import_get_enum_variable_max (
fmi1_import_enum_variable_t * v)

Get max value for the variable.

6.32.3.31 FMILIB_EXPORT fmi1_variable_alias_kind_enumeration_t
fmi1_import_get_variable_alias_kind (fmi1_import_variable_t * v)

Get the variable alias kind.

6.32.3.32 size_t fmi1_import_get_variable_original_order (
fmi1_import_variable_t * v)

Get the original index in xml of the variable.

6.33 Basic support for vendor annotations.

Functions

- `FMILIB_EXPORT unsigned int fmi1_import_get_number_of_vendors (fmi1_import_vendor_list_t *)`
Get the number of different vendors.
- `FMILIB_EXPORT fmi1_import_vendor_t * fmi1_import_get_vendor (fmi1_import_vendor_list_t *, unsigned int index)`
Get the annotations associated with vendor specified by the index.
- `FMILIB_EXPORT const char * fmi1_import_get_vendor_name (fmi1_import_vendor_t *)`
Get the vendor name.
- `FMILIB_EXPORT unsigned int fmi1_import_get_number_of_vendor_annotations (fmi1_import_vendor_t *)`
Get the number of annotations provided for the vendor.
- `FMILIB_EXPORT fmi1_import_annotation_t * fmi1_import_get_vendor_annotation (fmi1_import_vendor_t *, unsigned int index)`
Get an annotation object for the vendor by index.
- `FMILIB_EXPORT const char * fmi1_import_get_annotation_name (fmi1_import_annotation_t *)`
Get the name of the annotation.
- `FMILIB_EXPORT const char * fmi1_import_get_annotation_value (fmi1_import_annotation_t *)`
Get the value for the annotation.
- `FMILIB_EXPORT fmi1_import_vendor_list_t * fmi1_import_get_vendor_list (fmi1_import_t *fmu)`
Get the list of all the vendor annotations present in the XML file.

Vendor annotation supporting structures

- `typedef struct fmi1_xml_vendor_list_t fmi1_import_vendor_list_t`
Opaque list of vendor annotations.
- `typedef struct fmi1_xml_vendor_t fmi1_import_vendor_t`
Opaque vendor object.
- `typedef struct fmi1_xml_annotation_t fmi1_import_annotation_t`
Opaque annotation object.

6.33.1 Typedef Documentation

6.33.1.1 `typedef struct fmi1_xml_vendor_list_t fmi1_import_vendor_list_t`

Opaque list of vendor annotations.

Definition at line 43 of file `fmi1_import_vendor_annotations.h`.

6.33.1.2 `typedef struct fmi1_xml_vendor_t fmi1_import_vendor_t`

Opaque vendor object.

Definition at line 45 of file `fmi1_import_vendor_annotations.h`.

6.33.1.3 `typedef struct fmi1_xml_annotation_t fmi1_import_annotation_t`

Opaque annotation object.

Definition at line 47 of file `fmi1_import_vendor_annotations.h`.

6.33.2 Function Documentation

6.33.2.1 `FMILIB_EXPORT unsigned int fmi1_import_get_number_of_vendors ( fmi1_import_vendor_list_t * )`

Get the number of different vendors.

6.33.2.2 `FMILIB_EXPORT fmi1_import_vendor_t* fmi1_import_get_vendor ( fmi1_import_vendor_list_t *, unsigned int index )`

Get the annotations associated with vendor specified by the index.

6.33.2.3 `FMILIB_EXPORT const char* fmi1_import_get_vendor_name ( fmi1_import_vendor_t * )`

Get the vendor name.

6.33.2.4 `FMILIB_EXPORT unsigned int fmi1_import_get_number_of_vendor_annotations ( fmi1_import_vendor_t * )`

Get the number of annotations provided for the vendor.

6.33.2.5 `FMILIB_EXPORT fmi1_import_annotation_t* fmi1_import_get_vendor_annotation ( fmi1_import_vendor_t *, unsigned int index )`

Get an annotation object for the vendor by index.

Note: Annotations may later be used in other places but have common interface name-value

6.33.2.6 FMILIB_EXPORT `const char* fmi1_import_get_annotation_name ( fmi1_import_annotation_t * )`

Get the name of the annotation.

6.33.2.7 FMILIB_EXPORT `const char* fmi1_import_get_annotation_value ( fmi1_import_annotation_t * )`

Get the value for the annotation.

6.33.2.8 FMILIB_EXPORT `fmi1_import_vendor_list_t* fmi1_import_get_vendor_list ( fmi1_import_t * fmu )`

Get the list of all the vendor annotations present in the XML file.

6.34 Constuction, destruction and error handling

Functions

- **FMILIB_EXPORT** const char * fmi2_import_get_last_error (fmi2_import_t *fmu)
Retrieve the last error message.
- **FMILIB_EXPORT** int fmi2_import_clear_last_error (fmi2_import_t *fmu)
Clear the error message.
- **FMILIB_EXPORT** void fmi2_import_free (fmi2_import_t *fmu)
Release the memory allocated.

6.34.1 Function Documentation

6.34.1.1 **FMILIB_EXPORT** const char* fmi2_import_get_last_error (fmi2_import_t * fmu)

Retrieve the last error message.

Error handling:

Many functions in the library return pointers to struct. An error is indicated by returning NULL/0-pointer. If error is returned than [fmi2_import_get_last_error\(\)](#) functions can be used to retrieve the error message. If logging callbacks were specified then the same information is reported via logger. Memory for the error string is allocated and deallocated in the module. Client code should not store the pointer to the string since it can become invalid.

Parameters

<i>fmu</i>	An FMU object as returned by fmi2_import_parse_xml() .
------------	--

Returns

NULL-terminated string with an error message.

6.34.1.2 **FMILIB_EXPORT** int fmi2_import_clear_last_error (fmi2_import_t * fmu)

Clear the error message.

Parameters

<i>fmu</i>	An FMU object as returned by fmi2_import_parse_xml() .
------------	--

Returns

0 if further processing is possible. If it returns 1 then the error was not recoverable.
The `fmu` object should then be freed and recreated.

6.34.1.3 FMILIB_EXPORT void fmi2_import_free (fmi2_import_t * *fmu*)

Release the memory allocated.

Parameters

<i>fmu</i>	An fmu object as returned by fmi2_import_parse_xml() .
------------	--

6.35 General information retrieval

Functions for retrieving general model information. Memory for the strings is allocated and deallocated in the module. All the functions take an FMU object as returned by `fmi2_import_parse_xml()` as a parameter. The information is retrieved from the XML file.

Functions

- `FMILIB_EXPORT const char * fmi2_import_get_model_name (fmi2_import_t *fmu)`
Get model name.
- `FMILIB_EXPORT unsigned int fmi2_import_get_capability (fmi2_import_t *, fmi2_capabilities_enu_t id)`
Retrieve capability flags by ID.
- `FMILIB_EXPORT const char * fmi2_import_get_model_identifier_ME (fmi2_import_t *fmu)`
Get model identifier for ModelExchange.
- `FMILIB_EXPORT const char * fmi2_import_get_model_identifier_CS (fmi2_import_t *fmu)`
Get model identifier for CoSimulation.
- `FMILIB_EXPORT const char * fmi2_import_get_GUID (fmi2_import_t *fmu)`
Get FMU GUID.
- `FMILIB_EXPORT const char * fmi2_import_get_description (fmi2_import_t *fmu)`
Get FMU description.
- `FMILIB_EXPORT const char * fmi2_import_get_author (fmi2_import_t *fmu)`
Get FMU author.
- `FMILIB_EXPORT const char * fmi2_import_get_copyright (fmi2_import_t *fmu)`
Get FMU copyright information.
- `FMILIB_EXPORT const char * fmi2_import_get_license (fmi2_import_t *fmu)`
Get FMU license information.
- `FMILIB_EXPORT const char * fmi2_import_get_model_version (fmi2_import_t *fmu)`
Get FMU version.
- `FMILIB_EXPORT const char * fmi2_import_get_model_standard_version (fmi2_import_t *fmu)`
Get FMI standard version (always 2.0).
- `FMILIB_EXPORT const char * fmi2_import_get_generation_tool (fmi2_import_t *fmu)`
Get FMU generation tool.
- `FMILIB_EXPORT const char * fmi2_import_get_generation_date_and_time (fmi2_import_t *fmu)`

Get FMU generation date and time.

- FMILIB_EXPORT fmi2_variable_naming_convention_enu_t fmi2_import_get_naming_convention (fmi2_import_t *fmu)

Get variable naming convention used.

- FMILIB_EXPORT size_t fmi2_import_get_number_of_continuous_states (fmi2_import_t *fmu)

Get the number of continuous states.

- FMILIB_EXPORT size_t fmi2_import_get_number_of_event_indicators (fmi2_import_t *fmu)

Get the number of event indicators.

- FMILIB_EXPORT double fmi2_import_get_default_experiment_start (fmi2_import_t *fmu)

Get the start time for default experiment as specified in the XML file.

- FMILIB_EXPORT double fmi2_import_get_default_experiment_stop (fmi2_import_t *fmu)

Get the stop time for default experiment as specified in the XML file.

- FMILIB_EXPORT double fmi2_import_get_default_experiment_tolerance (fmi2_import_t *fmu)

Get the tolerance for default experiment as specified in the XML file.

- FMILIB_EXPORT double fmi2_import_get_default_experiment_step (fmi2_import_t *fmu)

Get the step size for default experiment as specified in the XML file.

- FMILIB_EXPORT fmi2_fmu_kind_enu_t fmi2_import_get_fmu_kind (fmi2_import_t *fmu)

Get the type of the FMU (model exchange or co-simulation)

- FMILIB_EXPORT fmi2_import_type_definitions_t * fmi2_import_get_type_definitions (fmi2_import_t *)

Get the list of all the type definitions in the model.

- FMILIB_EXPORT fmi2_import_unit_definitions_t * fmi2_import_get_unit_definitions (fmi2_import_t *fmu)

Get a list of all the unit definitions in the model.

- FMILIB_EXPORT fmi2_import_variable_t * fmi2_import_get_variable_alias_base (fmi2_import_t *fmu, fmi2_import_variable_t *)

Get the variable with the same value reference that is not an alias.

- FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_variable_aliases (fmi2_import_t *fmu, fmi2_import_variable_t *)
- FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_variable_list (fmi2_import_t *fmu, int sortOrder)

Get the list of all the variables in the model.

- FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_create_var_list (fmi2_import_t *fmu, fmi2_import_variable_t *v)

Create a variable list with a single variable.

- FMILIB_EXPORT size_t fmi2_import_get_vendors_num (fmi2_import_t *fmu)

Get the number of vendors that had annotations in the XML.

- `FMILIB_EXPORT const char * fmi2_import_get_vendor_name (fmi2_import_t *fmu, size_t index)`
Get the name of the vendor with that had annotations in the XML by index.
- `FMILIB_EXPORT size_t fmi2_import_get_log_categories_num (fmi2_import_t *fmu)`
Get the number of log categories defined in the XML.
- `FMILIB_EXPORT const char * fmi2_import_get_log_category (fmi2_import_t *fmu, size_t index)`
Get the log category by index.
- `FMILIB_EXPORT const char * fmi2_import_get_log_category_description (fmi2_import_t *fmu, size_t index)`
Get the log category description by index.
- `FMILIB_EXPORT size_t fmi2_import_get_source_files_me_num (fmi2_import_t *fmu)`
Get the number of source files for ME defined in the XML.
- `FMILIB_EXPORT const char * fmi2_import_get_source_file_me (fmi2_import_t *fmu, size_t index)`
Get the ME source file by index.
- `FMILIB_EXPORT size_t fmi2_import_get_source_files_cs_num (fmi2_import_t *fmu)`
Get the number of source files for CS defined in the XML.
- `FMILIB_EXPORT const char * fmi2_import_get_source_file_cs (fmi2_import_t *fmu, size_t index)`
Get the CS source file by index.
- `FMILIB_EXPORT fmi2_import_variable_t * fmi2_import_get_variable_by_name (fmi2_import_t *fmu, const char *name)`
Get variable by variable name.
- `FMILIB_EXPORT fmi2_import_variable_t * fmi2_import_get_variable_by_vr (fmi2_import_t *fmu, fmi2_base_type_enu_t baseType, fmi2_value_reference_t vr)`
Get variable by value reference.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_outputs_list (fmi2_import_t *fmu)`
Get the list of all the output variables in the model.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_derivatives_list (fmi2_import_t *fmu)`
Get the list of all the derivative variables in the model.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_discrete_states_list (fmi2_import_t *fmu)`
Get the list of all the discrete state variables in the model.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_initial_unknowns_list (fmi2_import_t *fmu)`
Get the list of all the initial unknown variables in the model.
- `FMILIB_EXPORT void fmi2_import_get_outputs_dependencies (fmi2_import_t *fmu, size_t **startIndex, size_t **dependency, char **factorKind)`

Get dependency information in row-compressed format.

- **FMILIB_EXPORT** void [fmi2_import_get_derivatives_dependencies](#) ([fmi2_import_t](#) *fmu, [size_t](#) **startIndex, [size_t](#) **dependency, [char](#) **factorKind)

Get dependency information in row-compressed format.

- **FMILIB_EXPORT** void [fmi2_import_get_discrete_states_dependencies](#) ([fmi2_import_t](#) *fmu, [size_t](#) **startIndex, [size_t](#) **dependency, [char](#) **factorKind)

Get dependency information in row-compressed format.

- **FMILIB_EXPORT** void [fmi2_import_get_initial_unknowns_dependencies](#) ([fmi2_import_t](#) *fmu, [size_t](#) **startIndex, [size_t](#) **dependency, [char](#) **factorKind)

Get dependency information in row-compressed format.

6.35.1 Detailed Description

Functions for retrieving general model information. Memory for the strings is allocated and deallocated in the module. All the functions take an FMU object as returned by [fmi2_import_parse_xml\(\)](#) as a parameter. The information is retrieved from the XML file.

6.35.2 Function Documentation

6.35.2.1 FMILIB_EXPORT const [char](#)* [fmi2_import_get_model_name](#) ([fmi2_import_t](#) * fmu)

Get model name.

Parameters

fmu	An fmu object as returned by fmi2_import_parse_xml() .
---------------------	--

6.35.2.2 FMILIB_EXPORT unsigned int [fmi2_import_get_capability](#) ([fmi2_import_t](#) *, [fmi2_capabilities_enu_t](#) id)

Retrieve capability flags by ID.

6.35.2.3 FMILIB_EXPORT const [char](#)* [fmi2_import_get_model_identifier_ME](#) ([fmi2_import_t](#) * fmu)

Get model identifier for ModelExchange.

Parameters

fmu	An fmu object as returned by fmi2_import_parse_xml() .
---------------------	--

6.35.2.4 FMILIB_EXPORT const char* fmi2_import_get_model_identifier_CS (fmi2_import_t * fmu)

Get model identifier for CoSimulation.

Parameters

<i>fmu</i>	An fmu object as returned by fmi2_import_parse_xml() .
------------	--

6.35.2.5 FMILIB_EXPORT const char* fmi2_import_get_GUID (fmi2_import_t * fmu)

Get FMU GUID.

Parameters

<i>fmu</i>	An fmu object as returned by fmi2_import_parse_xml() .
------------	--

6.35.2.6 FMILIB_EXPORT const char* fmi2_import_get_description (fmi2_import_t * fmu)

Get FMU description.

Parameters

<i>fmu</i>	An fmu object as returned by fmi2_import_parse_xml() .
------------	--

6.35.2.7 FMILIB_EXPORT const char* fmi2_import_get_author (fmi2_import_t * fmu)

Get FMU author.

Parameters

<i>fmu</i>	An fmu object as returned by fmi2_import_parse_xml() .
------------	--

6.35.2.8 FMILIB_EXPORT const char* fmi2_import_get_copyright (fmi2_import_t * fmu)

Get FMU copyright information.

Parameters

<i>fmu</i>	An fmu object as returned by fmi2_import_parse_xml() .
------------	--

6.35.2.9 FMILIB_EXPORT const char* fmi2_import_get_license (fmi2_import_t *
fmu)

Get FMU license information.

Parameters

fmu	An fmu object as returned by fmi2_import_parse_xml() .
-----	--

6.35.2.10 FMILIB_EXPORT const char* fmi2_import_get_model_version (fmi2_import_t *
fmu)

Get FMU version.

Parameters

fmu	An fmu object as returned by fmi2_import_parse_xml() .
-----	--

6.35.2.11 FMILIB_EXPORT const char* fmi2_import_get_model_standard_version (fmi2_import_t *
fmu)

Get FMI standard version (always 2.0).

Parameters

fmu	An fmu object as returned by fmi2_import_parse_xml() .
-----	--

6.35.2.12 FMILIB_EXPORT const char* fmi2_import_get_generation_tool (fmi2_import_t *
fmu)

Get FMU generation tool.

Parameters

fmu	An fmu object as returned by fmi2_import_parse_xml() .
-----	--

6.35.2.13 FMILIB_EXPORT const char* fmi2_import_get_generation_date_and_time (fmi2_import_t *
fmu)

Get FMU generation date and time.

Parameters

fmu	An fmu object as returned by fmi2_import_parse_xml() .
-----	--

6.35.2.14 **FMILIB_EXPORT** fmi2_variable_naming_convension_enu_t
fmi2_import_get_naming_convention (fmi2_import_t * fmu)

Get variable naming convention used.

Parameters

<i>fmu</i>	An fmu object as returned by fmi2_import_parse_xml() .
------------	--

6.35.2.15 **FMILIB_EXPORT** size_t fmi2_import_get_number-
_of_continuous_states (fmi2_import_t * fmu
)

Get the number of continuous states.

6.35.2.16 **FMILIB_EXPORT** size_t fmi2_import_get_number_of_event_indicators (fmi2_import_t * fmu)

Get the number of event indicators.

6.35.2.17 **FMILIB_EXPORT** double fmi2_import_get_default_experiment_start (fmi2_import_t * fmu)

Get the start time for default experiment as specified in the XML file.

6.35.2.18 **FMILIB_EXPORT** double fmi2_import_get_default_experiment_stop (fmi2_import_t * fmu)

Get the stop time for default experiment as specified in the XML file.

6.35.2.19 **FMILIB_EXPORT** double fmi2_import_get_default_experiment_tolerance (fmi2_import_t * fmu)

Get the tolerance for default experiment as specified in the XML file.

6.35.2.20 **FMILIB_EXPORT** double fmi2_import_get_default_experiment_step (fmi2_import_t * fmu)

Get the step size for default experiment as specified in the XML file.

6.35.2.21 **FMILIB_EXPORT** fmi2_fmu_kind_enu_t fmi2_import_get_fmu_kind (fmi2_import_t * fmu)

Get the type of the FMU (model exchange or co-simulation)

6.35.2.22 FMILIB_EXPORT fmi2_import_type_definitions_t*
fmi2_import_get_type_definitions (fmi2_import_t *)

Get the list of all the type definitions in the model.

6.35.2.23 FMILIB_EXPORT fmi2_import_unit_definitions_t*
fmi2_import_get_unit_definitions (fmi2_import_t * *fmu*)

Get a list of all the unit definitions in the model.

6.35.2.24 FMILIB_EXPORT fmi2_import_variable_t* fmi2_import_get -
variable_alias_base (fmi2_import_t * *fmu*, fmi2_import_variable_t *
)

Get the variable with the same value reference that is not an alias.

6.35.2.25 FMILIB_EXPORT fmi2_import_variable_list_t* fmi2_import_get -
variable_aliases (fmi2_import_t * *fmu*, fmi2_import_variable_t *
)

Get the list of all the variables aliased to the given one (including the base one).

Note that the list is ordered: base variable, aliases, negated aliases.

6.35.2.26 FMILIB_EXPORT fmi2_import_variable_list_t*
fmi2_import_get_variable_list (fmi2_import_t * *fmu*, int *sortOrder*)

Get the list of all the variables in the model.

Parameters

<i>fmu</i>	An FMU object as returned by fmi2_import_parse_xml() .
<i>sortOrder</i>	Specifies the order of the variables in the list: 0 - original order as found in the XML file; 1 - sorted alphabetically by variable name; 2 sorted by types/value references.

Returns

a variable list with all the variables in the model.

Note that variable lists are allocated dynamically and must be freed when not needed any longer.

6.35.2.27 FMILIB_EXPORT fmi2_import_variable_list_t* fmi2_import_create_var_list (fmi2_import_t * *fmu*, fmi2_import_variable_t * *v*)

Create a variable list with a single variable.

Parameters

<i>fmu</i>	An FMU object that this variable list will reference.
<i>v</i>	A variable.

6.35.2.28 FMILIB_EXPORT size_t fmi2_import_get_vendors_num (fmi2_import_t * *fmu*)

Get the number of vendors that had annotations in the XML.

6.35.2.29 FMILIB_EXPORT const char* fmi2_import_get_vendor_name (fmi2_import_t * *fmu*, size_t *index*)

Get the name of the vendor with that had annotations in the XML by index.

6.35.2.30 FMILIB_EXPORT size_t fmi2_import_get_log_categories_num (fmi2_import_t * *fmu*)

Get the number of log categories defined in the XML.

6.35.2.31 FMILIB_EXPORT const char* fmi2_import_get_log_category (fmi2_import_t * *fmu*, size_t *index*)

Get the log category by index.

6.35.2.32 FMILIB_EXPORT const char* fmi2_import_get_log_category_description (fmi2_import_t * *fmu*, size_t *index*)

Get the log category description by index.

6.35.2.33 **FMILIB_EXPORT** `size_t fmi2_import_get_source_files_me_num ( fmi2_import_t * fmu )`

Get the number of source files for ME defined in the XML.

6.35.2.34 **FMILIB_EXPORT** `const char* fmi2_import_get_source_file_me ( fmi2_import_t * fmu, size_t index )`

Get the ME source file by index.

6.35.2.35 **FMILIB_EXPORT** `size_t fmi2_import_get_source_files_cs_num ( fmi2_import_t * fmu )`

Get the number of source files for CS defined in the XML.

6.35.2.36 **FMILIB_EXPORT** `const char* fmi2_import_get_source_file_cs ( fmi2_import_t * fmu, size_t index )`

Get the CS source file by index.

6.35.2.37 **FMILIB_EXPORT** `fmi2_import_variable_t* fmi2_import_get_variable_by_name ( fmi2_import_t * fmu, const char * name )`

Get variable by variable name.

Parameters

<i>fmu</i>	- An fmu object as returned by fmi2_import_parse_xml() .
<i>name</i>	- variable name

Returns

variable pointer.

6.35.2.38 **FMILIB_EXPORT** `fmi2_import_variable_t* fmi2_import_get_variable_by_vr ( fmi2_import_t * fmu, fmi2_base_type_enumeration_t baseType, fmi2_value_reference_t vr )`

Get variable by value reference.

Parameters

<i>fmu</i>	- An fmu object as returned by fmi2_import_parse_xml() .
<i>baseType</i>	- basic data type
<i>vr</i>	- value reference

Returns

variable pointer.

6.35.2.39 FMILIB_EXPORT fmi2_import_variable_list_t*
fmi2_import_get_outputs_list (fmi2_import_t * fmu)

Get the list of all the output variables in the model.

Parameters

<i>fmu</i>	An FMU object as returned by fmi2_import_parse_xml() .
------------	--

Returns

a variable list with all the output variables in the model.

Note that variable lists are allocated dynamically and must be freed when not needed any longer.

6.35.2.40 FMILIB_EXPORT fmi2_import_variable_list_t*
fmi2_import_get_derivatives_list (fmi2_import_t * fmu)

Get the list of all the derivative variables in the model.

Parameters

<i>fmu</i>	An FMU object as returned by fmi2_import_parse_xml() .
------------	--

Returns

a variable list with all the continuous state derivatives in the model.

Note that variable lists are allocated dynamically and must be freed when not needed any longer.

6.35.2.41 FMILIB_EXPORT fmi2_import_variable_list_t*
fmi2_import_get_discrete_states_list (fmi2_import_t * fmu)

Get the list of all the discrete state variables in the model.

Parameters

<i>fmu</i>	An FMU object as returned by fmi2_import_parse_xml() .
------------	--

Returns

a variable list with all the discrete state variables in the model.

Note that variable lists are allocated dynamically and must be freed when not needed any longer.

6.35.2.42 FMILIB_EXPORT fmi2_import_variable_list_t*
fmi2_import_get_initial_unknowns_list (fmi2_import_t * *fmu*)

Get the list of all the initial unknown variables in the model.

Parameters

<i>fmu</i>	An FMU object as returned by fmi2_import_parse_xml() .
------------	--

Returns

a variable list with all the initial unknowns in the model.

Note that variable lists are allocated dynamically and must be freed when not needed any longer.

6.35.2.43 FMILIB_EXPORT void fmi2_import_get_outputs_dependencies (
fmi2_import_t * *fmu*, size_t ** *startIndex*, size_t ** *dependency*, char **
***factorKind*)**

Get dependency information in row-compressed format.

Parameters

<i>fmu</i>	An FMU object as returned by fmi2_import_parse_xml() .
<i>startIndex</i>	- outputs a pointer to an array of start indices (size of array is number of outputs + 1). First element is zero, last is equal to the number of elements in the dependency and factor arrays. NULL pointer is returned if no dependency information was provided in the XML.
<i>dependency</i>	- outputs a pointer to the dependency index data. Indices are 1-based. Index equals to zero means "depends on all" (no information in the XML).
<i>factorKind</i>	- outputs a pointer to the factor kind data. The values can be converted to fmi2_dependency_factor_kind_enumeration

6.35.2.44 FMILIB_EXPORT void fmi2_import_get_derivatives_dependencies (
fmi2_import_t * *fmu*, size_t ** *startIndex*, size_t ** *dependency*, char **
***factorKind*)**

Get dependency information in row-compressed format.

Parameters

<i>fmU</i>	An FMU object as returned by fmi2_import_parse_xml() .
<i>startIndex</i>	- outputs a pointer to an array of start indices (size of array is number of derivatives + 1). First element is zero, last is equal to the number of elements in the dependency and factor arrays. NULL pointer is returned if no dependency information was provided in the XML.
<i>dependency</i>	- outputs a pointer to the dependency index data. Indices are 1-based. Index equals to zero means "depends on all" (no information in the XML).
<i>factorKind</i>	- outputs a pointer to the factor kind data. The values can be converted to fmi2_dependency_factor_kind_enu_t

6.35.2.45 FMILIB_EXPORT void fmi2_import_get_discrete_states_dependencies (fmi2_import_t * fmU, size_t ** startIndex, size_t ** dependency, char ** factorKind)

Get dependency information in row-compressed format.

Parameters

<i>fmU</i>	An FMU object as returned by fmi2_import_parse_xml() .
<i>startIndex</i>	- outputs a pointer to an array of start indices (size of array is number of discrete states + 1). First element is zero, last is equal to the number of elements in the dependency and factor arrays. NULL pointer is returned if no dependency information was provided in the XML.
<i>dependency</i>	- outputs a pointer to the dependency index data. Indices are 1-based. Index equals to zero means "depends on all" (no information in the XML).
<i>factorKind</i>	- outputs a pointer to the factor kind data. The values can be converted to fmi2_dependency_factor_kind_enu_t

6.35.2.46 FMILIB_EXPORT void fmi2_import_get_initial_unknowns_dependencies (fmi2_import_t * fmU, size_t ** startIndex, size_t ** dependency, char ** factorKind)

Get dependency information in row-compressed format.

Parameters

<i>fmU</i>	An FMU object as returned by fmi2_import_parse_xml() .
<i>startIndex</i>	- outputs a pointer to an array of start indices (size of array is number of initial unknowns + 1). First element is zero, last is equal to the number of elements in the dependency and factor arrays. NULL pointer is returned if no dependency information was provided in the XML.
<i>dependency</i>	- outputs a pointer to the dependency index data. Indices are 1-based. Index equals to zero means "depends on all" (no information in the XML).

<i>factorKind</i>	- outputs a pointer to the factor kind data. The values can be converted to fmi2_dependency_factor_kind_enumeration
-------------------	---

6.36 Interface to the standard FMI 2.0 "C" API

Convenient functions for calling the FMI functions. This interface wraps the "C" API.

Modules

- [FMI 2.0 Constructor and Destructor](#)
Functions for instantiating and freeing the container of the struct that is responsible for the FMI functions.
- [FMI 2.0 \(ME\) Model Exchange functions](#)
List of Model Exchange wrapper functions. Common functions are not listed.
- [FMI 2.0 \(CS\) Co-Simulation functions](#)
List of Co-Simulation wrapper functions. Common functions are not listed.
- [FMI 2.0 \(ME & CS\) Common functions](#)
List of wrapper functions that are in common for both Model Exchange and Co-Simulation.

6.36.1 Detailed Description

Convenient functions for calling the FMI functions. This interface wraps the "C" API.

6.37 FMI 2.0 Constructor and Destructor

Functions for instantiating and freeing the container of the struct that is responsible for the FMI functions.

Functions

- **FMLIB_EXPORT** `jm_status_enu_t fmi2_import_create_dllfmu (fmi2_import_t *fmu, fmi2_fmu_kind_enu_t fmuKind, const fmi2_callback_functions_t *callBackFunctions)`

Create a C-API struct. The C-API struct is a placeholder for the FMI DLL functions.

- **FMLIB_EXPORT** `void fmi2_import_destroy_dllfmu (fmi2_import_t *fmu)`

Free a C-API struct. All memory allocated since the struct was created is freed.

- **FMLIB_EXPORT** `void fmi2_import_set_debug_mode (fmi2_import_t *fmu, int mode)`

Set CAPI debug mode flag. Setting to non-zero prevents DLL unloading in fmi2_import_destroy_dllfmu while all the memory is deallocated. This is to support valgrind debugging.

6.37.1 Detailed Description

Functions for instantiating and freeing the container of the struct that is responsible for the FMI functions. Before any of the FMI functions may be called, the construction function must instantiate a `fmi_import_t` module. After the `fmi_import_t` module has been successfully instantiated, all the FMI functions can be called. To unload the FMI functions, the destroy functions shall be called.

6.37.2 Function Documentation

- 6.37.2.1 FMLIB_EXPORT jm_status_enu_t fmi2_import_create_dllfmu**
(`fmi2_import_t * fmu`, `fmi2_fmu_kind_enu_t fmuKind`, `const fmi2_callback_functions_t * callBackFunctions`)

Create a C-API struct. The C-API struct is a placeholder for the FMI DLL functions.

This function may only be called once if it returned successfully. `fmi2_import_destroy_dllfmu` must be called before this function can be called again.

Parameters

<i>fmu</i>	A model description object returned by <code>fmi2_import_parse_xml()</code> .
<i>fmuKind</i>	Specifies if ModelExchange or CoSimulation binary should be loaded.
<i>callBack-Functions</i>	Callback functions to be used by the FMI functions internally. If this parameter is NULL then the <code>jm_callbacks::</code> and <code>fmi2_log_forwarding</code> are utilized to fill in the default structure.

Returns

Error status. If the function returns with an error, it is not allowed to call any of the other C-API functions.

6.37.2.2 FMILIB_EXPORT void fmi2_import_destroy_dllfmu (fmi2_import_t * *fmu*)

Free a C-API struct. All memory allocated since the struct was created is freed.

Parameters

<i>fmu</i>	A model description object returned from fmi2_import_parse_xml() .
------------	--

6.37.2.3 FMILIB_EXPORT void fmi2_import_set_debug_mode (fmi2_import_t * *fmu*, int *mode*)

Set CAPI debug mode flag. Setting to non-zero prevents DLL unloading in fmi2_import_destroy_dllfmu while all the memory is deallocated. This is to support valgrind debugging.

Parameters

<i>fmu</i>	C-API struct that has successfully loaded the FMI function.
<i>mode</i>	The debug mode to set.

6.38 FMI 2.0 (ME) Model Exchange functions

List of Model Exchange wrapper functions. Common functions are not listed.

Functions

- `FMILIB_EXPORT fmi2_status_t fmi2_import_enter_event_mode (fmi2_import_t *fmu)`
Calls the FMI function `fmiEnterEventMode(...)`
- `FMILIB_EXPORT fmi2_status_t fmi2_import_new_discrete_states (fmi2_import_t *fmu, fmi2_event_info_t *eventInfo)`
Calls the FMI function `fmiNewDiscreteStates(...)`
- `FMILIB_EXPORT fmi2_status_t fmi2_import_enter_continuous_time_mode (fmi2_import_t *fmu)`
Calls the FMI function `fmiEnterContinuousTimeMode(...)`
- `FMILIB_EXPORT fmi2_status_t fmi2_import_set_time (fmi2_import_t *fmu, fmi2_real_t time)`
Wrapper for the FMI function `fmiSetTime(...)`
- `FMILIB_EXPORT fmi2_status_t fmi2_import_set_continuous_states (fmi2_import_t *fmu, const fmi2_real_t x[], size_t nx)`
Wrapper for the FMI function `fmiSetContinuousStates(...)`
- `FMILIB_EXPORT fmi2_status_t fmi2_import_completed_integrator_step (fmi2_import_t *fmu, fmi2_boolean_t noSetFMUStatePriorToCurrentPoint, fmi2_boolean_t *enterEventMode, fmi2_boolean_t *terminateSimulation)`
Wrapper for the FMI function `fmiCompletedIntegratorStep(...)`
- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_derivatives (fmi2_import_t *fmu, fmi2_real_t derivatives[], size_t nx)`
Wrapper for the FMI function `fmiGetDerivatives(...)`
- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_event_indicators (fmi2_import_t *fmu, fmi2_real_t eventIndicators[], size_t ni)`
Wrapper for the FMI function `fmiGetEventIndicators(...)`
- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_continuous_states (fmi2_import_t *fmu, fmi2_real_t states[], size_t nx)`
Wrapper for the FMI function `fmiGetContinuousStates(...)`
- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_nominals_of_continuous_states (fmi2_import_t *fmu, fmi2_real_t x_nominal[], size_t nx)`
Wrapper for the FMI function `fmiGetNominalsOfContinuousStates(...)`

6.38.1 Detailed Description

List of Model Exchange wrapper functions. Common functions are not listed.

6.38.2 Function Documentation

6.38.2.1 FMILIB_EXPORT fmi2_status_t fmi2_import_enter_event_mode (fmi2_import_t * *fmu*)

Calls the FMI function fmiEnterEventMode(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
------------	---

Returns

FMI status.

6.38.2.2 FMILIB_EXPORT fmi2_status_t fmi2_import_new_discrete_states (fmi2_import_t * *fmu*, fmi2_event_info_t * *eventInfo*)

Calls the FMI function fmiNewDiscreteStates(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>eventInfo</i>	Pointer to fmi2_event_info_t structure that will be filled in.

Returns

FMI status.

6.38.2.3 FMILIB_EXPORT fmi2_status_t fmi2_import_enter-_continuous_time_mode (fmi2_import_t * *fmu*)

Calls the FMI function fmiEnterContinuousTimeMode(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
------------	---

Returns

FMI status.

**6.38.2.4 FMILIB_EXPORT fmi2_status_t fmi2_import_set_time (fmi2_import_t *
fmu, fmi2_real_t time)**

Wrapper for the FMI function fmiSetTime(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>time</i>	Set the current time.

Returns

FMI status.

**6.38.2.5 FMILIB_EXPORT fmi2_status_t fmi2_import_set_continuous_states (fmi2_import_t *
fmu, const fmi2_real_t x[], size_t nx)**

Wrapper for the FMI function fmiSetContinuousStates(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>x</i>	Array of state values.
<i>nx</i>	Number of states.

Returns

FMI status.

**6.38.2.6 FMILIB_EXPORT fmi2_status_t fmi2_import_completed_integrator_step (fmi2_import_t *
fmu, fmi2_boolean_t noSetFMUStatePriorToCurrentPoint, fmi2_boolean_t *
enterEventMode, fmi2_boolean_t * terminateSimulation)**

Wrapper for the FMI function fmiCompletedIntegratorStep(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>noSetFMU-StatePrior-ToCurrent-Point</i>	True if fmiSetFMUState will no longer be called for time instants prior to current time in this simulation run.
<i>enterEvent-Mode</i>	(Output) Call fmiEnterEventMode indicator.
<i>terminate-Simulation</i>	(Output) Terminate simulation indicator.

Returns

FMI status.

6.38.2.7 FMILIB_EXPORT fmi2_status_t fmi2_import_get_derivatives (fmi2_import_t * fmu, fmi2_real_t derivatives[], size_t nx)

Wrapper for the FMI function fmiGetDerivatives(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>derivatives</i>	(Output) Array of the derivatives.
<i>nx</i>	Number of derivatives.

Returns

FMI status.

6.38.2.8 FMILIB_EXPORT fmi2_status_t fmi2_import_get_event_indicators (fmi2_import_t * fmu, fmi2_real_t eventIndicators[], size_t ni)

Wrapper for the FMI function fmiGetEventIndicators(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>event-Indicators</i>	(Output) The event indicators.
<i>ni</i>	Number of event indicators.

Returns

FMI status.

6.38.2.9 FMILIB_EXPORT fmi2_status_t fmi2_import_get_continuous_states (fmi2_import_t * fmu, fmi2_real_t states[], size_t nx)

Wrapper for the FMI function fmiGetContinuousStates(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>states</i>	(Output) Array of state values.
<i>nx</i>	Number of states.

Returns

FMI status.

6.38.2.10 **FMILIB_EXPORT** `fmi2_status_t fmi2_import_get_nominals_of_continuous_states ( fmi2_import_t * fmu, fmi2_real_t x_nominal[], size_t nx )`

Wrapper for the FMI function `fmiGetNominalsOfContinuousStates(...)`

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>x_nominal</i>	(Output) The nominal values.
<i>nx</i>	Number of nominal values.

Returns

FMI status.

6.39 FMI 2.0 (CS) Co-Simulation functions

List of Co-Simulation wrapper functions. Common functions are not listed.

Functions

- `FMILIB_EXPORT fmi2_status_t fmi2_import_set_real_input_derivatives (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_integer_t order[], const fmi2_real_t value[])`

Wrapper for the FMI function fmiSetRealInputDerivatives(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_real_output_derivatives (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_integer_t order[], fmi2_real_t value[])`

Wrapper for the FMI function fmiGetOutputDerivatives(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_cancel_step (fmi2_import_t *fmu)`

Wrapper for the FMI function fmiCancelStep(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_do_step (fmi2_import_t *fmu, fmi2_real_t currentCommunicationPoint, fmi2_real_t communicationStepSize, fmi2_boolean_t newStep)`

Wrapper for the FMI function fmiDoStep(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_status_t *value)`

Wrapper for the FMI function fmiGetStatus(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_real_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_real_t *value)`

Wrapper for the FMI function fmiGetRealStatus(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_integer_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_integer_t *value)`

Wrapper for the FMI function fmiGetIntegerStatus(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_boolean_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_boolean_t *value)`

Wrapper for the FMI function fmiGetBooleanStatus(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_string_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_string_t *value)`

Wrapper for the FMI function fmiGetStringStatus(...)

6.39.1 Detailed Description

List of Co-Simulation wrapper functions. Common functions are not listed.

6.39.2 Function Documentation

6.39.2.1 **FMILIB_EXPORT** fmi2_status_t fmi2_import_set_real_input_derivatives (fmi2_import_t * *fmu*, const fmi2_value_reference_t *vr*[], size_t *nvr*, const fmi2_integer_t *order*[], const fmi2_real_t *value*[])

Wrapper for the FMI function fmiSetRealInputDerivatives(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>order</i>	Array of derivative orders.
<i>value</i>	Array of variable values.

Returns

FMI status.

6.39.2.2 **FMILIB_EXPORT** fmi2_status_t fmi2_import_get_real_output_derivatives (fmi2_import_t * *fmu*, const fmi2_value_reference_t *vr*[], size_t *nvr*, const fmi2_integer_t *order*[], fmi2_real_t *value*[])

Wrapper for the FMI function fmiGetOutputDerivatives(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>order</i>	Array of derivative orders.
<i>value</i>	(Output) Array of variable values.

Returns

FMI status.

6.39.2.3 **FMILIB_EXPORT** fmi2_status_t fmi2_import_cancel_step (fmi2_import_t * *fmu*)

Wrapper for the FMI function fmiCancelStep(...)

Parameters

<i>fm</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
-----------	---

Returns

FMI status.

6.39.2.4 FMILIB_EXPORT fmi2_status_t fmi2_import_do_step (fmi2_import_t * *fm*, fmi2_real_t *currentCommunicationPoint*, fmi2_real_t *communicationStepSize*, fmi2_boolean_t *newStep*)

Wrapper for the FMI function fmiDoStep(...)

Parameters

<i>fm</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>current-Communication-Point</i>	Current communication point of the master.
<i>communication-StepSize</i>	Communication step size.
<i>newStep</i>	Indicates whether or not the last communication step was accepted by the master.

Returns

FMI status.

6.39.2.5 FMILIB_EXPORT fmi2_status_t fmi2_import_get_status (fmi2_import_t * *fm*, const fmi2_status_kind_t *s*, fmi2_status_t * *value*)

Wrapper for the FMI function fmiGetStatus(...)

Parameters

<i>fm</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI status value.

Returns

FMI status.

6.39.2.6 FMILIB_EXPORT fmi2_status_t fmi2_import_get_real_status (
fmi2_import_t * *fmu*, const fmi2_status_kind_t *s*, fmi2_real_t * *value*)

Wrapper for the FMI function fmiGetRealStatus(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI real value.

Returns

FMI status.

6.39.2.7 FMILIB_EXPORT fmi2_status_t fmi2_import_get_integer_status (
fmi2_import_t * *fmu*, const fmi2_status_kind_t *s*, fmi2_integer_t * *value*)

Wrapper for the FMI function fmiGetIntegerStatus(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI integer value.

Returns

FMI status.

6.39.2.8 FMILIB_EXPORT fmi2_status_t fmi2_import_get_boolean_status (
fmi2_import_t * *fmu*, const fmi2_status_kind_t *s*, fmi2_boolean_t * *value*)

Wrapper for the FMI function fmiGetBooleanStatus(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI boolean value.

Returns

FMI status.

6.39.2.9 FMILIB_EXPORT fmi2_status_t fmi2_import_get_string_status (
fmi2_import_t * *fmu*, const fmi2_status_kind_t *s*, fmi2_string_t * *value*)

Wrapper for the FMI function fmiGetStringStatus(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	Kind of status to return the value for.
<i>value</i>	(Output) FMI string value.

Returns

FMI status.

6.40 FMI 2.0 (ME & CS) Common functions

List of wrapper functions that are in common for both Model Exchange and Co-Simulation.

Functions

- **FMILIB_EXPORT** `const char * fmi2_import_get_version (fmi2_import_t *fmu)`
Wrapper for the FMI function `fmiGetVersion()`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_set_debug_logging (fmi2_import_t *fmu, fmi2_boolean_t loggingOn, size_t nCategories, fmi2_string_t categories[])`
Wrapper for the FMI function `fmiSetDebugLogging(...)`
- **FMILIB_EXPORT** `jm_status_enu_t fmi2_import_instantiate (fmi2_import_t *fmu, fmi2_string_t instanceName, fmi2_type_t fmuType, fmi2_string_t fmuResourceLocation, fmi2_boolean_t visible)`
Wrapper for the FMI function `fmiInstantiate(...)`
- **FMILIB_EXPORT** `void fmi2_import_free_instance (fmi2_import_t *fmu)`
Wrapper for the FMI function `fmiFreeInstance(...)`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_setup_experiment (fmi2_import_t *fmu, fmi2_boolean_t toleranceDefined, fmi2_real_t tolerance, fmi2_real_t startTime, fmi2_boolean_t stopTimeDefined, fmi2_real_t stopTime)`
Calls the FMI function `fmiSetupExperiment(...)`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_enter_initialization_mode (fmi2_import_t *fmu)`
Calls the FMI function `fmiEnterInitializationMode(...)`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_exit_initialization_mode (fmi2_import_t *fmu)`
Calls the FMI function `fmiExitInitializationMode(...)`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_terminate (fmi2_import_t *fmu)`
Wrapper for the FMI function `fmiTerminate(...)`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_reset (fmi2_import_t *fmu)`
Wrapper for the FMI function `fmiReset(...)`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_set_real (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_real_t value[])`
Wrapper for the FMI function `fmiSetReal(...)`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_set_integer (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_integer_t value[])`
Wrapper for the FMI function `fmiSetInteger(...)`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_set_boolean (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_boolean_t value[])`
Wrapper for the FMI function `fmiSetBoolean(...)`
- **FMILIB_EXPORT** `fmi2_status_t fmi2_import_set_string (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_string_t value[])`

Wrapper for the FMI function fmiSetString(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_real (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_real_t value[])`

Wrapper for the FMI function fmiGetReal(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_integer (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_integer_t value[])`

Wrapper for the FMI function fmiGetInteger(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_boolean (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_boolean_t value[])`

Wrapper for the FMI function fmiGetBoolean(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_string (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_string_t value[])`

Wrapper for the FMI function fmiGetString(...)

- `FMILIB_EXPORT const char * fmi2_import_get_types_platform (fmi2_import_t *fmu)`

Wrapper for the FMI function fmiGetTypesPlatform(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_fmu_state (fmi2_import_t *fmu, fmi2_FMU_state_t *s)`

Wrapper for the FMI function fmiGetFMUstate(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_set_fmu_state (fmi2_import_t *fmu, fmi2_FMU_state_t s)`

Wrapper for the FMI function fmiSetFMUstate(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_free_fmu_state (fmi2_import_t *fmu, fmi2_FMU_state_t *s)`

Wrapper for the FMI function fmiFreeFMUstate(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_serialized_fmu_state_size (fmi2_import_t *fmu, fmi2_FMU_state_t s, size_t *sz)`

Wrapper for the FMI function fmiSerializedFMUstateSize(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_serialize_fmu_state (fmi2_import_t *fmu, fmi2_FMU_state_t s, fmi2_byte_t data[], size_t sz)`

Wrapper for the FMI function fmiSerializeFMUstate(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_de_serialize_fmu_state (fmi2_import_t *fmu, const fmi2_byte_t data[], size_t sz, fmi2_FMU_state_t *s)`

Wrapper for the FMI function fmiSerializeFMUstate(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_directional_derivative (fmi2_import_t *fmu, const fmi2_value_reference_t v_ref[], size_t nv, const fmi2_value_reference_t z_ref[], size_t nz, const fmi2_real_t dv[], fmi2_real_t dz[])`

Wrapper for the FMI function fmiGetDirectionalDerivative(...)

6.40.1 Detailed Description

List of wrapper functions that are in common for both Model Exchange and Co-Simulation.

6.40.2 Function Documentation

6.40.2.1 FMILIB_EXPORT `const char* fmi2_import_get_version ( fmi2_import_t * fmu )`

Wrapper for the FMI function [fmiGetVersion\(\)](#)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
------------	---

Returns

FMI version.

6.40.2.2 FMILIB_EXPORT `fmi2_status_t fmi2_import_set_debug_logging ( fmi2_import_t * fmu, fmi2_boolean_t loggingOn, size_t nCategories, fmi2_string_t categories[] )`

Wrapper for the FMI function [fmiSetDebugLogging\(...\)](#)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>loggingOn</i>	Enable or disable the debug logger.
<i>nCategories</i>	Number of categories to log.
<i>categories</i>	Which categories to log.

Returns

FMI status.

6.40.2.3 FMILIB_EXPORT `jm_status_enu_t fmi2_import_instantiate ( fmi2_import_t * fmu, fmi2_string_t instanceName, fmi2_type_t fmuType, fmi2_string_t fmuResourceLocation, fmi2_boolean_t visible )`

Wrapper for the FMI function [fmiInstantiate\(...\)](#)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>instance-Name</i>	The name of the instance.
<i>fmuType</i>	<code>fmi2_model_exchange</code> or <code>fmi2_cosimulation</code> .

<i>fmu-Resource-Location</i>	Access path URI to the FMU archive resources. If this is NULL pointer the FMU will get the path to the unzipped location.
<i>visible</i>	Indicates whether or not the simulator application window should be visible.

Returns

Error status. Returns `jm_status_error` if `fmiInstantiate` returned NULL, otherwise `jm_status_success`.

6.40.2.4 FMILIB_EXPORT void fmi2_import_free_instance (fmi2_import_t * fmu)

Wrapper for the FMI function `fmiFreeInstance(...)`

Parameters

<i>fmu</i>	An fmu description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
------------	--

6.40.2.5 FMILIB_EXPORT fmi2_status_t fmi2_import_setup_experiment (fmi2_import_t * fmu, fmi2_boolean_t toleranceDefined, fmi2_real_t tolerance, fmi2_real_t startTime, fmi2_boolean_t stopTimeDefined, fmi2_real_t stopTime)

Calls the FMI function `fmiSetupExperiment(...)`

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>tolerance_defined</i>	True if the <code>tolerance</code> argument is to be used
<i>tolerance</i>	Solvers internal to the FMU should use this tolerance or finer, if <code>tolerance_defined</code> is true
<i>start_time</i>	Start time of the experiment
<i>stop_time_defined</i>	True if the <code>stop_time</code> argument is to be used
<i>stop_time</i>	Stop time of the experiment, if <code>stop_time_defined</code> is true

Returns

FMI status.

6.40.2.6 FMILIB_EXPORT fmi2_status_t fmi2_import_enter_initialization_mode (fmi2_import_t * *fmu*)

Calls the FMI function fmiEnterInitializationMode(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
------------	---

Returns

FMI status.

6.40.2.7 FMILIB_EXPORT fmi2_status_t fmi2_import_exit_initialization_mode (fmi2_import_t * *fmu*)

Calls the FMI function fmiExitInitializationMode(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
------------	---

Returns

FMI status.

6.40.2.8 FMILIB_EXPORT fmi2_status_t fmi2_import_terminate (fmi2_import_t * *fmu*)

Wrapper for the FMI function fmiTerminate(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
------------	---

Returns

FMI status.

6.40.2.9 FMILIB_EXPORT fmi2_status_t fmi2_import_reset (fmi2_import_t * *fmu*)

Wrapper for the FMI function fmiReset(...)

Parameters

<i>fm</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
-----------	---

Returns

FMI status.

6.40.2.10 FMILIB_EXPORT fmi2_status_t fmi2_import_set_real (fmi2_import_t * *fm*, const fmi2_value_reference_t *vr*[], size_t *nvr*, const fmi2_real_t *value*[])

Wrapper for the FMI function fmiSetReal(...)

Parameters

<i>fm</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	Array of variable values.

Returns

FMI status.

6.40.2.11 FMILIB_EXPORT fmi2_status_t fmi2_import_set_integer (fmi2_import_t * *fm*, const fmi2_value_reference_t *vr*[], size_t *nvr*, const fmi2_integer_t *value*[])

Wrapper for the FMI function fmiSetInteger(...)

Parameters

<i>fm</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	Array of variable values.

Returns

FMI status.

6.40.2.12 FMILIB_EXPORT fmi2_status_t fmi2_import_set_boolean (
fmi2_import_t * *fmu*, const fmi2_value_reference_t *vr*[], size_t *nvr*, const
fmi2_boolean_t *value*[])

Wrapper for the FMI function fmiSetBoolean(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	Array of variable values.

Returns

FMI status.

6.40.2.13 FMILIB_EXPORT fmi2_status_t fmi2_import_set_string (fmi2_import_t
*** *fmu*, const fmi2_value_reference_t *vr*[], size_t *nvr*, const fmi2_string_t *value*[])**

Wrapper for the FMI function fmiSetString(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	Array of variable values.

Returns

FMI status.

6.40.2.14 FMILIB_EXPORT fmi2_status_t fmi2_import_get_real (fmi2_import_t *
***fmu*, const fmi2_value_reference_t *vr*[], size_t *nvr*, fmi2_real_t *value*[])**

Wrapper for the FMI function fmiGetReal(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	(Output)Array of variable values.

Returns

FMI status.

```
6.40.2.15  FMILIB_EXPORT fmi2_status_t fmi2_import_get_integer (
    fmi2_import_t * fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_integer_t
    value[] )
```

Wrapper for the FMI function `fmiGetInteger(...)`

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	(Output)Array of variable values.

Returns

FMI status.

```
6.40.2.16  FMILIB_EXPORT fmi2_status_t fmi2_import_get_boolean (
    fmi2_import_t * fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_boolean_t
    value[] )
```

Wrapper for the FMI function `fmiGetBoolean(...)`

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	(Output)Array of variable values.

Returns

FMI status.

6.40.2.17 **FMLIB_EXPORT** `fmi2_status_t fmi2_import_get_string ( fmi2_import_t * fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_string_t value[] )`

Wrapper for the FMI function `fmiGetString(...)`

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>vr</i>	Array of value references.
<i>nvr</i>	Number of array elements.
<i>value</i>	(Output)Array of variable values.

Returns

FMI status.

6.40.2.18 **FMLIB_EXPORT** `const char* fmi2_import_get_types_platform ( fmi2_import_t * fmu )`

Wrapper for the FMI function `fmiGetTypesPlatform(...)`

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
------------	---

Returns

The platform the FMU was compiled for.

6.40.2.19 **FMLIB_EXPORT** `fmi2_status_t fmi2_import_get_fmu_state ( fmi2_import_t * fmu, fmi2_FMU_state_t * s )`

Wrapper for the FMI function `fmiGetFMUstate(...)`

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	The state object to be set by the FMU

Returns

FMI status.

6.40.2.20 **FMILIB_EXPORT fmi2_status_t fmi2_import_set_fmu_state (**
fmi2_import_t * fmu, fmi2_FMU_state_t s)

Wrapper for the FMI function fmiSetFMUstate(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	The FMU state object

Returns

FMI status.

6.40.2.21 **FMILIB_EXPORT fmi2_status_t fmi2_import_free_fmu_state (**
fmi2_import_t * fmu, fmi2_FMU_state_t * s)

Wrapper for the FMI function fmiFreeFMUstate(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	The FMU state object

Returns

FMI status.

6.40.2.22 **FMILIB_EXPORT fmi2_status_t fmi2_import_serialized_fmu_state_size**
(fmi2_import_t * fmu, fmi2_FMU_state_t s, size_t * sz)

Wrapper for the FMI function fmiSerializedFMUstateSize(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	The FMU state object
<i>sz</i>	The size of the serialized state in bytes

Returns

FMI status.

6.40.2.23 **FMLIB_EXPORT** fmi2_status_t fmi2_import_serialize_fmu_state (fmi2_import_t * *fmu*, fmi2_FMU_state_t *s*, fmi2_byte_t *data*[], size_t *sz*)

Wrapper for the FMI function fmiSerializeFMUstate(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>s</i>	The FMU state object
<i>data</i>	The buffer that will receive serialized FMU state
<i>sz</i>	The size of the data buffer

Returns

FMI status.

6.40.2.24 **FMLIB_EXPORT** fmi2_status_t fmi2_import_de_serialize_fmu_state (fmi2_import_t * *fmu*, const fmi2_byte_t *data*[], size_t *sz*, fmi2_FMU_state_t * *s*)

Wrapper for the FMI function fmiSerializeFMUstate(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>data</i>	The buffer that contains serialized FMU state
<i>sz</i>	The size of the data buffer
<i>s</i>	The FMU state object to be created

Returns

FMI status.

6.40.2.25 **FMLIB_EXPORT** fmi2_status_t fmi2_import_get_directional_derivative (fmi2_import_t * *fmu*, const fmi2_value_reference_t *v_ref*[], size_t *nv*, const fmi2_value_reference_t *z_ref*[], size_t *nz*, const fmi2_real_t *dv*[], fmi2_real_t *dz*[])

Wrapper for the FMI function fmiGetDirectionalDerivative(...)

Parameters

<i>fmu</i>	A model description object returned by fmi2_import_parse_xml() that has loaded the FMI functions, see fmi2_import_create_dllfmu() .
<i>v_ref</i>	Value references for the seed vector
<i>nv</i>	size of <i>v_ref</i> array
<i>z_ref</i>	Value references for the derivatives/outputs to be processed

<i>nz</i>	Size of <i>z_ref</i> array
<i>dv</i>	The seed vector.
<i>dz</i>	Calculated directional derivative on output.

Returns

FMI status.

6.41 Convenience functions.

The functions in this module are provided for convenience. The functionality is already available via other lower level functions.

Data Structures

- struct [fmi2_import_model_counts_t](#)
Collection of counters providing model information.

Functions

- **FMILIB_EXPORT** void [fmi2_import_collect_model_counts](#) ([fmi2_import_t](#) *fmu, [fmi2_import_model_counts_t](#) *counts)
Collect model information by counting the number of variables with specific properties and fillin in [fmi2_import_model_counts_t](#) struct.
- **FMILIB_EXPORT** void [fmi2_import_expand_variable_references](#) ([fmi2_import_t](#) *fmu, const char *msgIn, char *msgOut, [size_t](#) maxMsgSize)
Print msgIn into msgOut by expanding variable references of the form #<Type><V-R># into variable names and replacing '##' with a single #.
- **FMILIB_EXPORT** void [fmi2_log_forwarding](#) ([fmi2_component_t](#) c, [fmi2_string_t](#) instanceName, [fmi2_status_t](#) status, [fmi2_string_t](#) category, [fmi2_string_t](#) message,...)
An implementation of FMI 2.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.
- **FMILIB_EXPORT** void [fmi2_log_forwarding_v](#) ([fmi2_component_t](#) c, [fmi2_string_t](#) instanceName, [fmi2_status_t](#) status, [fmi2_string_t](#) category, [fmi2_string_t](#) message, va_list args)
An implementation of FMI 2.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.
- **FMILIB_EXPORT** void [fmi2_default_callback_logger](#) ([fmi2_component_t](#) c, [fmi2_string_t](#) instanceName, [fmi2_status_t](#) status, [fmi2_string_t](#) category, [fmi2_string_t](#) message,...)
Default FMI 2.0 logger may be used when instantiating FMUs.
- **FMILIB_EXPORT** void [fmi2_import_init_logger](#) ([jm_callbacks](#) *cb, [fmi2_callback_functions_t](#) *fmiCallbacks)
Given [fmi2_callback_functions_t](#) logger ([fmi2_logger](#)), the [jm_callbacks](#) logger may be setup to redirect the messages to the [fmi2_logger](#).

6.41.1 Detailed Description

The functions in this module are provided for convenience. The functionality is already available via other lower level functions.

6.41.2 Function Documentation

6.41.2.1 FMILIB_EXPORT void fmi2_import_collect_model_counts (
fmi2_import_t * fmu, fmi2_import_model_counts_t * counts)

Collect model information by counting the number of variables with specific properties and fillin in [fmi2_import_model_counts_t](#) struct.

Parameters

<i>fmu</i>	- An fmu object as returned by fmi2_import_parse_xml() .
<i>counts</i>	- a pointer to a preallocated struct.

6.41.2.2 FMILIB_EXPORT void fmi2_import_expand_variable_references (
fmi2_import_t * fmu, const char * msgIn, char * msgOut, size_t maxMsgSize)

Print msgIn into msgOut by expanding variable references of the form #<Type><V-R># into variable names and replacing '##' with a single #.

Parameters

<i>fmu</i>	- An fmu object as returned by fmi2_import_parse_xml() .
<i>msgIn</i>	- Log message as produced by an FMU.
<i>msgOut</i>	- Output message buffer.
<i>maxMsgSize</i>	- maximum message size

6.41.2.3 FMILIB_EXPORT void fmi2_log_forwarding (fmi2_component_t c, fmi2_string_t instanceName, fmi2_status_t status, fmi2_string_t category, fmi2_string_t message, ...)

An implementation of FMI 2.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.

The function is using a global array of active FMUs to find out which FMU is sending the log message. It then forwards the message to the logger connected to the particular [fmi2_import_t](#) struct. The function is called by the FMU. The FMU must be loaded with non-zero registerGlobally parameter of [fmi2_import_create_dllfmu\(\)](#) in order to work. If no matching [fmi2_import_t](#) struct is found on the global list then [jm_get_default_callbacks\(\)](#) is used to get the default logger. Note that this function is not thread safe due to the use of the global list.

6.41.2.4 FMILIB_EXPORT void fmi2_log_forwarding_v (fmi2_component_t c, fmi2_string_t instanceName, fmi2_status_t status, fmi2_string_t category, fmi2_string_t message, va_list args)

An implementation of FMI 2.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.

See [fmi2_log_forwarding\(\)](#) for more information.

6.41.2.5 **FMILIB_EXPORT** void **fmi2_default_callback_logger** (**fmi2_component_t** *c*, **fmi2_string_t** *instanceName*, **fmi2_status_t** *status*, **fmi2_string_t** *category*, **fmi2_string_t** *message*, ...)

Default FMI 2.0 logger may be used when instantiating FMUs.

6.41.2.6 **FMILIB_EXPORT** void **fmi2_import_init_logger** (**jm_callbacks** * *cb*, **fmi2_callback_functions_t** * *fmiCallbacks*)

Given [fmi2_callback_functions_t](#) logger (**fmi2_logger**), the [jm_callbacks](#) logger may be setup to redirect the messages to the **fmi2_logger**.

The functions sets up the redirection. Note that the context field in [jm_callbacks](#) is set to point to the provided [fmi2_callback_functions_t](#).

Parameters

<i>cb</i>	FMI Library callbacks
<i>fmiCallbacks</i>	FMI 2.0 standard callbacks

6.42 Support for processing variable types

Functions

- `FMILIB_EXPORT unsigned int fmi2_import_get_type_definition_number (fmi2_import_type_definitions_t *td)`
Get the number of available type definitions.
- `FMILIB_EXPORT fmi2_import_variable_typedef_t * fmi2_import_get_typedef (fmi2_import_type_definitions_t *td, unsigned int index)`
Get a type definition specified by the index.
- `FMILIB_EXPORT fmi2_import_display_unit_t * fmi2_import_get_type_display_unit (fmi2_import_real_typedef_t *)`
Get associated display unit for a type definition if any.
- `FMILIB_EXPORT const char * fmi2_import_get_type_name (fmi2_import_variable_typedef_t *)`
Get the type name.
- `FMILIB_EXPORT const char * fmi2_import_get_type_description (fmi2_import_variable_typedef_t *)`
Get type description.
- `FMILIB_EXPORT fmi2_base_type_enumeration_t fmi2_import_get_base_type (fmi2_import_variable_typedef_t *)`
Get base type used for the type definition.
- `FMILIB_EXPORT fmi2_import_real_typedef_t * fmi2_import_get_type_as_real (fmi2_import_variable_typedef_t *)`
Cast the general type definition object to an object with a specific base type.
- `FMILIB_EXPORT fmi2_import_integer_typedef_t * fmi2_import_get_type_as_int (fmi2_import_variable_typedef_t *)`
Cast the general type definition object to an object with a specific base type.
- `FMILIB_EXPORT fmi2_import_enumeration_typedef_t * fmi2_import_get_type_as_enum (fmi2_import_variable_typedef_t *)`
Cast the general type definition object to an object with a specific base type.
- `FMILIB_EXPORT const char * fmi2_import_get_type_quantity (fmi2_import_variable_typedef_t *)`
Get the quantity associated with the type definition.
- `FMILIB_EXPORT double fmi2_import_get_real_type_min (fmi2_import_real_typedef_t *)`
Get minimal value for the type.
- `FMILIB_EXPORT double fmi2_import_get_real_type_max (fmi2_import_real_typedef_t *)`
Get maximum value for the type.
- `FMILIB_EXPORT double fmi2_import_get_real_type_nominal (fmi2_import_real_typedef_t *)`
Get the nominal value associated with the type definition.
- `FMILIB_EXPORT fmi2_import_unit_t * fmi2_import_get_real_type_unit (fmi2_import_real_typedef_t *)`

Get the unit object associated with the type definition if any.

- `FMILIB_EXPORT int fmi2_import_get_real_type_is_relative_quantity (fmi2_import_real_typedef_t *)`

Get the 'relativeQuantity' flag.

- `FMILIB_EXPORT int fmi2_import_get_real_type_is_unbounded (fmi2_import_real_typedef_t *)`

Get the 'unbounded' flag.

- `FMILIB_EXPORT int fmi2_import_get_integer_type_min (fmi2_import_integer_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT int fmi2_import_get_integer_type_max (fmi2_import_integer_typedef_t *)`

Get maximum value for the type.

- `FMILIB_EXPORT unsigned int fmi2_import_get_enum_type_min (fmi2_import_enumeration_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT unsigned int fmi2_import_get_enum_type_max (fmi2_import_enumeration_typedef_t *)`

Get maximum value for the type.

- `FMILIB_EXPORT unsigned int fmi2_import_get_enum_type_size (fmi2_import_enumeration_typedef_t *)`

Get the number of elements in the enum.

- `FMILIB_EXPORT const char * fmi2_import_get_enum_type_item_name (fmi2_import_enumeration_typedef_t *, unsigned int item)`

Get an enumeration item name by index.

- `FMILIB_EXPORT int fmi2_import_get_enum_type_item_value (fmi2_import_enumeration_typedef_t *, unsigned int item)`

Get an enumeration item value by index.

- `FMILIB_EXPORT const char * fmi2_import_get_enum_type_item_description (fmi2_import_enumeration_typedef_t *, unsigned int item)`

Get an enumeration item description by index.

- `FMILIB_EXPORT const char * fmi2_import_get_enum_type_value_name (fmi2_import_enumeration_typedef_t *t, int value)`

Get an enumeration item name for the given value.

Type definitions supporting structures

- `typedef struct fmi2_xml_real_typedef_t fmi2_import_real_typedef_t`

Opaque type definition object.

- `typedef struct fmi2_xml_integer_typedef_t fmi2_import_integer_typedef_t`

Opaque integer type definition object.

- `typedef struct fmi2_xml_enumeration_typedef_t fmi2_import_enumeration_typedef_t`

Opaque enumeration type definition object.

- `typedef struct fmi2_xml_variable_typedef_t fmi2_import_variable_typedef_t`
Opaque general variable type definition object.
- `typedef struct fmi2_xml_type_definitions_t fmi2_import_type_definitions_t`
Opaque list of the type definitions in the model.

6.42.1 Typedef Documentation

6.42.1.1 `typedef struct fmi2_xml_real_typedef_t fmi2_import_real_typedef_t`

Opaque type definition object.

Definition at line 42 of file `fmi2_import_type.h`.

6.42.1.2 `typedef struct fmi2_xml_integer_typedef_t fmi2_import_integer_typedef_t`

Opaque integer type definition object.

Definition at line 44 of file `fmi2_import_type.h`.

6.42.1.3 `typedef struct fmi2_xml_enumeration_typedef_t fmi2_import_enumeration_typedef_t`

Opaque enumeration type definition object.

Definition at line 46 of file `fmi2_import_type.h`.

6.42.1.4 `typedef struct fmi2_xml_variable_typedef_t fmi2_import_variable_typedef_t`

Opaque general variable type definition object.

Definition at line 48 of file `fmi2_import_type.h`.

6.42.1.5 `typedef struct fmi2_xml_type_definitions_t fmi2_import_type_definitions_t`

Opaque list of the type definitions in the model.

Definition at line 50 of file `fmi2_import_type.h`.

6.42.2 Function Documentation

6.42.2.1 `FMLIB_EXPORT unsigned int fmi2_import_get_type_definition_number ( fmi2_import_type_definitions_t * td )`

Get the number of available type definitions.

6.42.2.2 FMILIB_EXPORT fmi2_import_variable_typedef_t*
fmi2_import_get_typedef (fmi2_import_type_definitions_t * *td*, unsigned
int *index*)

Get a type definition specified by the index.

Parameters

<i>td</i>	the type definition list object
<i>index</i>	the index of type definition. Must be less than the numbere returned by fmi2_import_get_type_definition_number()

Returns

A type definition object or NULL if index is out of range.

6.42.2.3 FMILIB_EXPORT fmi2_import_display_unit_t *
fmi2_import_get_type_display_unit (fmi2_import_real_typedef_t *)

Get associated display unit for a type defition if any.

Get display unit associated with a real type definition.

Returns

Display unit object of NULL if none was given.

6.42.2.4 FMILIB_EXPORT const char* fmi2_import_get_type_name (
fmi2_import_variable_typedef_t *)

Get the type name.

6.42.2.5 FMILIB_EXPORT const char* fmi2_import_get_type_description (
fmi2_import_variable_typedef_t *)

Get type description.

Note that an empty string is returned if the attribute is not present in the XML.

6.42.2.6 FMILIB_EXPORT fmi2_base_type_enumeration_t fmi2_import_get_base_type (
fmi2_import_variable_typedef_t *)

Get base type used for the type definition.

6.42.2.7 FMILIB_EXPORT fmi2_import_real_typedef_t*
fmi2_import_get_type_as_real (fmi2_import_variable_typedef_t *)

Cast the general type definition object to an object with a specific base type.

Returns

Pointer to the specific type object or NULL if base type does not match.

6.42.2.8 FMILIB_EXPORT fmi2_import_integer_typedef_t*
fmi2_import_get_type_as_int (fmi2_import_variable_typedef_t *)

Cast the general type definition object to an object with a specific base type.

Returns

Pointer to the specific type object or NULL if base type does not match.

6.42.2.9 FMILIB_EXPORT fmi2_import_enumeration_typedef_t*
fmi2_import_get_type_as_enum (fmi2_import_variable_typedef_t *)

Cast the general type definition object to an object with a specific base type.

Returns

Pointer to the specific type object or NULL if base type does not match.

6.42.2.10 FMILIB_EXPORT const char* fmi2_import_get_type_quantity (
fmi2_import_variable_typedef_t *)

Get the quantity associated with the type definition.

Returns

NULL-pointer is always returned for strings and booleans. Empty string is returned if attribute is not present for other types.

6.42.2.11 FMILIB_EXPORT double fmi2_import_get_real_type_min (
fmi2_import_real_typedef_t *)

Get minimal value for the type.

Returns

Either the value specified in the XML file or negated DBL_MAX as defined in <float.h>

6.42.2.12 `FMILIB_EXPORT double fmi2_import_get_real_type_max (`
`fmi2_import_real_typedef_t * )`

Get maximum value for the type.

Returns

Either the value specified in the XML file or DBL_MAX as defined in <float.h>

6.42.2.13 `FMILIB_EXPORT double fmi2_import_get_real_type_nominal (`
`fmi2_import_real_typedef_t * )`

Get the nominal value associated with the type definition.

6.42.2.14 `FMILIB_EXPORT fmi2_import_unit_t* fmi2_import_get_real_type_unit (`
`fmi2_import_real_typedef_t * )`

Get the unit object associated with the type definition if any.

6.42.2.15 `FMILIB_EXPORT int fmi2_import_get_real_type_is_relative_quantity (`
`fmi2_import_real_typedef_t * )`

Get the 'relativeQuantity' flag.

6.42.2.16 `FMILIB_EXPORT int fmi2_import_get_real_type_is_unbounded (`
`fmi2_import_real_typedef_t * )`

Get the 'unbounded' flag.

6.42.2.17 `FMILIB_EXPORT int fmi2_import_get_integer_type_min (`
`fmi2_import_integer_typedef_t * )`

Get minimal value for the type.

Returns

Either the value specified in the XML file or INT_MIN as defined in <limits.h>

6.42.2.18 `FMILIB_EXPORT int fmi2_import_get_integer_type_max (`
`fmi2_import_integer_typedef_t * )`

Get maximum value for the type.

Returns

Either the value specified in the XML file or INT_MAX as defined in <limits.h>

6.42.2.19 FMILIB_EXPORT unsigned int fmi2_import_get_enum_type_min (
fmi2_import_enumeration_typedef_t *)

Get minimal value for the type.

Returns

Either the value specified in the XML file or 0

6.42.2.20 FMILIB_EXPORT unsigned int fmi2_import_get_enum_type_max (
fmi2_import_enumeration_typedef_t *)

Get maximum value for the type.

Returns

Either the value specified in the XML file or INT_MAX as defined in <limits.h>

6.42.2.21 FMILIB_EXPORT unsigned int fmi2_import_get_enum_type_size (
fmi2_import_enumeration_typedef_t *)

Get the number of elements in the enum.

6.42.2.22 FMILIB_EXPORT const char* fmi2_import_get_enum_type_item_name (
fmi2_import_enumeration_typedef_t * , unsigned int *item*)

Get an enumeration item name by index.

6.42.2.23 FMILIB_EXPORT int fmi2_import_get_enum_type_item_value (
fmi2_import_enumeration_typedef_t * , unsigned int *item*)

Get an enumeration item value by index.

6.42.2.24 FMILIB_EXPORT const char* fmi2_import_get_enum_type_item_
description (fmi2_import_enumeration_typedef_t * , unsigned int *item*
)

Get an enumeration item description by index.

6.42.2.25 FMILIB_EXPORT const char* fmi2_import_get_enum_type_value_name (
fmi2_import_enumeration_typedef_t * *t* , int *value*)

Get an enumeration item name for the given value.

6.43 Functions for handling unit definitions.

Functions

- **FMILIB_EXPORT** unsigned int `fmi2_import_get_unit_definitions_number` (`fmi2_import_unit_definitions_t *`)
Get the number of unit definitions.
- **FMILIB_EXPORT** `fmi2_import_unit_t *` `fmi2_import_get_unit` (`fmi2_import_unit_definitions_t *`, unsigned int index)
Get a unit definition.
- **FMILIB_EXPORT** const char * `fmi2_import_get_unit_name` (`fmi2_import_unit_t *`)
Get a unit name.
- **FMILIB_EXPORT** unsigned int `fmi2_import_get_unit_display_unit_number` (`fmi2_import_unit_t *`)
Get the number of display units associated with this unit.
- **FMILIB_EXPORT** const int * `fmi2_import_get_SI_unit_exponents` (`fmi2_import_unit_t *`)
Get fmi2_SI_base_units_Num SI base units exponents associated with the unit.
- **FMILIB_EXPORT** double `fmi2_import_get_SI_unit_factor` (`fmi2_import_unit_t *`)
Get factor to the corresponding SI base units.
- **FMILIB_EXPORT** double `fmi2_import_get_SI_unit_offset` (`fmi2_import_unit_t *`)
Get offset to the corresponding SI base units.
- **FMILIB_EXPORT** double `fmi2_import_convert_to_SI_base_unit` (double, `fmi2_import_unit_t *`)
Convert a value with respect to the unit to the value with respect to the SI base unit.
- **FMILIB_EXPORT** double `fmi2_import_convert_from_SI_base_unit` (double, `fmi2_import_unit_t *`)
Convert a value with respect to the SI base unit to the value with respect to the unit.
- **FMILIB_EXPORT** `fmi2_import_display_unit_t *` `fmi2_import_get_unit_display_unit` (`fmi2_import_unit_t *`, size_t index)
Get a display unit object by index.
- **FMILIB_EXPORT** `fmi2_import_unit_t *` `fmi2_import_get_base_unit` (`fmi2_import_display_unit_t *`)
Get unit definition for a display unit.
- **FMILIB_EXPORT** const char * `fmi2_import_get_display_unit_name` (`fmi2_import_display_unit_t *`)
Get display unit name.
- **FMILIB_EXPORT** `fmi2_real_t` `fmi2_import_get_display_unit_factor` (`fmi2_import_display_unit_t *`)
Get the "factor" associated with the display unit.
- **FMILIB_EXPORT** `fmi2_real_t` `fmi2_import_get_display_unit_offset` (`fmi2_import_display_unit_t *`)

Get the "offset" associated with the display unit.

- **FMILIB_EXPORT** fmi2_real_t fmi2_import_convert_to_display_unit (fmi2_real_t value, fmi2_import_display_unit_t *du, int isRelativeQuantity)

Convert a value measured in "units" to a value measured with "display units".

- **FMILIB_EXPORT** fmi2_real_t fmi2_import_convert_from_display_unit (fmi2_real_t value, fmi2_import_display_unit_t *du, int isRelativeQuantity)

Convert a value measured in "display units" to a value measured with "units".

Structures encapsulating unit information

- typedef struct fmi2_xml_unit_t fmi2_import_unit_t

A variable unit defined with a unit definition.

- typedef struct fmi2_xml_display_unit_t fmi2_import_display_unit_t

A display unit.

- typedef struct fmi2_xml_unit_definitions_t fmi2_import_unit_definitions_t

The list of all the unit definitions in the model.

6.43.1 Typedef Documentation

6.43.1.1 typedef struct fmi2_xml_unit_t fmi2_import_unit_t

A variable unit defined with a unit definition.

Definition at line 41 of file fmi2_import_unit.h.

6.43.1.2 typedef struct fmi2_xml_display_unit_t fmi2_import_display_unit_t

A display unit.

Definition at line 43 of file fmi2_import_unit.h.

6.43.1.3 typedef struct fmi2_xml_unit_definitions_t fmi2_import_unit_definitions_t

The list of all the unit definitions in the model.

Definition at line 45 of file fmi2_import_unit.h.

6.43.2 Function Documentation

6.43.2.1 FMILIB_EXPORT unsigned int fmi2_import_get_unit_definitions_number (fmi2_import_unit_definitions_t *)

Get the number of unit definitions.

6.43.2.2 FMILIB_EXPORT fmi2_import_unit_t* fmi2_import_get_unit (fmi2_import_unit_definitions_t *, unsigned int *index*)

Get a unit definition.

6.43.2.3 FMILIB_EXPORT const char* fmi2_import_get_unit_name (fmi2_import_unit_t *)

Get a unit name.

6.43.2.4 FMILIB_EXPORT unsigned int fmi2_import_get_unit_display_unit_number (fmi2_import_unit_t *)

Get the number of display units associated with this unit.

6.43.2.5 FMILIB_EXPORT const int* fmi2_import_get_SI_unit_exponents (fmi2_import_unit_t *)

Get fmi2_SI_base_units_Num SI base units exponents associated with the unit.

6.43.2.6 FMILIB_EXPORT double fmi2_import_get_SI_unit_factor (fmi2_import_unit_t *)

Get factor to the corresponding SI base units.

6.43.2.7 FMILIB_EXPORT double fmi2_import_get_SI_unit_offset (fmi2_import_unit_t *)

Get offset to the corresponding SI base units.

6.43.2.8 FMILIB_EXPORT double fmi2_import_convert_to_SI_base_unit (double , fmi2_import_unit_t *)

Convert a value with respect to the unit to the value with respect to the SI base unit.

6.43.2.9 FMILIB_EXPORT double fmi2_import_convert_from_SI_base_unit (double , fmi2_import_unit_t *)

Convert a value with respect to the SI base unit to the value with respect to the unit.

```
6.43.2.10 FMILIB_EXPORT fmi2_import_display_unit_t*
          fmi2_import_get_unit_display_unit ( fmi2_import_unit_t *, size_t index
          )
```

Get a display unit object by index.

Parameters

<i>index</i>	The index of display unit to be returned. Must be less than the number returned by fmi2_import_get_unit_display_unit_number()
--------------	---

```
6.43.2.11 FMILIB_EXPORT fmi2_import_unit_t* fmi2_import_get_base_unit (
          fmi2_import_display_unit_t * )
```

Get unit definition for a display unit.

```
6.43.2.12 FMILIB_EXPORT const char* fmi2_import_get_display_unit_name (
          fmi2_import_display_unit_t * )
```

Get display unit name.

```
6.43.2.13 FMILIB_EXPORT fmi2_real_t fmi2_import_get_display_unit_factor (
          fmi2_import_display_unit_t * )
```

Get the "factor" associated with the display unit.

```
6.43.2.14 FMILIB_EXPORT fmi2_real_t fmi2_import_get_display_unit_offset (
          fmi2_import_display_unit_t * )
```

Get the "offset" associated with the display unit.

```
6.43.2.15 FMILIB_EXPORT fmi2_real_t fmi2_import_convert_to_display_unit (
          fmi2_real_t value, fmi2_import_display_unit_t * du, int isRelativeQuantity )
```

Convert a value measured in "units" to a value measured with "display units".

Parameters

<i>value</i>	The value to be converted.
<i>du</i>	The display unit object
<i>isRelative-Quantity</i>	specifies if "offset" should be incorporated into conversion

6.43.2.16 **FMLIB_EXPORT** fmi2_real_t fmi2_import_convert_from_display_unit (fmi2_real_t *value*, fmi2_import_display_unit_t * *du*, int *isRelativeQuantity*)

Convert a value measured in "display units" to a value measured with "units".

Parameters

<i>value</i>	The value to be converted.
<i>du</i>	The display unit object
<i>isRelative-Quantity</i>	specifies if "offset" should be incorporated into conversion

6.44 Functions for handling variable definitions.

All the functions in this group take a pointer to `fmi2_import_variable_t` as a parameter. A variable pointer may be obtained via a [Handling of variable lists](#) module or via functions `fmi2_import_get_variable_by_name()` and `fmi2_import_get_variable_by_vr()`.

Functions

- `FMILIB_EXPORT const char * fmi2_import_get_variable_name (fmi2_import_variable_t *)`
Get the variable name.
- `FMILIB_EXPORT const char * fmi2_import_get_variable_description (fmi2_import_variable_t *)`
Get variable description.
- `FMILIB_EXPORT fmi2_value_reference_t fmi2_import_get_variable_vr (fmi2_import_variable_t *)`
Get variable value reference.
- `FMILIB_EXPORT fmi2_import_variable_typedef_t * fmi2_import_get_variable_declared_type (fmi2_import_variable_t *)`
For scalar variable gives the type definition is present.
- `FMILIB_EXPORT fmi2_base_type_enumeration_t fmi2_import_get_variable_base_type (fmi2_import_variable_t *)`
Get variable base type.
- `FMILIB_EXPORT int fmi2_import_get_variable_has_start (fmi2_import_variable_t *)`
Check if the variable has "start" attribute.
- `FMILIB_EXPORT fmi2_variability_enumeration_t fmi2_import_get_variability (fmi2_import_variable_t *)`
Get variability attribute.
- `FMILIB_EXPORT fmi2_causality_enumeration_t fmi2_import_get_causality (fmi2_import_variable_t *)`
Get causality attribute.
- `FMILIB_EXPORT fmi2_initial_enumeration_t fmi2_import_get_initial (fmi2_import_variable_t *)`
Get initial attribute.
- `FMILIB_EXPORT fmi2_import_variable_t * fmi2_import_get_previous (fmi2_import_variable_t *v)`
Get the variable that holds the previous value of this variable, if defined.
- `FMILIB_EXPORT fmi2_boolean_t fmi2_import_get_canHandleMultipleSetPerTimeInstant (fmi2_import_variable_t *v)`
Get the canHandleMultipleSetPerTimeInstant flag for a variable.
- `FMILIB_EXPORT fmi2_import_real_variable_t * fmi2_import_get_variable_as_real (fmi2_import_variable_t *)`
Cast general variable to a one with the specific type.

- `FMILIB_EXPORT fmi2_import_integer_variable_t * fmi2_import_get_variable_as_integer (fmi2_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi2_import_enum_variable_t * fmi2_import_get_variable_as_enum (fmi2_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi2_import_string_variable_t * fmi2_import_get_variable_as_string (fmi2_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi2_import_bool_variable_t * fmi2_import_get_variable_as_boolean (fmi2_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_start (fmi2_import_real_variable_t *v)`
Get the variable start attribute.
- `FMILIB_EXPORT fmi2_import_real_variable_t * fmi2_import_get_real_variable_derivative_of (fmi2_import_real_variable_t *v)`
Get the variable that this is a derivative of, if defined.
- `FMILIB_EXPORT fmi2_boolean_t fmi2_import_get_real_variable_reinit (fmi2_import_real_variable_t *v)`
Get the reinit flag for a real variable.
- `FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_max (fmi2_import_real_variable_t *v)`
Get maximum value for the variable.
- `FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_min (fmi2_import_real_variable_t *v)`
Get minimal value for the variable.
- `FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_nominal (fmi2_import_real_variable_t *v)`
Get nominal value for the variable.
- `FMILIB_EXPORT fmi2_import_unit_t * fmi2_import_get_real_variable_unit (fmi2_import_real_variable_t *v)`
Get associated "unit" object if any.
- `FMILIB_EXPORT fmi2_import_display_unit_t * fmi2_import_get_real_variable_display_unit (fmi2_import_real_variable_t *v)`
Get associated "display unit" object if any.
- `FMILIB_EXPORT const char * fmi2_import_get_string_variable_start (fmi2_import_string_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT fmi2_boolean_t fmi2_import_get_boolean_variable_start (fmi2_import_bool_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_integer_variable_start (fmi2_import_integer_variable_t *v)`
Get start value for the variable.

- `FMILIB_EXPORT int fmi2_import_get_integer_variable_min (fmi2_import_integer_variable_t *v)`
Get minimal value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_integer_variable_max (fmi2_import_integer_variable_t *v)`
Get max value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_enum_variable_start (fmi2_import_enum_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_enum_variable_min (fmi2_import_enum_variable_t *v)`
Get minimal value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_enum_variable_max (fmi2_import_enum_variable_t *v)`
Get max value for the variable.
- `FMILIB_EXPORT fmi2_variable_alias_kind_enumeration_t fmi2_import_get_variable_alias_kind (fmi2_import_variable_t *)`
Get the variable alias kind.
- `FMILIB_EXPORT size_t fmi2_import_get_variable_original_order (fmi2_import_variable_t *v)`
Get the original index in xml of the variable.

Scalar variable types

- `typedef struct fmi2_xml_variable_t fmi2_import_variable_t`
General variable type.
- `typedef struct fmi2_xml_real_variable_t fmi2_import_real_variable_t`
Opaque real variable.
- `typedef struct fmi2_xml_integer_variable_t fmi2_import_integer_variable_t`
Opaque integer variable.
- `typedef struct fmi2_xml_string_variable_t fmi2_import_string_variable_t`
Opaque string variable.
- `typedef struct fmi2_xml_enum_variable_t fmi2_import_enum_variable_t`
Opaque enumeration variable.
- `typedef struct fmi2_xml_bool_variable_t fmi2_import_bool_variable_t`
Opaque boolean variable.
- `typedef struct fmi2_import_variable_list_t fmi2_import_variable_list_t`
List of variables.

6.44.1 Detailed Description

All the functions in this group take a pointer to `fmi2_import_variable_t` as a parameter. A variable pointer may be obtained via a [Handling of variable lists](#) module or via functions `fmi2_import_get_variable_by_name()` and `fmi2_import_get_variable_by_vr()`.

6.44.2 Typedef Documentation

6.44.2.1 `typedef struct fmi2_xml_variable_t fmi2_import_variable_t`

General variable type.

This type is convenient to unify all the variable list operations. However, typed variables are needed to support specific attributes.

Definition at line 51 of file `fmi2_import_variable.h`.

6.44.2.2 `typedef struct fmi2_xml_real_variable_t fmi2_import_real_variable_t`

Opaque real variable.

Definition at line 53 of file `fmi2_import_variable.h`.

6.44.2.3 `typedef struct fmi2_xml_integer_variable_t fmi2_import_integer_variable_t`

Opaque integer variable.

Definition at line 55 of file `fmi2_import_variable.h`.

6.44.2.4 `typedef struct fmi2_xml_string_variable_t fmi2_import_string_variable_t`

Opaque string variable.

Definition at line 57 of file `fmi2_import_variable.h`.

6.44.2.5 `typedef struct fmi2_xml_enum_variable_t fmi2_import_enum_variable_t`

Opaque enumeration variable.

Definition at line 59 of file `fmi2_import_variable.h`.

6.44.2.6 `typedef struct fmi2_xml_bool_variable_t fmi2_import_bool_variable_t`

Opaque boolean variable.

Definition at line 61 of file `fmi2_import_variable.h`.

6.44.2.7 `typedef struct fmi2_import_variable_list_t fmi2_import_variable_list_t`

List of variables.

Definition at line 63 of file `fmi2_import_variable.h`.

6.44.3 Function Documentation

6.44.3.1 FMILIB_EXPORT `const char* fmi2_import_get_variable_name ( fmi2_import_variable_t * )`

Get the variable name.

6.44.3.2 FMILIB_EXPORT `const char* fmi2_import_get_variable_description ( fmi2_import_variable_t * )`

Get variable description.

Returns

Description string or empty string ("") if no description in the XML file was given.

6.44.3.3 FMILIB_EXPORT `fmi2_value_reference_t fmi2_import_get_variable_vr ( fmi2_import_variable_t * )`

Get variable value reference.

6.44.3.4 FMILIB_EXPORT `fmi2_import_variable_typedef_t* fmi2_import_get_variable_declared_type ( fmi2_import_variable_t * )`

For scalar variable gives the type definition is present.

Returns

Pointer of a type [fmi2_import_variable_typedef_t](#) object or NULL if not present.

6.44.3.5 FMILIB_EXPORT `fmi2_base_type_enumeration_t fmi2_import_get_variable_base_type ( fmi2_import_variable_t * )`

Get variable base type.

6.44.3.6 FMILIB_EXPORT `int fmi2_import_get_variable_has_start ( fmi2_import_variable_t * )`

Check if the variable has "start" attribute.

6.44.3.7 FMILIB_EXPORT `fmi2_variability_enumeration_t fmi2_import_get_variability ( fmi2_import_variable_t * )`

Get variability attribute.

6.44.3.8 FMILIB_EXPORT fmi2_causality_enu_t fmi2_import_get_causality (
fmi2_import_variable_t *)

Get causality attribute.

6.44.3.9 FMILIB_EXPORT fmi2_initial_enu_t fmi2_import_get_initial (
fmi2_import_variable_t *)

Get initial attribute.

6.44.3.10 FMILIB_EXPORT fmi2_import_variable_t* fmi2_import_get_previous (
fmi2_import_variable_t * v)

Get the variable that holds the previous value of this variable, if defined.

Returns

If this variable is a discrete-time state, return the variable holds its previous value;
 NULL otherwise.

6.44.3.11 FMILIB_EXPORT fmi2_boolean_t fmi2_import_get_canHandle-
MultipleSetPerTimeInstant (fmi2_import_variable_t * v
)

Get the canHandleMultipleSetPerTimeInstant flag for a variable.

Returns

For inputs: If false, then only one fmiSetXXX call is allowed at one super dense
 time instant. In other words, this input is not allowed to appear in an algebraic loop.

6.44.3.12 FMILIB_EXPORT fmi2_import_real_variable_t*
fmi2_import_get_variable_as_real (fmi2_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.44.3.13 FMILIB_EXPORT fmi2_import_integer_variable_t*
fmi2_import_get_variable_as_integer (fmi2_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.44.3.14 FMILIB_EXPORT fmi2_import_enum_variable_t*
fmi2_import_get_variable_as_enum (fmi2_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.44.3.15 FMILIB_EXPORT fmi2_import_string_variable_t*
fmi2_import_get_variable_as_string (fmi2_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.44.3.16 FMILIB_EXPORT fmi2_import_bool_variable_t*
fmi2_import_get_variable_as_boolean (fmi2_import_variable_t *)

Cast general variable to a one with the specific type.

Returns

Typed object or NULL if base type does not match

6.44.3.17 FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_start (
fmi2_import_real_variable_t * v)

Get the variable start attribute.

Returns

The "start" attribute as specified in the XML file or variable nominal value.

6.44.3.18 FMILIB_EXPORT fmi2_import_real_variable_t* fmi2_import_get_-
real_variable_derivative_of (fmi2_import_real_variable_t * v
)

Get the variable that this is a derivative of, if defined.

Returns

If this variable is a derivative, return the variable that it is a derivative of; NULL otherwise.

6.44.3.19 FMILIB_EXPORT fmi2_boolean_t fmi2_import_get_real_variable_reinit (fmi2_import_real_variable_t * v)

Get the reinit flag for a real variable.

Returns

True if the real variable may change value at events.

6.44.3.20 FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_max (fmi2_import_real_variable_t * v)

Get maximum value for the variable.

Returns

Either the value specified in the XML file or DBL_MAX as defined in <float.h>

6.44.3.21 FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_min (fmi2_import_real_variable_t * v)

Get minimal value for the variable.

Returns

Either the value specified in the XML file or negated DBL_MAX as defined in <float.h>

6.44.3.22 FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_nominal (fmi2_import_real_variable_t * v)

Get nominal value for the variable.

6.44.3.23 FMILIB_EXPORT fmi2_import_unit_t* fmi2_import_get_real_variable_unit (fmi2_import_real_variable_t * v)

Get associated "unit" object if any.

6.44.3.24 **FMILIB_EXPORT** fmi2_import_display_unit_t* fmi2_import_get_real_variable_display_unit (fmi2_import_real_variable_t * v)

Get associated "display unit" object if any.

6.44.3.25 **FMILIB_EXPORT** const char* fmi2_import_get_string_variable_start (fmi2_import_string_variable_t * v)

Get start value for the variable.

6.44.3.26 **FMILIB_EXPORT** fmi2_boolean_t fmi2_import_get_boolean_variable_start (fmi2_import_bool_variable_t * v)

Get start value for the variable.

6.44.3.27 **FMILIB_EXPORT** int fmi2_import_get_integer_variable_start (fmi2_import_integer_variable_t * v)

Get start value for the variable.

6.44.3.28 **FMILIB_EXPORT** int fmi2_import_get_integer_variable_min (fmi2_import_integer_variable_t * v)

Get minimal value for the variable.

6.44.3.29 **FMILIB_EXPORT** int fmi2_import_get_integer_variable_max (fmi2_import_integer_variable_t * v)

Get max value for the variable.

6.44.3.30 **FMILIB_EXPORT** int fmi2_import_get_enum_variable_start (fmi2_import_enum_variable_t * v)

Get start value for the variable.

6.44.3.31 **FMILIB_EXPORT** int fmi2_import_get_enum_variable_min (fmi2_import_enum_variable_t * v)

Get minimal value for the variable.

6.44.3.32 **FMILIB_EXPORT** int fmi2_import_get_enum_variable_max (
 fmi2_import_enum_variable_t * v)

Get max value for the variable.

6.44.3.33 **FMILIB_EXPORT** fmi2_variable_alias_kind_enumeration_t
 fmi2_import_get_variable_alias_kind (fmi2_import_variable_t * v)

Get the variable alias kind.

6.44.3.34 **FMILIB_EXPORT** size_t fmi2_import_get_variable_original_order (
 fmi2_import_variable_t * v)

Get the original index in xml of the variable.

Chapter 7

Directory Documentation

7.1 build/ Directory Reference

Directories

- directory [doc](#)

7.2 build/doc/ Directory Reference

Files

- file [fmilib.h](#)
Include file to be used in client applications of the FMI Library.
- file [fmilib_config.h](#)
Library configuration file generated by the build system.
- file [fmilib_mainpage.h](#)
Autogenerated file with documentation.

7.3 src/Util/include/FMI/ Directory Reference

Files

- file [fmi_util.h](#)
Some low-level utility functions suitable for all standards.
- file [fmi_version.h](#)
Enum defining supported FMI versions.

7.4 src/Import/include/FMI/ Directory Reference

Files

- file [fmi_import_context.h](#)

Import context is the entry point to the library. It is used to initialize, unzip, get FMI version and start parsing.

- file [fmi_import_util.h](#)

7.5 Test/FMI1/ Directory Reference

Directories

- directory [fmu_dummy](#)

Files

- file [compress_test_fmu_zip.c](#)
- file [fmi1_capi_cs_test.c](#)
- file [fmi1_capi_me_test.c](#)
- file [fmi1_import_test.c](#)
- file [fmi1_logger_test.c](#)
- file [fmi2_import_xml_test.cc](#)
- file [fmi_import_cs_test.c](#)
- file [fmi_import_me_test.c](#)
- file [fmi_import_xml_test.cc](#)
- file [fmi_total_test.c](#)
- file [fmi_zip_unzip_test.c](#)
- file [fmi_zip_zip_test.c](#)
- file [jm_vector_test.c](#)

7.6 src/Util/include/FMI1/ Directory Reference

Files

- file [fmi1_enums.h](#)

Definitions the enum types used with FMI 1.0 libs.

- file [fmi1_functions.h](#)
- file [fmi1_types.h](#)

7.7 src/Import/include/FMI1/ Directory Reference

Files

- file [fmi1_import.h](#)
Public interface to the FMI import C-library.
- file [fmi1_import_capabilities.h](#)
- file [fmi1_import_capi.h](#)
- file [fmi1_import_convenience.h](#)
Public interface to the FMI import C-library. Convenience functions.
- file [fmi1_import_cosim.h](#)
- file [fmi1_import_type.h](#)
Public interface to the FMI XML C-library: variable types handling.
- file [fmi1_import_unit.h](#)
Public interface to the FMI import C-library. Handling of variable units.
- file [fmi1_import_variable.h](#)
Public interface to the FMI import C-library. Handling of model variables.
- file [fmi1_import_variable_list.h](#)
Public interface to the FMI XML C-library. Handling of variable lists.
- file [fmi1_import_vendor_annotations.h](#)
Public interface to the FMI XML C-library. Handling of vendor annotations.

7.8 Test/FMI2/ Directory Reference

Directories

- directory [fmu_dummy](#)

Files

- file [fmi2_import_cs_test.c](#)
- file [fmi2_import_me_test.c](#)
- file [fmi2_import_test.c](#)
- file [fmi2_import_xml_test.cc](#)

7.9 src/Util/include/FMI2/ Directory Reference

Files

- file [fmi2_enums.h](#)
Definitions the enum types used with FMI 2.0 libs.
- file [fmi2_functions.h](#)
- file [fmi2_types.h](#)
- file [fmi2_xml_callbacks.h](#)

7.10 src/Import/include/FMI2/ Directory Reference

Files

- file [fmi2_import.h](#)
Public interface to the FMI import C-library.
- file [fmi2_import_capi.h](#)
- file [fmi2_import_convenience.h](#)
Public interface to the FMI import C-library. Convenience functions.
- file [fmi2_import_type.h](#)
Public interface to the FMI XML C-library: variable types handling.
- file [fmi2_import_unit.h](#)
Public interface to the FMI import C-library. Handling of variable units.
- file [fmi2_import_variable.h](#)
Public interface to the FMI import C-library. Handling of model variables.
- file [fmi2_import_variable_list.h](#)
Public interface to the FMI XML C-library. Handling of variable lists.

7.11 Test/FMI2/fmu_dummy/ Directory Reference

Files

- file [fmu2_model.c](#)
- file [fmu2_model.h](#)
- file [fmu2_model_cs.c](#)
- file [fmu2_model_defines.h](#)
- file [fmu2_model_me.c](#)

7.12 Test/FMI1/fmu_dummy/ Directory Reference

Files

- file [fmu1_model.c](#)
- file [fmu1_model.h](#)
- file [fmu1_model_cs.c](#)
- file [fmu1_model_defines.h](#)
- file [fmu1_model_me.c](#)

7.13 src/Import/ Directory Reference

Directories

- directory [include](#)

7.14 src/Util/include/ Directory Reference

Directories

- directory [FMI](#)
- directory [FMI1](#)
- directory [FMI2](#)
- directory [JM](#)

7.15 src/Import/include/ Directory Reference

Directories

- directory [FMI](#)
- directory [FMI1](#)
- directory [FMI2](#)

7.16 src/Util/include/JM/ Directory Reference

Files

- file [jm_callbacks.h](#)
- file [jm_named_ptr.h](#)
- file [jm_portability.h](#)
- file [jm_stack.h](#)
- file [jm_string_set.h](#)
- file [jm_types.h](#)
- file [jm_vector.h](#)
- file [jm_vector_template.h](#)

Vector template definition.

7.17 src/ Directory Reference

Directories

- directory [Import](#)
- directory [Util](#)

7.18 Test/ Directory Reference

Directories

- directory [FMI1](#)
- directory [FMI2](#)

Files

- file [compress_test_fmu_zip.c](#)
- file [fmi_import_test.c](#)
- file [fmi_zip_unzip_test.c](#)
- file [fmi_zip_zip_test.c](#)
- file [jm_vector_test.c](#)

7.19 src/Util/ Directory Reference

Directories

- directory [include](#)

Chapter 8

Data Structure Documentation

8.1 component_t Struct Reference

```
#include <fmul_model.h>
```

Data Fields

- [fmiReal states \[N_STATES\]](#)
- [fmiReal states_nom \[N_STATES\]](#)
- [fmiValueReference states_vr \[N_STATES\]](#)
- [fmiReal states_der \[N_STATES\]](#)
- [fmiReal event_indicators \[N_EVENT_INDICATORS\]](#)
- [fmiReal reals \[N_REAL\]](#)
- [fmiInteger integers \[N_INTEGER\]](#)
- [fmiBoolean booleans \[N_BOOLEAN\]](#)
- [fmiString strings \[N_STRING\]](#)
- [fmiBoolean loggingOn](#)
- [char instanceName \[BUFFER\]](#)
- [char GUID \[BUFFER\]](#)
- [fmiCallbackFunctions functions](#)
- [fmiReal fmitime](#)
- [fmiBoolean callEventUpdate](#)
- [fmiBoolean toleranceControlled](#)
- [fmiReal relativeTolerance](#)
- [fmiEventInfo eventInfo](#)
- [fmiReal states_prev \[N_STATES\]](#)
- [char fmuLocation \[BUFFER\]](#)
- [char mimeType \[BUFFER\]](#)
- [fmiReal timeout](#)
- [fmiBoolean visible](#)
- [fmiBoolean interactive](#)

- fmiReal tStart
- fmiBoolean StopTimeDefined
- fmiReal tStop
- fmiReal input_real [N_INPUT_REAL][N_INPUT_REAL_MAX_ORDER+1]
- fmiReal output_real [N_OUTPUT_REAL][N_OUTPUT_REAL_MAX_ORDER+1]
- fmi2Real states [N_STATES]
- fmi2Real states_nom [N_STATES]
- fmi2Real states_der [N_STATES]
- fmi2Real event_indicators [N_EVENT_INDICATORS]
- fmi2Real reals [N_REAL]
- fmi2Integer integers [N_INTEGER]
- fmi2Boolean booleans [N_BOOLEAN]
- fmi2String strings [N_STRING]
- fmi2Boolean loggingOn
- const fmi2CallbackFunctions * functions
- fmi2Real fmitime
- fmi2Boolean toleranceControlled
- fmi2Real relativeTolerance
- fmi2EventInfo eventInfo
- fmi2Real states_prev [N_STATES]
- fmi2Boolean visible
- fmi2Real tStart
- fmi2Boolean StopTimeDefined
- fmi2Real tStop
- fmi2Real input_real [N_INPUT_REAL][N_INPUT_REAL_MAX_ORDER+1]
- fmi2Real output_real [N_OUTPUT_REAL][N_OUTPUT_REAL_MAX_ORDER+1]

8.1.1 Detailed Description

Definition at line 23 of file fmu1_model.h.

8.1.2 Field Documentation

8.1.2.1 fmiReal component_t::states[N_STATES]

Definition at line 25 of file fmu1_model.h.

8.1.2.2 fmiReal component_t::states_nom[N_STATES]

Definition at line 26 of file fmu1_model.h.

8.1.2.3 fmiValueReference component_t::states_vr[N_STATES]

Definition at line 27 of file fmu1_model.h.

8.1.2.4 fmiReal component_t::states_der[N_STATES]

Definition at line 28 of file fmu1_model.h.

8.1.2.5 fmiReal component_t::event_indicators[N_EVENT_INDICATORS]

Definition at line 29 of file fmu1_model.h.

8.1.2.6 fmiReal component_t::reals[N_REAL]

Definition at line 30 of file fmu1_model.h.

8.1.2.7 fmiInteger component_t::integers[N_INTEGER]

Definition at line 31 of file fmu1_model.h.

8.1.2.8 fmiBoolean component_t::booleans[N_BOOLEAN]

Definition at line 32 of file fmu1_model.h.

8.1.2.9 fmiString component_t::strings[N_STRING]

Definition at line 33 of file fmu1_model.h.

8.1.2.10 fmiBoolean component_t::loggingOn

Definition at line 36 of file fmu1_model.h.

8.1.2.11 char component_t::instanceName

Definition at line 37 of file fmu1_model.h.

8.1.2.12 char component_t::GUID

Definition at line 38 of file fmu1_model.h.

8.1.2.13 fmiCallbackFunctions component_t::functions

Definition at line 39 of file fmu1_model.h.

8.1.2.14 fmiReal component_t::fmitime

Definition at line 42 of file fmu1_model.h.

8.1.2.15 fmiBoolean component_t::callEventUpdate

Definition at line 45 of file fmu1_model.h.

8.1.2.16 fmiBoolean component_t::toleranceControlled

Definition at line 48 of file fmu1_model.h.

8.1.2.17 fmiReal component_t::relativeTolerance

Definition at line 49 of file fmu1_model.h.

8.1.2.18 fmiEventInfo component_t::eventInfo

Definition at line 50 of file fmu1_model.h.

8.1.2.19 fmiReal component_t::states_prev[N_STATES]

Definition at line 53 of file fmu1_model.h.

8.1.2.20 char component_t::fmuLocation

Definition at line 56 of file fmu1_model.h.

8.1.2.21 char component_t::mimeType[BUFFER]

Definition at line 57 of file fmu1_model.h.

8.1.2.22 fmiReal component_t::timeout

Definition at line 58 of file fmu1_model.h.

8.1.2.23 fmiBoolean component_t::visible

Definition at line 59 of file fmu1_model.h.

8.1.2.24 fmiBoolean component_t::interactive

Definition at line 60 of file fmu1_model.h.

8.1.2.25 fmiReal component_t::tStart

Definition at line 63 of file fmu1_model.h.

8.1.2.26 fmiBoolean component_t::StopTimeDefined

Definition at line 64 of file fmu1_model.h.

8.1.2.27 fmiReal component_t::tStop

Definition at line 65 of file fmu1_model.h.

8.1.2.28 fmiReal component_t::input_real[N_INPUT_REAL][N_INPUT_REAL_MAX_ORDER+1]

Definition at line 68 of file fmu1_model.h.

8.1.2.29 fmiReal component_t::output_real[N_OUTPUT_REAL][N_OUTPUT_REAL_MAX_ORDER+1]

Definition at line 71 of file fmu1_model.h.

8.1.2.30 fmi2Real component_t::states[N_STATES]

Definition at line 28 of file fmu2_model.h.

8.1.2.31 fmi2Real component_t::states_nom[N_STATES]

Definition at line 29 of file fmu2_model.h.

8.1.2.32 fmi2Real component_t::states_der[N_STATES]

Definition at line 30 of file fmu2_model.h.

8.1.2.33 fmi2Real component_t::event_indicators[N_EVENT_INDICATORS]

Definition at line 31 of file fmu2_model.h.

8.1.2.34 fmi2Real component_t::reals[N_REAL]

Definition at line 32 of file fmu2_model.h.

8.1.2.35 fmi2Integer component_t::integers[N_INTEGER]

Definition at line 33 of file fmu2_model.h.

8.1.2.36 fmi2Boolean component_t::booleans[N_BOOLEAN]

Definition at line 34 of file fmu2_model.h.

8.1.2.37 fmi2String component_t::strings[N_STRING]

Definition at line 35 of file fmu2_model.h.

8.1.2.38 fmi2Boolean component_t::loggingOn

Definition at line 38 of file fmu2_model.h.

8.1.2.39 const fmi2CallbackFunctions* component_t::functions

Definition at line 41 of file fmu2_model.h.

8.1.2.40 fmi2Real component_t::fmitime

Definition at line 44 of file fmu2_model.h.

8.1.2.41 fmi2Boolean component_t::toleranceControlled

Definition at line 47 of file fmu2_model.h.

8.1.2.42 fmi2Real component_t::relativeTolerance

Definition at line 48 of file fmu2_model.h.

8.1.2.43 fmi2EventInfo component_t::eventInfo

Definition at line 49 of file fmu2_model.h.

8.1.2.44 fmi2Real component_t::states_prev[N_STATES]

Definition at line 52 of file fmu2_model.h.

8.1.2.45 fmi2Boolean component_t::visible

Definition at line 56 of file fmu2_model.h.

8.1.2.46 fmi2Real component_t::tStart

Definition at line 59 of file fmu2_model.h.

8.1.2.47 fmi2Boolean component_t::StopTimeDefined

Definition at line 60 of file fmu2_model.h.

8.1.2.48 fmi2Real component_t::tStop

Definition at line 61 of file fmu2_model.h.

8.1.2.49 fmi2Real component_t::input_real[N_INPUT_REAL][N_INPUT_REAL_MAX_ORDER+1]

Definition at line 64 of file fmu2_model.h.

8.1.2.50 fmi2Real component_t::output_real[N_OUTPUT_REAL][N_OUTPUT_REAL_MAX_ORDER+1]

Definition at line 67 of file fmu2_model.h.

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

- Test/FMI1/fmu_dummy/fmu1_model.h
- Test/FMI2/fmu_dummy/fmu2_model.h

8.2 fmi1_callback_functions_t Struct Reference

```
#include <fmi1_functions.h>
```

Data Fields

- [fmi1_callback_logger_ft](#) logger
- [fmi1_callback_allocate_memory_ft](#) allocateMemory

- [fmi1_callback_free_memory_ft freeMemory](#)
- [fmi1_step_finished_ft stepFinished](#)

8.2.1 Detailed Description

The FMI 1.0 CS structure adds one field to the ME, otherwise compatible
Definition at line 60 of file fmi1_functions.h.

8.2.2 Field Documentation

8.2.2.1 `fmi1_callback_logger_ft fmi1_callback_functions_t::logger`

Definition at line 61 of file fmi1_functions.h.

8.2.2.2 `fmi1_callback_allocate_memory_ft fmi1_callback_functions_t::allocateMemory`

Definition at line 62 of file fmi1_functions.h.

8.2.2.3 `fmi1_callback_free_memory_ft fmi1_callback_functions_t::freeMemory`

Definition at line 63 of file fmi1_functions.h.

8.2.2.4 `fmi1_step_finished_ft fmi1_callback_functions_t::stepFinished`

Definition at line 64 of file fmi1_functions.h.

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

- `src/Util/include/FMI1/fmi1_functions.h`

8.3 `fmi1_event_info_t` Struct Reference

```
#include <fmi1_functions.h>
```

Data Fields

- `fmi1_boolean_t iterationConverged`
- `fmi1_boolean_t stateValueReferencesChanged`
- `fmi1_boolean_t stateValuesChanged`
- `fmi1_boolean_t terminateSimulation`
- `fmi1_boolean_t upcomingTimeEvent`
- `fmi1_real_t nextEventTime`

8.3.1 Detailed Description

Event info structure as used in FMI 1.0 ME

Definition at line 68 of file fmi1_functions.h.

8.3.2 Field Documentation

8.3.2.1 fmi1_boolean_t fmi1_event_info_t::iterationConverged

Definition at line 69 of file fmi1_functions.h.

8.3.2.2 fmi1_boolean_t fmi1_event_info_t::stateValueReferencesChanged

Definition at line 70 of file fmi1_functions.h.

8.3.2.3 fmi1_boolean_t fmi1_event_info_t::stateValuesChanged

Definition at line 71 of file fmi1_functions.h.

8.3.2.4 fmi1_boolean_t fmi1_event_info_t::terminateSimulation

Definition at line 72 of file fmi1_functions.h.

8.3.2.5 fmi1_boolean_t fmi1_event_info_t::upcomingTimeEvent

Definition at line 73 of file fmi1_functions.h.

8.3.2.6 fmi1_real_t fmi1_event_info_t::nextEventTime

Definition at line 74 of file fmi1_functions.h.

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

- [src/Util/include/FMI1/fmi1_functions.h](#)

8.4 fmi1_import_model_counts_t Struct Reference

Collection of counters providing model information.

```
#include <fmi1_import_convenience.h>
```

Data Fields

- unsigned int [num_constants](#)
Number of constants.
- unsigned int [num_parameters](#)
Number of parameters.
- unsigned int [num_discrete](#)
Number of discrete variables.
- unsigned int [num_continuous](#)
Number of continuous variables.
- unsigned int [num_inputs](#)
Number of inputs.
- unsigned int [num_outputs](#)
Number of outputs.
- unsigned int [num_internal](#)
Number of internal variables.
- unsigned int [num_causality_none](#)
Number of variables with causality 'none'.
- unsigned int [num_real_vars](#)
Number of real variables.
- unsigned int [num_integer_vars](#)
Number of integer variables.
- unsigned int [num_enum_vars](#)
Number of enumeration variables.
- unsigned int [num_bool_vars](#)
Number of boolean variables.
- unsigned int [num_string_vars](#)
Number of string variables.

8.4.1 Detailed Description

Collection of counters providing model information.

Definition at line 48 of file `fmi1_import_convenience.h`.

8.4.2 Field Documentation

8.4.2.1 unsigned int `fmi1_import_model_counts_t::num_constants`

Number of constants.

Definition at line 50 of file `fmi1_import_convenience.h`.

8.4.2.2 unsigned int fmi1_import_model_counts_t::num_parameters

Number of parameters.

Definition at line 52 of file fmi1_import_convenience.h.

8.4.2.3 unsigned int fmi1_import_model_counts_t::num_discrete

Number of discrete variables.

Definition at line 55 of file fmi1_import_convenience.h.

8.4.2.4 unsigned int fmi1_import_model_counts_t::num_continuous

Number of continuous variables.

Definition at line 57 of file fmi1_import_convenience.h.

8.4.2.5 unsigned int fmi1_import_model_counts_t::num_inputs

Number of inputs.

Definition at line 60 of file fmi1_import_convenience.h.

8.4.2.6 unsigned int fmi1_import_model_counts_t::num_outputs

Number of outputs.

Definition at line 62 of file fmi1_import_convenience.h.

8.4.2.7 unsigned int fmi1_import_model_counts_t::num_internal

Number of internal variables.

Definition at line 64 of file fmi1_import_convenience.h.

8.4.2.8 unsigned int fmi1_import_model_counts_t::num_causality_none

Number of variables with causality 'none'.

Definition at line 66 of file fmi1_import_convenience.h.

8.4.2.9 unsigned int fmi1_import_model_counts_t::num_real_vars

Number of real variables.

Definition at line 69 of file fmi1_import_convenience.h.

8.4.2.10 `unsigned int fmi1_import_model_counts_t::num_integer_vars`

Number of integer variables.

Definition at line 71 of file `fmi1_import_convenience.h`.

8.4.2.11 `unsigned int fmi1_import_model_counts_t::num_enum_vars`

Number of enumeration variables.

Definition at line 73 of file `fmi1_import_convenience.h`.

8.4.2.12 `unsigned int fmi1_import_model_counts_t::num_bool_vars`

Number of boolean variables.

Definition at line 75 of file `fmi1_import_convenience.h`.

8.4.2.13 `unsigned int fmi1_import_model_counts_t::num_string_vars`

Number of string variables.

Definition at line 77 of file `fmi1_import_convenience.h`.

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

- `src/Import/include/FMI1/fmi1_import_convenience.h`

8.5 `fmi1_me_callback_functions_t` Struct Reference

```
#include <fmil_functions.h>
```

Data Fields

- `fmi1_callback_logger_ft` `logger`
- `fmi1_callback_allocate_memory_ft` `allocateMemory`
- `fmi1_callback_free_memory_ft` `freeMemory`

8.5.1 Detailed Description

Functions for FMI 1.0 ME

Definition at line 53 of file `fmi1_functions.h`.

8.5.2 Field Documentation

8.5.2.1 fmi1_callback_logger_ft fmi1_me_callback_functions_t::logger

Definition at line 54 of file fmi1_functions.h.

8.5.2.2 fmi1_callback_allocate_memory_ft fmi1_me_callback_functions_t::allocateMemory

Definition at line 55 of file fmi1_functions.h.

8.5.2.3 fmi1_callback_free_memory_ft fmi1_me_callback_functions_t::freeMemory

Definition at line 56 of file fmi1_functions.h.

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

- [src/Util/include/FMI1/fmi1_functions.h](#)

8.6 fmi2_callback_functions_t Struct Reference

```
#include <fmi2_functions.h>
```

Data Fields

- [fmi2_callback_logger_ft](#) logger
- [fmi2_callback_allocate_memory_ft](#) allocateMemory
- [fmi2_callback_free_memory_ft](#) freeMemory
- [fmi2_step_finished_ft](#) stepFinished
- [fmi2_component_environment_t](#) [componentEnvironment](#)

8.6.1 Detailed Description

The FMI 2.0 callbacks

Definition at line 70 of file fmi2_functions.h.

8.6.2 Field Documentation

8.6.2.1 fmi2_callback_logger_ft fmi2_callback_functions_t::logger

Definition at line 71 of file fmi2_functions.h.

8.6.2.2 `fmi2_callback_allocate_memory_ft` `fmi2_callback_functions_t::allocateMemory`

Definition at line 72 of file `fmi2_functions.h`.

8.6.2.3 `fmi2_callback_free_memory_ft` `fmi2_callback_functions_t::freeMemory`

Definition at line 73 of file `fmi2_functions.h`.

8.6.2.4 `fmi2_step_finished_ft` `fmi2_callback_functions_t::stepFinished`

Definition at line 74 of file `fmi2_functions.h`.

8.6.2.5 `fmi2_component_environment_t` `fmi2_callback_functions_t::componentEnvironment`

Definition at line 75 of file `fmi2_functions.h`.

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

- `src/Util/include/FMI2/fmi2_functions.h`

8.7 `fmi2_event_info_t` Struct Reference

```
#include <fmi2_functions.h>
```

Data Fields

- `fmi2_boolean_t` [newDiscreteStatesNeeded](#)
- `fmi2_boolean_t` [terminateSimulation](#)
- `fmi2_boolean_t` [nominalsOfContinuousStatesChanged](#)
- `fmi2_boolean_t` [valuesOfContinuousStatesChanged](#)
- `fmi2_boolean_t` [nextEventTimeDefined](#)
- `fmi2_real_t` [nextEventTime](#)

8.7.1 Detailed Description

Event info structure as used in FMI 2.0 ME

Definition at line 80 of file `fmi2_functions.h`.

8.7.2 Field Documentation

8.7.2.1 fmi2_boolean_t fmi2_event_info_t::newDiscreteStatesNeeded

Definition at line 81 of file fmi2_functions.h.

8.7.2.2 fmi2_boolean_t fmi2_event_info_t::terminateSimulation

Definition at line 82 of file fmi2_functions.h.

8.7.2.3 fmi2_boolean_t fmi2_event_info_t::nominalsOfContinuousStatesChanged

Definition at line 83 of file fmi2_functions.h.

8.7.2.4 fmi2_boolean_t fmi2_event_info_t::valuesOfContinuousStatesChanged

Definition at line 84 of file fmi2_functions.h.

8.7.2.5 fmi2_boolean_t fmi2_event_info_t::nextEventTimeDefined

Definition at line 85 of file fmi2_functions.h.

8.7.2.6 fmi2_real_t fmi2_event_info_t::nextEventTime

Definition at line 86 of file fmi2_functions.h.

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

- [src/Util/include/FMI2/fmi2_functions.h](#)

8.8 fmi2_import_model_counts_t Struct Reference

Collection of counters providing model information.

```
#include <fmi2_import_convenience.h>
```

Data Fields

- unsigned int [num_constants](#)
Number of constants.
- unsigned int [num_fixed](#)
Number of fixed.
- unsigned int [num_tunable](#)

Number of tunable.

- unsigned int [num_discrete](#)

Number of discrete variables.

- unsigned int [num_continuous](#)

Number of continuous variables.

- unsigned int [num_parameters](#)

Number of parameters.

- unsigned int [num_calculated_parameters](#)

Number of calculated parameters.

- unsigned int [num_inputs](#)

Number of inputs.

- unsigned int [num_outputs](#)

Number of outputs.

- unsigned int [num_local](#)

Number of local variables.

- unsigned int [num_independent](#)

Number of independent variables.

- unsigned int [num_real_vars](#)

Number of real variables.

- unsigned int [num_integer_vars](#)

Number of integer variables.

- unsigned int [num_enum_vars](#)

Number of enumeration variables.

- unsigned int [num_bool_vars](#)

Number of boolean variables.

- unsigned int [num_string_vars](#)

Number of string variables.

8.8.1 Detailed Description

Collection of counters providing model information.

Definition at line 48 of file `fmi2_import_convenience.h`.

8.8.2 Field Documentation

8.8.2.1 unsigned int `fmi2_import_model_counts_t::num_constants`

Number of constants.

Definition at line 50 of file `fmi2_import_convenience.h`.

8.8.2.2 unsigned int fmi2_import_model_counts_t::num_fixed

Number of fixed.

Definition at line 52 of file fmi2_import_convenience.h.

8.8.2.3 unsigned int fmi2_import_model_counts_t::num_tunable

Number of tunable.

Definition at line 54 of file fmi2_import_convenience.h.

8.8.2.4 unsigned int fmi2_import_model_counts_t::num_discrete

Number of discrete variables.

Definition at line 56 of file fmi2_import_convenience.h.

8.8.2.5 unsigned int fmi2_import_model_counts_t::num_continuous

Number of continuous variables.

Definition at line 58 of file fmi2_import_convenience.h.

8.8.2.6 unsigned int fmi2_import_model_counts_t::num_parameters

Number of parameters.

Definition at line 61 of file fmi2_import_convenience.h.

8.8.2.7 unsigned int fmi2_import_model_counts_t::num_calculated_parameters

Number of calculated parameters.

Definition at line 63 of file fmi2_import_convenience.h.

8.8.2.8 unsigned int fmi2_import_model_counts_t::num_inputs

Number of inputs.

Definition at line 65 of file fmi2_import_convenience.h.

8.8.2.9 unsigned int fmi2_import_model_counts_t::num_outputs

Number of outputs.

Definition at line 67 of file fmi2_import_convenience.h.

8.8.2.10 unsigned int fmi2_import_model_counts_t::num_local

Number of local variables.

Definition at line 69 of file fmi2_import_convenience.h.

8.8.2.11 unsigned int fmi2_import_model_counts_t::num_independent

Number of independent variables.

Definition at line 71 of file fmi2_import_convenience.h.

8.8.2.12 unsigned int fmi2_import_model_counts_t::num_real_vars

Number of real variables.

Definition at line 74 of file fmi2_import_convenience.h.

8.8.2.13 unsigned int fmi2_import_model_counts_t::num_integer_vars

Number of integer variables.

Definition at line 76 of file fmi2_import_convenience.h.

8.8.2.14 unsigned int fmi2_import_model_counts_t::num_enum_vars

Number of enumeration variables.

Definition at line 78 of file fmi2_import_convenience.h.

8.8.2.15 unsigned int fmi2_import_model_counts_t::num_bool_vars

Number of boolean variables.

Definition at line 80 of file fmi2_import_convenience.h.

8.8.2.16 unsigned int fmi2_import_model_counts_t::num_string_vars

Number of string variables.

Definition at line 82 of file fmi2_import_convenience.h.

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

- [src/Import/include/FMI2/fmi2_import_convenience.h](#)

8.9 fmi2_xml_callbacks_t Struct Reference

XML callbacks are used to process parts of XML that are not handled by the library.

```
#include <fmi2_xml_callbacks.h>
```

Data Fields

- [fmi2_xml_element_start_handle_ft startHandle](#)
- [fmi2_xml_element_data_handle_ft dataHandle](#)
Handle start of an XML element within tool annotation in a SAX parser.
- [fmi2_xml_element_end_handle_ft endHandle](#)
Handle data of an XML element within tool annotation in a SAX parser.
- void * [context](#)
Handle end of an XML element within tool annotation in a SAX parser.

8.9.1 Detailed Description

XML callbacks are used to process parts of XML that are not handled by the library.

Definition at line 67 of file [fmi2_xml_callbacks.h](#).

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

- [src/Util/include/FMI2/fmi2_xml_callbacks.h](#)

8.10 fmul_t Struct Reference

Data Fields

- [fmi1_import_t * fmu](#)
- [fmi_import_context_t * context](#)
- [jm_callbacks * callbacks](#)
- [fmi1_callback_functions_t callBackFunctions](#)

8.10.1 Detailed Description

Definition at line 167 of file [fmi_import_me_test.c](#).

8.10.2 Field Documentation

8.10.2.1 fmi1_import_t* fmul_t::fmu

Definition at line 168 of file [fmi_import_me_test.c](#).

8.10.2.2 fmi_import_context_t* fmul_t::context

Definition at line 169 of file fmi_import_me_test.c.

8.10.2.3 jm_callbacks* fmul_t::callbacks

Definition at line 170 of file fmi_import_me_test.c.

8.10.2.4 fmi1_callback_functions_t fmul_t::callBackFunctions

Definition at line 171 of file fmi_import_me_test.c.

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

- [Test/FMI1/fmi_import_me_test.c](#)

8.11 jm_callbacks Struct Reference

The callbacks struct is sent to all the modules in the library.

```
#include <jm_callbacks.h>
```

Data Fields

- [jm_malloc_f malloc](#)
Allocate non-initialized memory.
- [jm_calloc_f calloc](#)
Allocate zero initialized memory.
- [jm_realloc_f realloc](#)
Re-allocate memory.
- [jm_free_f free](#)
Free-allocated memory.
- [jm_logger_f logger](#)
Logging callback.
- [jm_log_level_enu_t log_level](#)
Logging level.
- [jm_voidp context](#)
Arbitrary context pointer passed to the logger function.
- [char errorMessageBuffer \[JM_MAX_ERROR_MESSAGE_SIZE\]](#)
The buffer used along with [jm_get_last_error\(\)](#)

8.11.1 Detailed Description

The callbacks struct is sent to all the modules in the library.

Definition at line 73 of file jm_callbacks.h.

8.11.2 Field Documentation

8.11.2.1 jm_malloc_f jm_callbacks::malloc

Allocate non-initialized memory.

Definition at line 75 of file jm_callbacks.h.

8.11.2.2 jm_calloc_f jm_callbacks::calloc

Allocate zero initialized memory.

Definition at line 77 of file jm_callbacks.h.

8.11.2.3 jm_realloc_f jm_callbacks::realloc

Re-allocate memory.

Definition at line 79 of file jm_callbacks.h.

8.11.2.4 jm_free_f jm_callbacks::free

Free-allocated memory.

Definition at line 81 of file jm_callbacks.h.

8.11.2.5 jm_logger_f jm_callbacks::logger

Logging callback.

Definition at line 83 of file jm_callbacks.h.

8.11.2.6 jm_log_level_enu_t jm_callbacks::log_level

Logging level.

Definition at line 85 of file jm_callbacks.h.

8.11.2.7 jm_voidp jm_callbacks::context

Arbitrary context pointer passed to the logger function.

Definition at line 87 of file `jm_callbacks.h`.

8.11.2.8 `char jm_callbacks::errMessageBuffer[JM_MAX_ERROR_MESSAGE_SIZE]`

The buffer used along with `jm_get_last_error()`

Definition at line 89 of file `jm_callbacks.h`.

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

- `src/Util/include/JM/jm_callbacks.h`

8.12 `jm_name_ID_map_t` Struct Reference

Mapping between a string and an integer ID.

```
#include <jm_types.h>
```

Data Fields

- `jm_string` `name`
- unsigned int `ID`

8.12.1 Detailed Description

Mapping between a string and an integer ID.

Definition at line 38 of file `jm_types.h`.

8.12.2 Field Documentation

8.12.2.1 `jm_string jm_name_ID_map_t::name`

Definition at line 39 of file `jm_types.h`.

8.12.2.2 unsigned int `jm_name_ID_map_t::ID`

Definition at line 40 of file `jm_types.h`.

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

- `src/Util/include/JM/jm_types.h`

8.13 jm_named_ptr Struct Reference

Name and object pointer pair.

```
#include <jm_named_ptr.h>
```

Data Fields

- [jm_voidp ptr](#)
- [jm_string name](#)

Object pointer.

8.13.1 Detailed Description

Name and object pointer pair.

Definition at line 39 of file `jm_named_ptr.h`.

8.13.2 Field Documentation

8.13.2.1 jm_voidp jm_named_ptr::ptr

Definition at line 40 of file `jm_named_ptr.h`.

8.13.2.2 jm_string jm_named_ptr::name

Object pointer.

Definition at line 41 of file `jm_named_ptr.h`.

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

- `src/Util/include/JM/jm_named_ptr.h`

Chapter 9

File Documentation

9.1 build/doc/fmilib.h File Reference

Include file to be used in client applications of the FMI Library.

```
#include <stddef.h> #include "fmilib_config.h" #include  
<FMI/fmi_import_context.h> #include <FMI1/fmil_import.-  
h> #include <FMI2/fmi2_import.h>
```

9.1.1 Detailed Description

Include file to be used in client applications of the FMI Library.

Definition in file [fmilib.h](#).

9.2 build/doc/fmilib_config.h File Reference

Library configuration file generated by the build system.

Defines

- #define [FMI_FILE_SEP](#) "/"
Platform folder separator.
- #define [FMI_DLL_EXT](#) ".so"
DLL file name suffix(.so, .dll, ...)
- #define [FMI_PLATFORM](#) "linux64"
Folder name inside FMU in which the DLL is found for this platform.
- #define [FMI_BINARIES](#) "binaries"
Folder name inside FMU where binaries are.
- #define [FMI_MODEL_DESCRIPTION_XML](#) "modelDescription.xml"

Model description XML file name.

- #define `FMILIB_EXPORT`

Used to declare the public API of the library needed for dynamic linking.

- #define `FMILIB_PRIVATE`

Intended for declaration of the private library functions.

- #define `FMILIB_SIZE_T_FORMAT` "%zu"
- #define `HAVE___VA_COPY`

Functions

- `FMILIB_EXPORT` `const char *` `fmilib_get_build_stamp` (void)

Get the library build stamp.

9.2.1 Detailed Description

Library configuration file generated by the build system.

Definition in file `fmilib_config.h`.

9.2.2 Define Documentation

9.2.2.1 #define `FMI_FILE_SEP` "/"

Platform folder separator.

Definition at line 48 of file `fmilib_config.h`.

9.2.2.2 #define `FMI_DLL_EXT` ".so"

DLL file name suffix(.so, .dll, ...)

Definition at line 51 of file `fmilib_config.h`.

9.2.2.3 #define `FMI_PLATFORM` "linux64"

Folder name inside FMU in which the DLL is found for this platform.

Definition at line 54 of file `fmilib_config.h`.

9.2.2.4 #define `FMI_BINARIES` "binaries"

Folder name inside FMU where binaries are.

Definition at line 57 of file `fmilib_config.h`.

9.2.2.5 `#define FMI_MODEL_DESCRIPTION_XML "modelDescription.xml"`

Model description XML file name.

Definition at line 59 of file `fmilib_config.h`.

9.2.2.6 `#define FMILIB_EXPORT`

Used to declare the public API of the library needed for dynamic linking.

9.2.2.7 `#define FMILIB_PRIVATE`

Intended for declaration of the private library functions.

9.2.2.8 `#define FMILIB_SIZET_FORMAT "%zu"`

Definition at line 125 of file `fmilib_config.h`.

9.2.2.9 `#define HAVE___VA_COPY`

Definition at line 156 of file `fmilib_config.h`.

9.2.3 Function Documentation

9.2.3.1 `FMILIB_EXPORT const char* fmilib_get_build_stamp ( void )`

Get the library build stamp.

Returns

A string giving SVN revision and build time for the library.

The function is only active if the library was configured with `#FMILIB_GENERATE_BUILD_STAMP ON`

9.3 build/doc/fmilib_mainpage.h File Reference

Autogenerated file with documentation.

9.3.1 Detailed Description

Autogenerated file with documentation.

Definition in file [fmilib_mainpage.h](#).

9.4 src/Import/include/FMI/fmi_import_context.h File Reference

Import context is the entry point to the library. It is used to initialize, unzip, get FMI version and start parsing.

```
#include <stddef.h> #include <fmilib_config.h> #include
<JM/jm_callbacks.h> #include <FMI2/fmi2_xml_callbacks.-
h> #include <FMI/fmi_version.h> #include <FMI1/fmi1-
_types.h> #include <FMI1/fmi1_enums.h> #include <FM-
I2/fmi2_types.h> #include <FMI2/fmi2_enums.h>
```

Typedefs

- typedef struct fmi_xml_context_t [fmi_import_context_t](#)
FMI version independent library context. Opaque struct returned from [fmi_import_allocate_context\(\)](#)
- typedef struct [fmi1_import_t](#) [fmi1_import_t](#)
FMU version 1.0 object.
- typedef struct [fmi2_import_t](#) [fmi2_import_t](#)
FMU version 2.0 object.

Functions

- [FMILIB_EXPORT](#) [fmi_import_context_t](#) * [fmi_import_allocate_context](#) ([jm_callbacks](#) *callbacks)
Create [fmi_import_context_t](#) structure.
- [FMILIB_EXPORT](#) void [fmi_import_free_context](#) ([fmi_import_context_t](#) *c)
Free memory allocated for the library context.
- [FMILIB_EXPORT](#) [fmi_version_enums_t](#) [fmi_import_get_fmi_version](#) ([fmi_import_context_t](#) *c, const char *fileName, const char *dirName)
Unzip an FMU specified by the fileName into directory dirName and parse XML to get FMI standard version.
- [FMILIB_EXPORT](#) [fmi1_import_t](#) * [fmi1_import_parse_xml](#) ([fmi_import_context_t](#) *c, const char *dirName)
Parse FMI 1.0 XML file found in the directory dirName.
- [FMILIB_EXPORT](#) [fmi2_import_t](#) * [fmi2_import_parse_xml](#) ([fmi_import_context_t](#) *context, const char *dirPath, [fmi2_xml_callbacks_t](#) *xml_callbacks)
Create [fmi2_import_t](#) structure and parse the FMI 2.0 XML file found in the directory dirName.

9.4.1 Detailed Description

Import context is the entry point to the library. It is used to initialize, unzip, get FMI version and start parsing.

Definition in file [fmi_import_context.h](#).

9.5 src/Import/include/FMI/fmi_import_util.h File Reference

```
#include <JM/jm_callbacks.h>
```

Functions

- **FMILIB_EXPORT** char * **fmi_import_mk_temp_dir** (jm_callbacks *cb, const char *systemTempDir, const char *tempPrefix)
Create a unique temporary directory.
- **FMILIB_EXPORT** jm_status_enu_t **fmi_import_rmdir** (jm_callbacks *cb, const char *dir)
Remove directory and all its contents.
- **FMILIB_EXPORT** char * **fmi_import_create_URL_from_abs_path** (jm_callbacks *cb, const char *absPath)
Create a file:// URL from absolute path.
- **FMILIB_EXPORT** char * **fmi_import_get_dll_path** (const char *fmu_unzipped_path, const char *model_identifier, jm_callbacks *callBackFunctions)
- **FMILIB_EXPORT** char * **fmi_import_get_model_description_path** (const char *fmu_unzipped_path, jm_callbacks *callBackFunctions)

9.6 src/Import/include/FMI1/fmi1_import.h File Reference

Public interface to the FMI import C-library.

```
#include <stddef.h> #include <fmilib_config.h> #include
<JM/jm_callbacks.h> #include <FMI/fmi_import_util.h>
#include <FMI/fmi_import_context.h> #include <FMI1/fmil-
_types.h> #include <FMI1/fmil_functions.h> #include <F-
MI1/fmil_enums.h> #include "fmil_import_type.h" #include
"fmil_import_unit.h" #include "fmil_import_variable.-
h" #include "fmil_import_vendor_annotations.h" #include
"fmil_import_capabilities.h" #include "fmil_import_variable-
_list.h" #include "fmil_import_capi.h" #include "fmil_-
import_convenience.h" #include "fmil_import_cosim.h"
```

Functions

- **FMILIB_EXPORT** fmi1_import_t * **fmi1_import_parse_xml** (fmi_import_context_t *context, const char *dirPath)
Create fmi1_import_t structure and parse the XML file.
- **FMILIB_EXPORT** const char * **fmi1_import_get_last_error** (fmi1_import_t *fmu)
Retrieve the last error message.
- **FMILIB_EXPORT** int **fmi1_import_clear_last_error** (fmi1_import_t *fmu)
Clear the error message.

- **FMILIB_EXPORT** void `fmi1_import_free` (`fmi1_import_t *fmu`)
Release the memory allocated.
- **FMILIB_EXPORT** const char * `fmi1_import_get_model_name` (`fmi1_import_t *fmu`)
Get model name.
- **FMILIB_EXPORT** const char * `fmi1_import_get_model_identifier` (`fmi1_import_t *fmu`)
Get model identifier.
- **FMILIB_EXPORT** const char * `fmi1_import_get_GUID` (`fmi1_import_t *fmu`)
Get FMU GUID.
- **FMILIB_EXPORT** const char * `fmi1_import_get_description` (`fmi1_import_t *fmu`)
Get FMU description.
- **FMILIB_EXPORT** const char * `fmi1_import_get_author` (`fmi1_import_t *fmu`)
Get FMU author.
- **FMILIB_EXPORT** const char * `fmi1_import_get_model_version` (`fmi1_import_t *fmu`)
Get FMU version.
- **FMILIB_EXPORT** const char * `fmi1_import_get_model_standard_version` (`fmi1_import_t *fmu`)
Get FMI standard version (always 1.0).
- **FMILIB_EXPORT** const char * `fmi1_import_get_generation_tool` (`fmi1_import_t *fmu`)
Get FMU generation tool.
- **FMILIB_EXPORT** const char * `fmi1_import_get_generation_date_and_time` (`fmi1_import_t *fmu`)
Get FMU generation date and time.
- **FMILIB_EXPORT** `fmi1_variable_naming_convention_enu_t` `fmi1_import_get_naming_convention` (`fmi1_import_t *fmu`)
Get variable naming convention used.
- **FMILIB_EXPORT** unsigned int `fmi1_import_get_number_of_continuous_states` (`fmi1_import_t *fmu`)
Get the number of continuous states.
- **FMILIB_EXPORT** unsigned int `fmi1_import_get_number_of_event_indicators` (`fmi1_import_t *fmu`)
Get the number of event indicators.
- **FMILIB_EXPORT** double `fmi1_import_get_default_experiment_start` (`fmi1_import_t *fmu`)
Get the start time for default experiment as specified in the XML file.
- **FMILIB_EXPORT** double `fmi1_import_get_default_experiment_stop` (`fmi1_import_t *fmu`)
Get the stop time for default experiment as specified in the XML file.
- **FMILIB_EXPORT** double `fmi1_import_get_default_experiment_tolerance` (`fmi1_import_t *fmu`)
Get the tolerance default experiment as specified in the XML file.

- `FMILIB_EXPORT fmi1_fmu_kind_enu_t fmi1_import_get_fmu_kind (fmi1_import_t *fmu)`

Get the type of the FMU (model exchange or co-simulation)

- `FMILIB_EXPORT fmi1_import_capabilities_t * fmi1_import_get_capabilities (fmi1_import_t *fmu)`

Get the structure with capability flags.

- `FMILIB_EXPORT fmi1_import_type_definitions_t * fmi1_import_get_type_definitions (fmi1_import_t *)`

Get the list of all the type definitions in the model.

- `FMILIB_EXPORT fmi1_import_unit_definitions_t * fmi1_import_get_unit_definitions (fmi1_import_t *fmu)`

Get a list of all the unit definitions in the model.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_direct_dependency (fmi1_import_t *fmu, fmi1_import_variable_t *)`

Get the direct dependency information.

- `FMILIB_EXPORT fmi1_import_variable_t * fmi1_import_get_variable_alias_base (fmi1_import_t *fmu, fmi1_import_variable_t *)`

Get the variable with the same value reference that is not an alias.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_variable_aliases (fmi1_import_t *fmu, fmi1_import_variable_t *)`

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_variable_list (fmi1_import_t *fmu)`

Get the list of all the variables in the model.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_variable_list_alphabetical_order (fmi1_import_t *fmu)`

Get the list of all the variables in the model in alphabetical order.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_create_var_list (fmi1_import_t *fmu, fmi1_import_variable_t *v)`

Create a variable list with a single variable.

9.6.1 Detailed Description

Public interface to the FMI import C-library.

Definition in file [fmi1_import.h](#).

9.7 src/Import/include/FMI1/fmi1_import_capabilities.h File Reference

```
#include <fmilib_config.h>
```

Typedefs

FMU capabilities flags

- typedef struct fmi1_xml_capabilities_t fmi1_import_capabilities_t
A container for all the capability flags.

Functions

- FMILIB_EXPORT int fmi1_import_get_canHandleVariableCommunicationStepSize (fmi1_import_capabilities_t *)
Retrieve canHandleVariableCommunicationStepSize flag.
- FMILIB_EXPORT int fmi1_import_get_canHandleEvents (fmi1_import_capabilities_t *)
Retrieve canHandleEvents flag.
- FMILIB_EXPORT int fmi1_import_get_canRejectSteps (fmi1_import_capabilities_t *)
Retrieve canRejectSteps flag.
- FMILIB_EXPORT int fmi1_import_get_canInterpolateInputs (fmi1_import_capabilities_t *)
Retrieve canInterpolateInputs flag.
- FMILIB_EXPORT unsigned int fmi1_import_get_maxOutputDerivativeOrder (fmi1_import_capabilities_t *)
Retrieve maxOutputDerivativeOrder.
- FMILIB_EXPORT int fmi1_import_get_canRunAsynchronously (fmi1_import_capabilities_t *)
Retrieve canRunAsynchronously flag.
- FMILIB_EXPORT int fmi1_import_get_canSignalEvents (fmi1_import_capabilities_t *)
Retrieve canSignalEvents flag.
- FMILIB_EXPORT int fmi1_import_get_canBeInstantiatedOnlyOncePerProcess (fmi1_import_capabilities_t *)
Retrieve canBeInstantiatedOnlyOncePerProcess flag.
- FMILIB_EXPORT int fmi1_import_get_canNotUseMemoryManagementFunctions (fmi1_import_capabilities_t *)
Retrieve canNotUseMemoryManagementFunctions flag.

9.7.1 Detailed Description

Functions to retrieve capability flags.

Definition in file [fmi1_import_capabilities.h](#).

9.8 src/Import/include/FMI1/fmi1_import_capi.h File Reference

```
#include <JM/jm_callbacks.h>      #include <FMI/fmi_import-
_util.h>      #include <FMI/fmi_import_context.h>      #include
<FMI1/fmi1_types.h>      #include <FMI1/fmi1_functions.h> ×
#include <FMI1/fmi1_enums.h>
```

Functions

- **FMILIB_EXPORT** `jm_status_enu_t fmi1_import_create_dllfmu (fmi1_import_t *fmu, fmi1_callback_functions_t callBackFunctions, int registerGlobally)`
Create a C-API struct. The C-API struct is a placeholder for the FMI DLL functions.
- **FMILIB_EXPORT** `void fmi1_import_destroy_dllfmu (fmi1_import_t *fmu)`
Free a C-API struct. All memory allocated since the struct was created is freed.
- **FMILIB_EXPORT** `void fmi1_import_set_debug_mode (fmi1_import_t *fmu, int mode)`
Set CAPI debug mode flag. Setting to non-zero prevents DLL unloading in fmi1_import_destroy_dllfmu while all the memory is deallocated. This is to support valgrind debugging.
- **FMILIB_EXPORT** `const char * fmi1_import_get_version (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiGetVersion()
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_debug_logging (fmi1_import_t *fmu, fmi1_boolean_t loggingOn)`
Wrapper for the FMI function fmiSetDebugLogging(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_real (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_real_t value[])`
Wrapper for the FMI function fmiSetReal(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_integer (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t value[])`
Wrapper for the FMI function fmiSetInteger(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_boolean (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_boolean_t value[])`
Wrapper for the FMI function fmiSetBoolean(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_string (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_string_t value[])`
Wrapper for the FMI function fmiSetString(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_real (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, fmi1_real_t value[])`
Wrapper for the FMI function fmiGetReal(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_integer (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, fmi1_integer_t value[])`
Wrapper for the FMI function fmiGetInteger(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_boolean (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, fmi1_boolean_t value[])`
Wrapper for the FMI function fmiGetBoolean(...)

- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_string (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, fmi1_string_t value[])`
Wrapper for the FMI function fmiGetString(...)
- `FMILIB_EXPORT const char * fmi1_import_get_model_types_platform (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiGetModelTypesPlatform(...)
- `FMILIB_EXPORT jm_status_enumeration_t fmi1_import_instantiate_model (fmi1_import_t *fmu, fmi1_string_t instanceName)`
Wrapper for the FMI function fmiInstantiateModel(...)
- `FMILIB_EXPORT void fmi1_import_free_model_instance (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiFreeModelInstance(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_set_time (fmi1_import_t *fmu, fmi1_real_t time)`
Wrapper for the FMI function fmiSetTime(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_set_continuous_states (fmi1_import_t *fmu, const fmi1_real_t x[], size_t nx)`
Wrapper for the FMI function fmiSetContinuousStates(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_completed_integrator_step (fmi1_import_t *fmu, fmi1_boolean_t *callEventUpdate)`
Wrapper for the FMI function fmiCompletedIntegratorStep(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_initialize (fmi1_import_t *fmu, fmi1_boolean_t toleranceControlled, fmi1_real_t relativeTolerance, fmi1_event_info_t *eventInfo)`
Wrapper for the FMI function fmiInitialize(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_derivatives (fmi1_import_t *fmu, fmi1_real_t derivatives[], size_t nx)`
Wrapper for the FMI function fmiGetDerivatives(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_event_indicators (fmi1_import_t *fmu, fmi1_real_t eventIndicators[], size_t ni)`
Wrapper for the FMI function fmiGetEventIndicators(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_eventUpdate (fmi1_import_t *fmu, fmi1_boolean_t intermediateResults, fmi1_event_info_t *eventInfo)`
Wrapper for the FMI function fmiEventUpdate(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_continuous_states (fmi1_import_t *fmu, fmi1_real_t states[], size_t nx)`
Wrapper for the FMI function fmiGetContinuousStates(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_nominal_continuous_states (fmi1_import_t *fmu, fmi1_real_t x_nominal[], size_t nx)`
Wrapper for the FMI function fmiGetNominalContinuousStates(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_state_value_references (fmi1_import_t *fmu, fmi1_value_reference_t vrx[], size_t nx)`
Wrapper for the FMI function fmiGetStateValueReferences(...)
- `FMILIB_EXPORT fmi1_status_t fmi1_import_terminate (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiTerminate(...)

- **FMILIB_EXPORT** `const char * fmi1_import_get_types_platform (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiGetTypesPlatform(...)
- **FMILIB_EXPORT** `jm_status_enu_t fmi1_import_instantiate_slave (fmi1_import_t *fmu, fmi1_string_t instanceName, fmi1_string_t fmuLocation, fmi1_string_t mimeType, fmi1_real_t timeout, fmi1_boolean_t visible, fmi1_boolean_t interactive)`
Wrapper for the FMI function fmiInstantiateSlave(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_initialize_slave (fmi1_import_t *fmu, fmi1_real_t tStart, fmi1_boolean_t StopTimeDefined, fmi1_real_t tStop)`
Wrapper for the FMI function fmiInitializeSlave(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_terminate_slave (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiTerminateSlave(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_reset_slave (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiResetSlave(...)
- **FMILIB_EXPORT** `void fmi1_import_free_slave_instance (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiFreeSlaveInstance(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_set_real_input_derivatives (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t order[], const fmi1_real_t value[])`
Wrapper for the FMI function fmiSetRealInputDerivatives(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_real_output_derivatives (fmi1_import_t *fmu, const fmi1_value_reference_t vr[], size_t nvr, const fmi1_integer_t order[], fmi1_real_t value[])`
Wrapper for the FMI function fmiGetOutputDerivatives(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_cancel_step (fmi1_import_t *fmu)`
Wrapper for the FMI function fmiCancelStep(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_do_step (fmi1_import_t *fmu, fmi1_real_t currentCommunicationPoint, fmi1_real_t communicationStepSize, fmi1_boolean_t newStep)`
Wrapper for the FMI function fmiDoStep(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_status (fmi1_import_t *fmu, const fmi1_status_kind_t s, fmi1_status_t *value)`
Wrapper for the FMI function fmiGetStatus(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_real_status (fmi1_import_t *fmu, const fmi1_status_kind_t s, fmi1_real_t *value)`
Wrapper for the FMI function fmiGetRealStatus(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_integer_status (fmi1_import_t *fmu, const fmi1_status_kind_t s, fmi1_integer_t *value)`
Wrapper for the FMI function fmiGetIntegerStatus(...)
- **FMILIB_EXPORT** `fmi1_status_t fmi1_import_get_boolean_status (fmi1_import_t *fmu, const fmi1_status_kind_t s, fmi1_boolean_t *value)`
Wrapper for the FMI function fmiGetBooleanStatus(...)

- `FMILIB_EXPORT fmi1_status_t fmi1_import_get_string_status (fmi1_import_t *fmu, const fmi1_status_kind_t s, fmi1_string_t *value)`
Wrapper for the FMI function `fmiGetStringStatus(...)`

9.8.1 Detailed Description

Wrapper functions for the FMI 1.0 functions

Definition in file `fmi1_import_capi.h`.

9.9 src/Import/include/FMI1/fmi1_import_convenience.h File - Reference

Public interface to the FMI import C-library. Convenience functions.

```
#include <FMI/fmi_import_context.h>
```

Data Structures

- struct `fmi1_import_model_counts_t`
Collection of counters providing model information.

Functions

- `FMILIB_EXPORT void fmi1_import_collect_model_counts (fmi1_import_t *fmu, fmi1_import_model_counts_t *counts)`
Collect model information by counting the number of variables with specific properties and fillin in `fmi1_import_model_counts_t` struct.
- `FMILIB_EXPORT void fmi1_import_expand_variable_references (fmi1_import_t *fmu, const char *msgIn, char *msgOut, size_t maxMsgSize)`
Print `msgIn` into `msgOut` by expanding variable references of the form `#<Type><V-R>#` into variable names and replacing `##` with a single `#`.
- `FMILIB_EXPORT void fmi1_log_forwarding (fmi1_component_t c, fmi1_string_t instanceName, fmi1_status_t status, fmi1_string_t category, fmi1_string_t message,...)`
An implementation of FMI 1.0 logger that forwards the messages to logger function inside `jm_callbacks` structure.
- `FMILIB_EXPORT void fmi1_log_forwarding_v (fmi1_component_t c, fmi1_string_t instanceName, fmi1_status_t status, fmi1_string_t category, fmi1_string_t message, va_list args)`
An implementation of FMI 1.0 logger that forwards the messages to logger function inside `jm_callbacks` structure.
- `FMILIB_EXPORT void fmi1_default_callback_logger (fmi1_component_t c, fmi1_string_t instanceName, fmi1_status_t status, fmi1_string_t category, fmi1_string_t message,...)`

Default FMI 1.0 logger may be used when instantiating FMUs.

- **FMILIB_EXPORT** void `fmi1_import_init_logger` (`jm_callbacks *cb`, `fmi1_callback_functions_t *fmiCallbacks`)

Given `fmi1_callback_functions_t` logger (`fmi1_logger`), the `jm_callbacks` logger may be setup to redirect the messages to the `fmi1_logger`.

9.9.1 Detailed Description

Public interface to the FMI import C-library. Convenience functions. The functions in this file are provided for convenience. The functionality is already available via other lower level functions.

Definition in file `fmi1_import_convenience.h`.

9.10 src/Import/include/FMI1/fmi1_import_cosim.h File Reference

```
#include <FMI/fmi_import_context.h>
```

Functions

- **FMILIB_EXPORT** const char * `fmi1_import_get_entry_point` (`fmi1_import_t *`)
Get the entry point.
- **FMILIB_EXPORT** const char * `fmi1_import_get_mime_type` (`fmi1_import_t *`)
Get mime type.
- **FMILIB_EXPORT** int `fmi1_import_get_manual_start` (`fmi1_import_t *`)
Get manual start flag.
- **FMILIB_EXPORT** size_t `fmi1_import_get_number_of_additional_models` (`fmi1_import_t *fmu`)
Get the number of additional models specified.
- **FMILIB_EXPORT** const char * `fmi1_import_get_additional_model_name` (`fmi1_import_t *fmu`, size_t index)
Get the name of an additional model.

9.10.1 Detailed Description

Functions to retrieve co-simulation related information.

Definition in file `fmi1_import_cosim.h`.

9.11 src/Import/include/FMI1/fmi1_import_type.h File Reference

Public interface to the FMI XML C-library: variable types handling.

```
#include "fmi1_import_unit.h"
```

Typedefs

Type definitions supporting structures

- typedef struct fmi1_xml_real_typedef_t fmi1_import_real_typedef_t
Opaque type definition object.
- typedef struct fmi1_xml_integer_typedef_t fmi1_import_integer_typedef_t
Opaque integer type definition object.
- typedef struct fmi1_xml_enumeration_typedef_t fmi1_import_enumeration_ -
typedef_t
Opaque enumeration type definition object.
- typedef struct fmi1_xml_variable_typedef_t fmi1_import_variable_typedef_t
Opaque general variable type definition object.
- typedef struct fmi1_xml_type_definitions_t fmi1_import_type_definitions_t
Opaque list of the type definitions in the model.

Functions

- FMILIB_EXPORT size_t fmi1_import_get_type_definition_number (fmi1_import_ -
type_definitions_t *td)
Get the number of available type definitions.
- FMILIB_EXPORT fmi1_import_variable_typedef_t * fmi1_import_get_typedef
(fmi1_import_type_definitions_t *td, unsigned int index)
Get a type definition specified by the index.
- FMILIB_EXPORT fmi1_import_display_unit_t * fmi1_import_get_type_display_ -
unit (fmi1_import_real_typedef_t *)
Get associated display unit for a type definition if any.
- FMILIB_EXPORT const char * fmi1_import_get_type_name (fmi1_import_ -
variable_typedef_t *)
Get the type name.
- FMILIB_EXPORT const char * fmi1_import_get_type_description (fmi1_import_ -
variable_typedef_t *)
Get type description.
- FMILIB_EXPORT fmi1_base_type_enumeration_t fmi1_import_get_base_type (fmi1_ -
import_variable_typedef_t *)
Get base type used for the type definition.
- FMILIB_EXPORT fmi1_import_real_typedef_t * fmi1_import_get_type_as_real
(fmi1_import_variable_typedef_t *)
Cast the general type definition object to an object with a specific base type.
- FMILIB_EXPORT fmi1_import_integer_typedef_t * fmi1_import_get_type_as_ -
int (fmi1_import_variable_typedef_t *)
Cast the general type definition object to an object with a specific base type.
- FMILIB_EXPORT fmi1_import_enumeration_typedef_t * fmi1_import_get_type_ -
as_enum (fmi1_import_variable_typedef_t *)
Cast the general type definition object to an object with a specific base type.
- FMILIB_EXPORT const char * fmi1_import_get_type_quantity (fmi1_import_ -
variable_typedef_t *)

Get the quantity associated with the type definition.

- `FMILIB_EXPORT double fmi1_import_get_real_type_min (fmi1_import_real_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT double fmi1_import_get_real_type_max (fmi1_import_real_typedef_t *)`

Get maximum value for the type.

- `FMILIB_EXPORT double fmi1_import_get_real_type_nominal (fmi1_import_real_typedef_t *)`

Get the nominal value associated with the type definition.

- `FMILIB_EXPORT fmi1_import_unit_t * fmi1_import_get_real_type_unit (fmi1_import_real_typedef_t *)`

Get the unit object associated with the type definition if any.

- `FMILIB_EXPORT int fmi1_import_get_real_type_is_relative_quantity (fmi1_import_real_typedef_t *)`

Get the relativeQuantity flag.

- `FMILIB_EXPORT int fmi1_import_get_integer_type_min (fmi1_import_integer_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT int fmi1_import_get_integer_type_max (fmi1_import_integer_typedef_t *)`

Get maximum value for the type.

- `FMILIB_EXPORT unsigned int fmi1_import_get_enum_type_min (fmi1_import_enumeration_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT unsigned int fmi1_import_get_enum_type_max (fmi1_import_enumeration_typedef_t *)`

Get maximum value for the type.

- `FMILIB_EXPORT unsigned int fmi1_import_get_enum_type_size (fmi1_import_enumeration_typedef_t *)`

Get the number of elements in the enum.

- `FMILIB_EXPORT const char * fmi1_import_get_enum_type_item_name (fmi1_import_enumeration_typedef_t *, unsigned int item)`

Get an enumeration item name by index.

- `FMILIB_EXPORT const char * fmi1_import_get_enum_type_item_description (fmi1_import_enumeration_typedef_t *, unsigned int item)`

Get an enumeration item description by index.

9.11.1 Detailed Description

Public interface to the FMI XML C-library: variable types handling.

Definition in file `fmi1_import_type.h`.

9.12 src/Import/include/FMI1/fmi1_import_unit.h File Reference

Public interface to the FMI import C-library. Handling of variable units.

```
#include <fmilib_config.h>
```

Typedefs

Structures encapsulating unit information

- typedef struct fmi1_xml_unit_t [fmi1_import_unit_t](#)
A variable unit defined with a unit definition.
- typedef struct fmi1_xml_display_unit_t [fmi1_import_display_unit_t](#)
A display unit.
- typedef struct fmi1_xml_unit_definitions_t [fmi1_import_unit_definitions_t](#)
The list of all the unit definitions in the model.

Functions

- [FMILIB_EXPORT](#) unsigned int [fmi1_import_get_unit_definitions_number](#) ([fmi1_import_unit_definitions_t](#) *)
Get the number of unit definitions.
- [FMILIB_EXPORT](#) [fmi1_import_unit_t](#) * [fmi1_import_get_unit](#) ([fmi1_import_unit_definitions_t](#) *, unsigned int index)
Get a unit definition.
- [FMILIB_EXPORT](#) const char * [fmi1_import_get_unit_name](#) ([fmi1_import_unit_t](#) *)
Get a unit name.
- [FMILIB_EXPORT](#) unsigned int [fmi1_import_get_unit_display_unit_number](#) ([fmi1_import_unit_t](#) *)
Get the number of display units associated with this unit.
- [FMILIB_EXPORT](#) [fmi1_import_display_unit_t](#) * [fmi1_import_get_unit_display_unit](#) ([fmi1_import_unit_t](#) *, [size_t](#) index)
Get a display unit object by index.
- [FMILIB_EXPORT](#) [fmi1_import_unit_t](#) * [fmi1_import_get_base_unit](#) ([fmi1_import_display_unit_t](#) *)
Get unit definition for a display unit.
- [FMILIB_EXPORT](#) const char * [fmi1_import_get_display_unit_name](#) ([fmi1_import_display_unit_t](#) *)
Get display unit name.
- [FMILIB_EXPORT](#) [fmi1_real_t](#) [fmi1_import_get_display_unit_gain](#) ([fmi1_import_display_unit_t](#) *)
Get the "gain" associated with the display unit.
- [FMILIB_EXPORT](#) [fmi1_real_t](#) [fmi1_import_get_display_unit_offset](#) ([fmi1_import_display_unit_t](#) *)
Get the "offset" associated with the display unit.

- **FMILIB_EXPORT** fmi1_real_t fmi1_import_convert_to_display_unit (fmi1_real_t value, fmi1_import_display_unit_t *du, int isRelativeQuantity)
Convert a value measured in "units" to a value measured with "display units".
- **FMILIB_EXPORT** fmi1_real_t fmi1_import_convert_from_display_unit (fmi1_real_t value, fmi1_import_display_unit_t *du, int isRelativeQuantity)
Convert a value measured in "display units" to a value measured with "units".

9.12.1 Detailed Description

Public interface to the FMI import C-library. Handling of variable units.

Definition in file [fmi1_import_unit.h](#).

9.13 src/Import/include/FMI1/fmi1_import_variable.h File Reference

Public interface to the FMI import C-library. Handling of model variables.

```
#include <FMI/fmi_import_context.h> #include "fmil_import-  
_type.h" #include "fmil_import_unit.h"
```

Typedefs

Scalar variable types

- typedef struct fmi1_xml_variable_t [fmi1_import_variable_t](#)
General variable type.
- typedef struct fmi1_xml_real_variable_t [fmi1_import_real_variable_t](#)
Opaque real variable.
- typedef struct fmi1_xml_integer_variable_t [fmi1_import_integer_variable_t](#)
Opaque integer variable.
- typedef struct fmi1_xml_string_variable_t [fmi1_import_string_variable_t](#)
Opaque string variable.
- typedef struct fmi1_xml_enum_variable_t [fmi1_import_enum_variable_t](#)
Opaque enumeration variable.
- typedef struct fmi1_xml_bool_variable_t [fmi1_import_bool_variable_t](#)
Opaque boolean variable.
- typedef struct [fmi1_import_variable_list_t](#) fmi1_import_variable_list_t
List of variables.

Functions

- **FMILIB_EXPORT** [fmi1_import_variable_t](#)* fmi1_import_get_variable_by_name ([fmi1_import_t](#)* fmu, const char *name)
Get variable by variable name.

- `FMILIB_EXPORT fmi1_import_variable_t * fmi1_import_get_variable_by_vr (fmi1_import_t *fmu, fmi1_base_type_enumeration_t baseType, fmi1_value_reference_t vr)`
Get variable by value reference.
- `FMILIB_EXPORT const char * fmi1_import_get_variable_name (fmi1_import_variable_t *)`
Get the variable name.
- `FMILIB_EXPORT const char * fmi1_import_get_variable_description (fmi1_import_variable_t *)`
Get variable description.
- `FMILIB_EXPORT fmi1_value_reference_t fmi1_import_get_variable_vr (fmi1_import_variable_t *)`
Get variable value reference.
- `FMILIB_EXPORT fmi1_import_variable_typedef_t * fmi1_import_get_variable_declared_type (fmi1_import_variable_t *)`
For scalar variable gives the type definition is present.
- `FMILIB_EXPORT fmi1_base_type_enumeration_t fmi1_import_get_variable_base_type (fmi1_import_variable_t *)`
Get variable base type.
- `FMILIB_EXPORT int fmi1_import_get_variable_has_start (fmi1_import_variable_t *)`
Check if the variable has "start" attribute.
- `FMILIB_EXPORT int fmi1_import_get_variable_is_fixed (fmi1_import_variable_t *)`
Get the variable "fixed" attribute.
- `FMILIB_EXPORT fmi1_variability_enumeration_t fmi1_import_get_variability (fmi1_import_variable_t *)`
Get variability attribute.
- `FMILIB_EXPORT fmi1_causality_enumeration_t fmi1_import_get_causality (fmi1_import_variable_t *)`
Get causality attribute.
- `FMILIB_EXPORT fmi1_import_real_variable_t * fmi1_import_get_variable_as_real (fmi1_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi1_import_integer_variable_t * fmi1_import_get_variable_as_integer (fmi1_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi1_import_enum_variable_t * fmi1_import_get_variable_as_enum (fmi1_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi1_import_string_variable_t * fmi1_import_get_variable_as_string (fmi1_import_variable_t *)`
Cast general variable to a one with the specific type.
- `FMILIB_EXPORT fmi1_import_bool_variable_t * fmi1_import_get_variable_as_boolean (fmi1_import_variable_t *)`

Cast general variable to a one with the specific type.

- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_start (fmi1_import_real_variable_t *v)`

Get the variable start attribute.

- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_max (fmi1_import_real_variable_t *v)`

Get maximum value for the variable.

- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_min (fmi1_import_real_variable_t *v)`

Get minimal value for the variable.

- `FMILIB_EXPORT fmi1_real_t fmi1_import_get_real_variable_nominal (fmi1_import_real_variable_t *v)`

Get nominal value for the variable.

- `FMILIB_EXPORT fmi1_import_unit_t * fmi1_import_get_real_variable_unit (fmi1_import_real_variable_t *v)`

Get associated "unit" object if any.

- `FMILIB_EXPORT fmi1_import_display_unit_t * fmi1_import_get_real_variable_display_unit (fmi1_import_real_variable_t *v)`

Get associated "display unit" object if any.

- `FMILIB_EXPORT const char * fmi1_import_get_string_variable_start (fmi1_import_string_variable_t *v)`

Get start value for the variable.

- `FMILIB_EXPORT fmi1_boolean_t fmi1_import_get_boolean_variable_start (fmi1_import_bool_variable_t *v)`

Get start value for the variable.

- `FMILIB_EXPORT int fmi1_import_get_integer_variable_start (fmi1_import_integer_variable_t *v)`

Get start value for the variable.

- `FMILIB_EXPORT int fmi1_import_get_integer_variable_min (fmi1_import_integer_variable_t *v)`

Get minimal value for the variable.

- `FMILIB_EXPORT int fmi1_import_get_integer_variable_max (fmi1_import_integer_variable_t *v)`

Get max value for the variable.

- `FMILIB_EXPORT int fmi1_import_get_enum_variable_start (fmi1_import_enum_variable_t *v)`

Get start value for the variable.

- `FMILIB_EXPORT int fmi1_import_get_enum_variable_min (fmi1_import_enum_variable_t *v)`

Get minimal value for the variable.

- `FMILIB_EXPORT int fmi1_import_get_enum_variable_max (fmi1_import_enum_variable_t *v)`

Get max value for the variable.

- `FMILIB_EXPORT fmi1_variable_alias_kind_enu_t fmi1_import_get_variable_alias_kind (fmi1_import_variable_t *)`

Get the variable alias kind.

- `size_t fmi1_import_get_variable_original_order (fmi1_import_variable_t *v)`

Get the original index in xml of the variable.

9.13.1 Detailed Description

Public interface to the FMI import C-library. Handling of model variables.

Definition in file [fmi1_import_variable.h](#).

9.14 `src/Import/include/FMI1/fmi1_import_variable_list.h` File - Reference

Public interface to the FMI XML C-library. Handling of variable lists.

```
#include "fmi1_import_variable.h"
```

Functions

- `fmi1_import_variable_list_t * fmi1_import_alloc_variable_list (fmi1_import_t *fmu, size_t size)`
- `FMILIB_EXPORT void fmi1_import_free_variable_list (fmi1_import_variable_list_t *vl)`

Free a variable list. Note that variable lists are allocated dynamically and must be freed when not needed any longer.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_clone_variable_list (fmi1_import_variable_list_t *vl)`

Make a copy of the list.

- `FMILIB_EXPORT size_t fmi1_import_get_variable_list_size (fmi1_import_variable_list_t *vl)`

Get number of variables in a list.

- `FMILIB_EXPORT const fmi1_value_reference_t * fmi1_import_get_value_references_list (fmi1_import_variable_list_t *vl)`

Get a pointer to the list of the value references for all the variables.

- `FMILIB_EXPORT fmi1_import_variable_t * fmi1_import_get_variable (fmi1_import_variable_list_t *vl, unsigned int index)`

Get a single variable from the list.

Operations on variable lists. Every operation creates a new list.

- `typedef int(* fmi1_import_variable_filter_function_ft)(fmi1_import_variable_t *vl, void *data)`

Callback function typedef for the `fmiFilterVariables`.

9.15 src/Import/include/FMI1/fmi1_import_vendor_annotations.h File Reference 287

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_get_sublist (fmi1_import_variable_list_t *vl, unsigned int fromIndex, unsigned int toIndex)`

Select sub-lists.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_filter_variables (fmi1_import_variable_list_t *vl, fmi1_import_variable_filter_function_ft filter, void *context)`

Call the provided 'filter' function on every variable in the list and create a new list.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_join_var_list (fmi1_import_variable_list_t *a, fmi1_import_variable_list_t *b)`

Create a new variable list by concatenating two lists.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_append_to_var_list (fmi1_import_variable_list_t *vl, fmi1_import_variable_t *v)`

Append a variable to the variable list.

- `FMILIB_EXPORT fmi1_import_variable_list_t * fmi1_import_prepend_to_var_list (fmi1_import_variable_list_t *vl, fmi1_import_variable_t *v)`

Prepend a variable to the variable list.

- `FMILIB_EXPORT jm_status_enu_t fmi1_import_var_list_push_back (fmi1_import_variable_list_t *vl, fmi1_import_variable_t *v)`

Add a variable to a variable list.

9.14.1 Detailed Description

Public interface to the FMI XML C-library. Handling of variable lists.

Definition in file [fmi1_import_variable_list.h](#).

9.15 src/Import/include/FMI1/fmi1_import_vendor_annotations.h - File Reference

Public interface to the FMI XML C-library. Handling of vendor annotations.

```
#include <fmilib_config.h>
```

Typedefs

Vendor annotation supporting structures

- `typedef struct fmi1_xml_vendor_list_t fmi1_import_vendor_list_t`
Opaque list of vendor annotations.
- `typedef struct fmi1_xml_vendor_t fmi1_import_vendor_t`
Opaque vendor object.
- `typedef struct fmi1_xml_annotation_t fmi1_import_annotation_t`
Opaque annotation object.

Functions

- `FMILIB_EXPORT unsigned int fmi1_import_get_number_of_vendors (fmi1_import_vendor_list_t *)`

Get the number of different vendors.

- `FMILIB_EXPORT fmi1_import_vendor_t * fmi1_import_get_vendor (fmi1_import_vendor_list_t *, unsigned int index)`

Get the annotations associated with vendor specified by the index.

- `FMILIB_EXPORT const char * fmi1_import_get_vendor_name (fmi1_import_vendor_t *)`

Get the vendor name.

- `FMILIB_EXPORT unsigned int fmi1_import_get_number_of_vendor_annotations (fmi1_import_vendor_t *)`

Get the number of annotations provided for the vendor.

- `FMILIB_EXPORT fmi1_import_annotation_t * fmi1_import_get_vendor_annotation (fmi1_import_vendor_t *, unsigned int index)`

Get an annotation object for the vendor by index.

- `FMILIB_EXPORT const char * fmi1_import_get_annotation_name (fmi1_import_annotation_t *)`

Get the name of the annotation.

- `FMILIB_EXPORT const char * fmi1_import_get_annotation_value (fmi1_import_annotation_t *)`

Get the value for the annotation.

- `FMILIB_EXPORT fmi1_import_vendor_list_t * fmi1_import_get_vendor_list (fmi1_import_t *fmu)`

Get the list of all the vendor annotations present in the XML file.

9.15.1 Detailed Description

Public interface to the FMI XML C-library. Handling of vendor annotations.

Definition in file [fmi1_import_vendor_annotations.h](#).

9.16 src/Import/include/FMI2/fmi2_import.h File Reference

Public interface to the FMI import C-library.

```
#include <stddef.h> #include <fmilib_config.h> #include
<JM/jm_callbacks.h> #include <FMI/fmi_import_util.h> ×
#include <FMI/fmi_import_context.h> #include <FMI2/fmi2-
_types.h> #include <FMI2/fmi2_functions.h> #include <F-
MI2/fmi2_enums.h> #include "fmi2_import_type.h" #include
"fmi2_import_unit.h" #include "fmi2_import_variable.h" ×
#include "fmi2_import_variable_list.h" #include "fmi2_-
import_capi.h" #include "fmi2_import_convenience.h"
```

Functions

- **FMILIB_EXPORT** const char * fmi2_import_get_last_error (fmi2_import_t *fmu)
Retrieve the last error message.
- **FMILIB_EXPORT** int fmi2_import_clear_last_error (fmi2_import_t *fmu)
Clear the error message.
- **FMILIB_EXPORT** void fmi2_import_free (fmi2_import_t *fmu)
Release the memory allocated.
- **FMILIB_EXPORT** const char * fmi2_import_get_model_name (fmi2_import_t *fmu)
Get model name.
- **FMILIB_EXPORT** unsigned int fmi2_import_get_capability (fmi2_import_t *, fmi2_capabilities_enu_t id)
Retrieve capability flags by ID.
- **FMILIB_EXPORT** const char * fmi2_import_get_model_identifier_ME (fmi2_import_t *fmu)
Get model identifier for ModelExchange.
- **FMILIB_EXPORT** const char * fmi2_import_get_model_identifier_CS (fmi2_import_t *fmu)
Get model identifier for CoSimulation.
- **FMILIB_EXPORT** const char * fmi2_import_get_GUID (fmi2_import_t *fmu)
Get FMU GUID.
- **FMILIB_EXPORT** const char * fmi2_import_get_description (fmi2_import_t *fmu)
Get FMU description.
- **FMILIB_EXPORT** const char * fmi2_import_get_author (fmi2_import_t *fmu)
Get FMU author.
- **FMILIB_EXPORT** const char * fmi2_import_get_copyright (fmi2_import_t *fmu)
Get FMU copyright information.
- **FMILIB_EXPORT** const char * fmi2_import_get_license (fmi2_import_t *fmu)
Get FMU license information.
- **FMILIB_EXPORT** const char * fmi2_import_get_model_version (fmi2_import_t *fmu)
Get FMU version.
- **FMILIB_EXPORT** const char * fmi2_import_get_model_standard_version (fmi2_import_t *fmu)
Get FMI standard version (always 2.0).
- **FMILIB_EXPORT** const char * fmi2_import_get_generation_tool (fmi2_import_t *fmu)
Get FMU generation tool.
- **FMILIB_EXPORT** const char * fmi2_import_get_generation_date_and_time (fmi2_import_t *fmu)
Get FMU generation date and time.

- FMILIB_EXPORT fmi2_variable_naming_convension_enu_t fmi2_import_get_naming_convention (fmi2_import_t *fmu)
Get variable naming convention used.
- FMILIB_EXPORT size_t fmi2_import_get_number_of_continuous_states (fmi2_import_t *fmu)
Get the number of continuous states.
- FMILIB_EXPORT size_t fmi2_import_get_number_of_event_indicators (fmi2_import_t *fmu)
Get the number of event indicators.
- FMILIB_EXPORT double fmi2_import_get_default_experiment_start (fmi2_import_t *fmu)
Get the start time for default experiment as specified in the XML file.
- FMILIB_EXPORT double fmi2_import_get_default_experiment_stop (fmi2_import_t *fmu)
Get the stop time for default experiment as specified in the XML file.
- FMILIB_EXPORT double fmi2_import_get_default_experiment_tolerance (fmi2_import_t *fmu)
Get the tolerance for default experiment as specified in the XML file.
- FMILIB_EXPORT double fmi2_import_get_default_experiment_step (fmi2_import_t *fmu)
Get the step size for default experiment as specified in the XML file.
- FMILIB_EXPORT fmi2_fmu_kind_enu_t fmi2_import_get_fmu_kind (fmi2_import_t *fmu)
Get the type of the FMU (model exchange or co-simulation)
- FMILIB_EXPORT fmi2_import_type_definitions_t * fmi2_import_get_type_definitions (fmi2_import_t *)
Get the list of all the type definitions in the model.
- FMILIB_EXPORT fmi2_import_unit_definitions_t * fmi2_import_get_unit_definitions (fmi2_import_t *fmu)
Get a list of all the unit definitions in the model.
- FMILIB_EXPORT fmi2_import_variable_t * fmi2_import_get_variable_alias_base (fmi2_import_t *fmu, fmi2_import_variable_t *)
Get the variable with the same value reference that is not an alias.
- FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_variable_aliases (fmi2_import_t *fmu, fmi2_import_variable_t *)
- FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_variable_list (fmi2_import_t *fmu, int sortOrder)
Get the list of all the variables in the model.
- FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_create_var_list (fmi2_import_t *fmu, fmi2_import_variable_t *v)
Create a variable list with a single variable.
- FMILIB_EXPORT size_t fmi2_import_get_vendors_num (fmi2_import_t *fmu)
Get the number of vendors that had annotations in the XML.
- FMILIB_EXPORT const char * fmi2_import_get_vendor_name (fmi2_import_t *fmu, size_t index)

Get the name of the vendor with that had annotations in the XML by index.

- `FMILIB_EXPORT size_t fmi2_import_get_log_categories_num (fmi2_import_t *fmu)`

Get the number of log categories defined in the XML.

- `FMILIB_EXPORT const char * fmi2_import_get_log_category (fmi2_import_t *fmu, size_t index)`

Get the log category by index.

- `FMILIB_EXPORT const char * fmi2_import_get_log_category_description (fmi2_import_t *fmu, size_t index)`

Get the log category description by index.

- `FMILIB_EXPORT size_t fmi2_import_get_source_files_me_num (fmi2_import_t *fmu)`

Get the number of source files for ME defined in the XML.

- `FMILIB_EXPORT const char * fmi2_import_get_source_file_me (fmi2_import_t *fmu, size_t index)`

Get the ME source file by index.

- `FMILIB_EXPORT size_t fmi2_import_get_source_files_cs_num (fmi2_import_t *fmu)`

Get the number of source files for CS defined in the XML.

- `FMILIB_EXPORT const char * fmi2_import_get_source_file_cs (fmi2_import_t *fmu, size_t index)`

Get the CS source file by index.

- `FMILIB_EXPORT fmi2_import_variable_t * fmi2_import_get_variable_by_name (fmi2_import_t *fmu, const char *name)`

Get variable by variable name.

- `FMILIB_EXPORT fmi2_import_variable_t * fmi2_import_get_variable_by_vr (fmi2_import_t *fmu, fmi2_base_type_enumeration_t baseType, fmi2_value_reference_t vr)`

Get variable by value reference.

- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_outputs_list (fmi2_import_t *fmu)`

Get the list of all the output variables in the model.

- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_derivatives_list (fmi2_import_t *fmu)`

Get the list of all the derivative variables in the model.

- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_discrete_states_list (fmi2_import_t *fmu)`

Get the list of all the discrete state variables in the model.

- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_get_initial_unknowns_list (fmi2_import_t *fmu)`

Get the list of all the initial unknown variables in the model.

- `FMILIB_EXPORT void fmi2_import_get_outputs_dependencies (fmi2_import_t *fmu, size_t **startIndex, size_t **dependency, char **factorKind)`

Get dependency information in row-compressed format.

- `FMILIB_EXPORT void fmi2_import_get_derivatives_dependencies (fmi2_import_t *fmu, size_t **startIndex, size_t **dependency, char **factorKind)`
Get dependency information in row-compressed format.
- `FMILIB_EXPORT void fmi2_import_get_discrete_states_dependencies (fmi2_import_t *fmu, size_t **startIndex, size_t **dependency, char **factorKind)`
Get dependency information in row-compressed format.
- `FMILIB_EXPORT void fmi2_import_get_initial_unknowns_dependencies (fmi2_import_t *fmu, size_t **startIndex, size_t **dependency, char **factorKind)`
Get dependency information in row-compressed format.

9.16.1 Detailed Description

Public interface to the FMI import C-library.

Definition in file [fmi2_import.h](#).

9.17 src/Import/include/FMI2/fmi2_import_capi.h File Reference

```
#include <JM/jm_callbacks.h>    #include <FMI/fmi_import-
_util.h>    #include <FMI/fmi_import_context.h>    #include
<FMI2/fmi2_types.h>    #include <FMI2/fmi2_functions.h> ×
#include <FMI2/fmi2_enums.h>
```

Functions

- `FMILIB_EXPORT jm_status_enu_t fmi2_import_create_dllfmu (fmi2_import_t *fmu, fmi2_fmu_kind_enu_t fmuKind, const fmi2_callback_functions_t *callbackFunctions)`
Create a C-API struct. The C-API struct is a placeholder for the FMI DLL functions.
- `FMILIB_EXPORT void fmi2_import_destroy_dllfmu (fmi2_import_t *fmu)`
Free a C-API struct. All memory allocated since the struct was created is freed.
- `FMILIB_EXPORT void fmi2_import_set_debug_mode (fmi2_import_t *fmu, int mode)`
Set CAPI debug mode flag. Setting to non-zero prevents DLL unloading in fmi2_import_destroy_dllfmu while all the memory is deallocated. This is to support valgrind debugging.
- `FMILIB_EXPORT const char * fmi2_import_get_version (fmi2_import_t *fmu)`
Wrapper for the FMI function fmiGetVersion()
- `FMILIB_EXPORT fmi2_status_t fmi2_import_set_debug_logging (fmi2_import_t *fmu, fmi2_boolean_t loggingOn, size_t nCategories, fmi2_string_t categories[])`
Wrapper for the FMI function fmiSetDebugLogging(...)
- `FMILIB_EXPORT jm_status_enu_t fmi2_import_instantiate (fmi2_import_t *fmu, fmi2_string_t instanceName, fmi2_type_t fmuType, fmi2_string_t fmuResourceLocation, fmi2_boolean_t visible)`

Wrapper for the FMI function fmiInstantiate(...)

- **FMILIB_EXPORT** void **fmi2_import_free_instance** (fmi2_import_t *fmu)

Wrapper for the FMI function fmiFreeInstance(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_setup_experiment** (fmi2_import_t *fmu, fmi2_boolean_t toleranceDefined, fmi2_real_t tolerance, fmi2_real_t startTime, fmi2_boolean_t stopTimeDefined, fmi2_real_t stopTime)

Calls the FMI function fmiSetupExperiment(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_enter_initialization_mode** (fmi2_import_t *fmu)

Calls the FMI function fmiEnterInitializationMode(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_exit_initialization_mode** (fmi2_import_t *fmu)

Calls the FMI function fmiExitInitializationMode(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_terminate** (fmi2_import_t *fmu)

Wrapper for the FMI function fmiTerminate(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_reset** (fmi2_import_t *fmu)

Wrapper for the FMI function fmiReset(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_set_real** (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_real_t value[])

Wrapper for the FMI function fmiSetReal(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_set_integer** (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_integer_t value[])

Wrapper for the FMI function fmiSetInteger(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_set_boolean** (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_boolean_t value[])

Wrapper for the FMI function fmiSetBoolean(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_set_string** (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_string_t value[])

Wrapper for the FMI function fmiSetString(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_get_real** (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_real_t value[])

Wrapper for the FMI function fmiGetReal(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_get_integer** (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_integer_t value[])

Wrapper for the FMI function fmiGetInteger(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_get_boolean** (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_boolean_t value[])

Wrapper for the FMI function fmiGetBoolean(...)

- **FMILIB_EXPORT** fmi2_status_t **fmi2_import_get_string** (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, fmi2_string_t value[])

Wrapper for the FMI function fmiGetString(...)

- **FMILIB_EXPORT** const char * **fmi2_import_get_types_platform** (fmi2_import_t *fmu)

Wrapper for the FMI function fmiGetTypesPlatform(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_fmu_state (fmi2_import_t *fmu, fmi2_FMU_state_t *s)`
Wrapper for the FMI function fmiGetFMUstate(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_set_fmu_state (fmi2_import_t *fmu, fmi2_FMU_state_t s)`
Wrapper for the FMI function fmiSetFMUstate(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_free_fmu_state (fmi2_import_t *fmu, fmi2_FMU_state_t *s)`
Wrapper for the FMI function fmiFreeFMUstate(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_serialized_fmu_state_size (fmi2_import_t *fmu, fmi2_FMU_state_t s, size_t *sz)`
Wrapper for the FMI function fmiSerializedFMUstateSize(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_serialize_fmu_state (fmi2_import_t *fmu, fmi2_FMU_state_t s, fmi2_byte_t data[], size_t sz)`
Wrapper for the FMI function fmiSerializeFMUstate(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_de_serialize_fmu_state (fmi2_import_t *fmu, const fmi2_byte_t data[], size_t sz, fmi2_FMU_state_t *s)`
Wrapper for the FMI function fmiSerializeFMUstate(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_directional_derivative (fmi2_import_t *fmu, const fmi2_value_reference_t v_ref[], size_t nv, const fmi2_value_reference_t z_ref[], size_t nz, const fmi2_real_t dv[], fmi2_real_t dz[])`
Wrapper for the FMI function fmiGetDirectionalDerivative(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_enter_event_mode (fmi2_import_t *fmu)`
Calls the FMI function fmiEnterEventMode(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_new_discrete_states (fmi2_import_t *fmu, fmi2_event_info_t *eventInfo)`
Calls the FMI function fmiNewDiscreteStates(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_enter_continuous_time_mode (fmi2_import_t *fmu)`
Calls the FMI function fmiEnterContinuousTimeMode(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_set_time (fmi2_import_t *fmu, fmi2_real_t time)`
Wrapper for the FMI function fmiSetTime(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_set_continuous_states (fmi2_import_t *fmu, const fmi2_real_t x[], size_t nx)`
Wrapper for the FMI function fmiSetContinuousStates(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_completed_integrator_step (fmi2_import_t *fmu, fmi2_boolean_t noSetFMUStatePriorToCurrentPoint, fmi2_boolean_t *enterEventMode, fmi2_boolean_t *terminateSimulation)`
Wrapper for the FMI function fmiCompletedIntegratorStep(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_derivatives (fmi2_import_t *fmu, fmi2_real_t derivatives[], size_t nx)`
Wrapper for the FMI function fmiGetDerivatives(...)
- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_event_indicators (fmi2_import_t *fmu, fmi2_real_t eventIndicators[], size_t ni)`

Wrapper for the FMI function fmiGetEventIndicators(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_continuous_states (fmi2_import_t *fmu, fmi2_real_t states[], size_t nx)`

Wrapper for the FMI function fmiGetContinuousStates(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_nominals_of_continuous_states (fmi2_import_t *fmu, fmi2_real_t x_nominal[], size_t nx)`

Wrapper for the FMI function fmiGetNominalsOfContinuousStates(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_set_real_input_derivatives (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_integer_t order[], const fmi2_real_t value[])`

Wrapper for the FMI function fmiSetRealInputDerivatives(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_real_output_derivatives (fmi2_import_t *fmu, const fmi2_value_reference_t vr[], size_t nvr, const fmi2_integer_t order[], fmi2_real_t value[])`

Wrapper for the FMI function fmiGetOutputDerivatives(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_cancel_step (fmi2_import_t *fmu)`

Wrapper for the FMI function fmiCancelStep(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_do_step (fmi2_import_t *fmu, fmi2_real_t currentCommunicationPoint, fmi2_real_t communicationStepSize, fmi2_boolean_t newStep)`

Wrapper for the FMI function fmiDoStep(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_status_t *value)`

Wrapper for the FMI function fmiGetStatus(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_real_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_real_t *value)`

Wrapper for the FMI function fmiGetRealStatus(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_integer_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_integer_t *value)`

Wrapper for the FMI function fmiGetIntegerStatus(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_boolean_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_boolean_t *value)`

Wrapper for the FMI function fmiGetBooleanStatus(...)

- `FMILIB_EXPORT fmi2_status_t fmi2_import_get_string_status (fmi2_import_t *fmu, const fmi2_status_kind_t s, fmi2_string_t *value)`

Wrapper for the FMI function fmiGetStringStatus(...)

9.17.1 Detailed Description

Wrapper functions for the FMI 2.0 functions

Definition in file `fmi2_import_capi.h`.

9.18 src/Import/include/FMI2/fmi2_import_convenience.h File - Reference

Public interface to the FMI import C-library. Convenience functions.

```
#include <FMI/fmi_import_context.h> #include <FMI2/fmi2_
functions.h>
```

Data Structures

- struct [fmi2_import_model_counts_t](#)
Collection of counters providing model information.

Functions

- **FMILIB_EXPORT** void [fmi2_import_collect_model_counts](#) ([fmi2_import_t](#) *fmu, [fmi2_import_model_counts_t](#) *counts)
Collect model information by counting the number of variables with specific properties and fillin in [fmi2_import_model_counts_t](#) struct.
- **FMILIB_EXPORT** void [fmi2_import_expand_variable_references](#) ([fmi2_import_t](#) *fmu, const char *msgIn, char *msgOut, [size_t](#) maxMsgSize)
Print msgIn into msgOut by expanding variable references of the form #<Type><V-R># into variable names and replacing '##' with a single #.
- **FMILIB_EXPORT** void [fmi2_log_forwarding](#) ([fmi2_component_t](#) c, [fmi2_string_t](#) instanceName, [fmi2_status_t](#) status, [fmi2_string_t](#) category, [fmi2_string_t](#) message,...)
An implementation of FMI 2.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.
- **FMILIB_EXPORT** void [fmi2_log_forwarding_v](#) ([fmi2_component_t](#) c, [fmi2_string_t](#) instanceName, [fmi2_status_t](#) status, [fmi2_string_t](#) category, [fmi2_string_t](#) message, va_list args)
An implementation of FMI 2.0 logger that forwards the messages to logger function inside [jm_callbacks](#) structure.
- **FMILIB_EXPORT** void [fmi2_default_callback_logger](#) ([fmi2_component_t](#) c, [fmi2_string_t](#) instanceName, [fmi2_status_t](#) status, [fmi2_string_t](#) category, [fmi2_string_t](#) message,...)
Default FMI 2.0 logger may be used when instantiating FMUs.
- **FMILIB_EXPORT** void [fmi2_import_init_logger](#) ([jm_callbacks](#) *cb, [fmi2_callback_functions_t](#) *fmiCallbacks)
Given [fmi2_callback_functions_t](#) logger ([fmi2_logger](#)), the [jm_callbacks](#) logger may be setup to redirect the messages to the [fmi2_logger](#).

9.18.1 Detailed Description

Public interface to the FMI import C-library. Convenience functions. The functions in this file are provided for convenience. The functionality is already available via other lower level functions.

Definition in file [fmi2_import_convenience.h](#).

9.19 src/Import/include/FMI2/fmi2_import_type.h File Reference

Public interface to the FMI XML C-library: variable types handling.

```
#include "fmi2_import_unit.h"
```

Typedefs

Type definitions supporting structures

- typedef struct fmi2_xml_real_typedef_t [fmi2_import_real_typedef_t](#)
Opaque type definition object.
- typedef struct fmi2_xml_integer_typedef_t [fmi2_import_integer_typedef_t](#)
Opaque integer type definition object.
- typedef struct fmi2_xml_enumeration_typedef_t [fmi2_import_enumeration_ -
typedef_t](#)
Opaque enumeration type definition object.
- typedef struct fmi2_xml_variable_typedef_t [fmi2_import_variable_typedef_t](#)
Opaque general variable type definition object.
- typedef struct fmi2_xml_type_definitions_t [fmi2_import_type_definitions_t](#)
Opaque list of the type definitions in the model.

Functions

- **FMILIB_EXPORT** unsigned int [fmi2_import_get_type_definition_number](#) (fmi2_ -
import_type_definitions_t *td)
Get the number of available type definitions.
- **FMILIB_EXPORT** [fmi2_import_variable_typedef_t](#) * [fmi2_import_get_typedef](#)
(fmi2_import_type_definitions_t *td, unsigned int index)
Get a type definition specified by the index.
- **FMILIB_EXPORT** [fmi2_import_display_unit_t](#) * [fmi2_import_get_type_display_ -
unit](#) (fmi2_import_real_typedef_t *)
Get associated display unit for a type definition if any.
- **FMILIB_EXPORT** const char * [fmi2_import_get_type_name](#) (fmi2_import_ -
variable_typedef_t *)
Get the type name.
- **FMILIB_EXPORT** const char * [fmi2_import_get_type_description](#) (fmi2_import_ -
variable_typedef_t *)

Get type description.

- `FMILIB_EXPORT fmi2_base_type_enumeration_t fmi2_import_get_base_type (fmi2_import_variable_typedef_t *)`

Get base type used for the type definition.

- `FMILIB_EXPORT fmi2_import_real_typedef_t * fmi2_import_get_type_as_real (fmi2_import_variable_typedef_t *)`

Cast the general type definition object to an object with a specific base type.

- `FMILIB_EXPORT fmi2_import_integer_typedef_t * fmi2_import_get_type_as_int (fmi2_import_variable_typedef_t *)`

Cast the general type definition object to an object with a specific base type.

- `FMILIB_EXPORT fmi2_import_enumeration_typedef_t * fmi2_import_get_type_as_enum (fmi2_import_variable_typedef_t *)`

Cast the general type definition object to an object with a specific base type.

- `FMILIB_EXPORT const char * fmi2_import_get_type_quantity (fmi2_import_variable_typedef_t *)`

Get the quantity associated with the type definition.

- `FMILIB_EXPORT double fmi2_import_get_real_type_min (fmi2_import_real_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT double fmi2_import_get_real_type_max (fmi2_import_real_typedef_t *)`

Get maximum value for the type.

- `FMILIB_EXPORT double fmi2_import_get_real_type_nominal (fmi2_import_real_typedef_t *)`

Get the nominal value associated with the type definition.

- `FMILIB_EXPORT fmi2_import_unit_t * fmi2_import_get_real_type_unit (fmi2_import_real_typedef_t *)`

Get the unit object associated with the type definition if any.

- `FMILIB_EXPORT int fmi2_import_get_real_type_is_relative_quantity (fmi2_import_real_typedef_t *)`

Get the 'relativeQuantity' flag.

- `FMILIB_EXPORT int fmi2_import_get_real_type_is_unbounded (fmi2_import_real_typedef_t *)`

Get the 'unbounded' flag.

- `FMILIB_EXPORT int fmi2_import_get_integer_type_min (fmi2_import_integer_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT int fmi2_import_get_integer_type_max (fmi2_import_integer_typedef_t *)`

Get maximum value for the type.

- `FMILIB_EXPORT unsigned int fmi2_import_get_enum_type_min (fmi2_import_enumeration_typedef_t *)`

Get minimal value for the type.

- `FMILIB_EXPORT unsigned int fmi2_import_get_enum_type_max (fmi2_import_enumeration_typedef_t *)`

Get maximum value for the type.

- **FMILIB_EXPORT** unsigned int [fmi2_import_get_enum_type_size](#) ([fmi2_import_enumeration_typedef_t](#) *)

Get the number of elements in the enum.

- **FMILIB_EXPORT** const char * [fmi2_import_get_enum_type_item_name](#) ([fmi2_import_enumeration_typedef_t](#) *, unsigned int item)

Get an enumeration item name by index.

- **FMILIB_EXPORT** int [fmi2_import_get_enum_type_item_value](#) ([fmi2_import_enumeration_typedef_t](#) *, unsigned int item)

Get an enumeration item value by index.

- **FMILIB_EXPORT** const char * [fmi2_import_get_enum_type_item_description](#) ([fmi2_import_enumeration_typedef_t](#) *, unsigned int item)

Get an enumeration item description by index.

- **FMILIB_EXPORT** const char * [fmi2_import_get_enum_type_value_name](#) ([fmi2_import_enumeration_typedef_t](#) *t, int value)

Get an enumeration item name for the given value.

9.19.1 Detailed Description

Public interface to the FMI XML C-library: variable types handling.

Definition in file [fmi2_import_type.h](#).

9.20 src/Import/include/FMI2/fmi2_import_unit.h File Reference

Public interface to the FMI import C-library. Handling of variable units.

```
#include <fmilib_config.h>
```

Typedefs

Structures encapsulating unit information

- typedef struct [fmi2_xml_unit_t](#) [fmi2_import_unit_t](#)
A variable unit defined with a unit definition.
- typedef struct [fmi2_xml_display_unit_t](#) [fmi2_import_display_unit_t](#)
A display unit.
- typedef struct [fmi2_xml_unit_definitions_t](#) [fmi2_import_unit_definitions_t](#)
The list of all the unit definitions in the model.

Functions

- **FMILIB_EXPORT** unsigned int [fmi2_import_get_unit_definitions_number](#) ([fmi2_import_unit_definitions_t](#) *)
Get the number of unit definitions.

- `FMILIB_EXPORT fmi2_import_unit_t * fmi2_import_get_unit (fmi2_import_unit_definitions_t *, unsigned int index)`
Get a unit definition.
- `FMILIB_EXPORT const char * fmi2_import_get_unit_name (fmi2_import_unit_t *)`
Get a unit name.
- `FMILIB_EXPORT unsigned int fmi2_import_get_unit_display_unit_number (fmi2_import_unit_t *)`
Get the number of display units associated with this unit.
- `FMILIB_EXPORT const int * fmi2_import_get_SI_unit_exponents (fmi2_import_unit_t *)`
Get fmi2_SI_base_units_Num SI base units exponents associated with the unit.
- `FMILIB_EXPORT double fmi2_import_get_SI_unit_factor (fmi2_import_unit_t *)`
Get factor to the corresponding SI base units.
- `FMILIB_EXPORT double fmi2_import_get_SI_unit_offset (fmi2_import_unit_t *)`
Get offset to the corresponding SI base units.
- `FMILIB_EXPORT double fmi2_import_convert_to_SI_base_unit (double, fmi2_import_unit_t *)`
Convert a value with respect to the unit to the value with respect to the SI base unit.
- `FMILIB_EXPORT double fmi2_import_convert_from_SI_base_unit (double, fmi2_import_unit_t *)`
Convert a value with respect to the SI base unit to the value with respect to the unit.
- `FMILIB_EXPORT fmi2_import_display_unit_t * fmi2_import_get_unit_display_unit (fmi2_import_unit_t *, size_t index)`
Get a display unit object by index.
- `FMILIB_EXPORT fmi2_import_unit_t * fmi2_import_get_base_unit (fmi2_import_display_unit_t *)`
Get unit definition for a display unit.
- `FMILIB_EXPORT const char * fmi2_import_get_display_unit_name (fmi2_import_display_unit_t *)`
Get display unit name.
- `FMILIB_EXPORT fmi2_real_t fmi2_import_get_display_unit_factor (fmi2_import_display_unit_t *)`
Get the "factor" associated with the display unit.
- `FMILIB_EXPORT fmi2_real_t fmi2_import_get_display_unit_offset (fmi2_import_display_unit_t *)`
Get the "offset" associated with the display unit.
- `FMILIB_EXPORT fmi2_real_t fmi2_import_convert_to_display_unit (fmi2_real_t value, fmi2_import_display_unit_t *du, int isRelativeQuantity)`
Convert a value measured in "units" to a value measured with "display units".
- `FMILIB_EXPORT fmi2_real_t fmi2_import_convert_from_display_unit (fmi2_real_t value, fmi2_import_display_unit_t *du, int isRelativeQuantity)`
Convert a value measured in "display units" to a value measured with "units".

9.20.1 Detailed Description

Public interface to the FMI import C-library. Handling of variable units.

Definition in file [fmi2_import_unit.h](#).

9.21 src/Import/include/FMI2/fmi2_import_variable.h File Reference

Public interface to the FMI import C-library. Handling of model variables.

```
#include <FMI/fmi_import_context.h> #include "fmi2_import-
_type.h" #include "fmi2_import_unit.h"
```

Typedefs

Scalar variable types

- typedef struct fmi2_xml_variable_t [fmi2_import_variable_t](#)
General variable type.
- typedef struct fmi2_xml_real_variable_t [fmi2_import_real_variable_t](#)
Opaque real variable.
- typedef struct fmi2_xml_integer_variable_t [fmi2_import_integer_variable_t](#)
Opaque integer variable.
- typedef struct fmi2_xml_string_variable_t [fmi2_import_string_variable_t](#)
Opaque string variable.
- typedef struct fmi2_xml_enum_variable_t [fmi2_import_enum_variable_t](#)
Opaque enumeration variable.
- typedef struct fmi2_xml_bool_variable_t [fmi2_import_bool_variable_t](#)
Opaque boolean variable.
- typedef struct [fmi2_import_variable_list_t](#) [fmi2_import_variable_list_t](#)
List of variables.

Functions

- **FMLIB_EXPORT** const char * [fmi2_import_get_variable_name](#) ([fmi2_import_variable_t](#) *)
Get the variable name.
- **FMLIB_EXPORT** const char * [fmi2_import_get_variable_description](#) ([fmi2_import_variable_t](#) *)
Get variable description.
- **FMLIB_EXPORT** [fmi2_value_reference_t](#) [fmi2_import_get_variable_vr](#) ([fmi2_import_variable_t](#) *)
Get variable value reference.
- **FMLIB_EXPORT** [fmi2_import_variable_typedef_t](#) * [fmi2_import_get_variable_declared_type](#) ([fmi2_import_variable_t](#) *)
For scalar variable gives the type definition is present.

- FMILIB_EXPORT fmi2_base_type_enu_t fmi2_import_get_variable_base_type (fmi2_import_variable_t *)
Get variable base type.
- FMILIB_EXPORT int fmi2_import_get_variable_has_start (fmi2_import_variable_t *)
Check if the variable has "start" attribute.
- FMILIB_EXPORT fmi2_variability_enu_t fmi2_import_get_variability (fmi2_import_variable_t *)
Get variability attribute.
- FMILIB_EXPORT fmi2_causality_enu_t fmi2_import_get_causality (fmi2_import_variable_t *)
Get causality attribute.
- FMILIB_EXPORT fmi2_initial_enu_t fmi2_import_get_initial (fmi2_import_variable_t *)
Get initial attribute.
- FMILIB_EXPORT fmi2_import_variable_t * fmi2_import_get_previous (fmi2_import_variable_t *v)
Get the variable that holds the previous value of this variable, if defined.
- FMILIB_EXPORT fmi2_boolean_t fmi2_import_get_canHandleMultipleSetPerTimeInstant (fmi2_import_variable_t *v)
Get the canHandleMultipleSetPerTimeInstant flag for a variable.
- FMILIB_EXPORT fmi2_import_real_variable_t * fmi2_import_get_variable_as_real (fmi2_import_variable_t *)
Cast general variable to a one with the specific type.
- FMILIB_EXPORT fmi2_import_integer_variable_t * fmi2_import_get_variable_as_integer (fmi2_import_variable_t *)
Cast general variable to a one with the specific type.
- FMILIB_EXPORT fmi2_import_enum_variable_t * fmi2_import_get_variable_as_enum (fmi2_import_variable_t *)
Cast general variable to a one with the specific type.
- FMILIB_EXPORT fmi2_import_string_variable_t * fmi2_import_get_variable_as_string (fmi2_import_variable_t *)
Cast general variable to a one with the specific type.
- FMILIB_EXPORT fmi2_import_bool_variable_t * fmi2_import_get_variable_as_boolean (fmi2_import_variable_t *)
Cast general variable to a one with the specific type.
- FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_start (fmi2_import_real_variable_t *v)
Get the variable start attribute.
- FMILIB_EXPORT fmi2_import_real_variable_t * fmi2_import_get_real_variable_derivative_of (fmi2_import_real_variable_t *v)
Get the variable that this is a derivative of, if defined.
- FMILIB_EXPORT fmi2_boolean_t fmi2_import_get_real_variable_reinit (fmi2_import_real_variable_t *v)
Get the reinit flag for a real variable.

- `FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_max (fmi2_import_real_variable_t *v)`
Get maximum value for the variable.
- `FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_min (fmi2_import_real_variable_t *v)`
Get minimal value for the variable.
- `FMILIB_EXPORT fmi2_real_t fmi2_import_get_real_variable_nominal (fmi2_import_real_variable_t *v)`
Get nominal value for the variable.
- `FMILIB_EXPORT fmi2_import_unit_t * fmi2_import_get_real_variable_unit (fmi2_import_real_variable_t *v)`
Get associated "unit" object if any.
- `FMILIB_EXPORT fmi2_import_display_unit_t * fmi2_import_get_real_variable_display_unit (fmi2_import_real_variable_t *v)`
Get associated "display unit" object if any.
- `FMILIB_EXPORT const char * fmi2_import_get_string_variable_start (fmi2_import_string_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT fmi2_boolean_t fmi2_import_get_boolean_variable_start (fmi2_import_bool_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_integer_variable_start (fmi2_import_integer_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_integer_variable_min (fmi2_import_integer_variable_t *v)`
Get minimal value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_integer_variable_max (fmi2_import_integer_variable_t *v)`
Get max value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_enum_variable_start (fmi2_import_enum_variable_t *v)`
Get start value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_enum_variable_min (fmi2_import_enum_variable_t *v)`
Get minimal value for the variable.
- `FMILIB_EXPORT int fmi2_import_get_enum_variable_max (fmi2_import_enum_variable_t *v)`
Get max value for the variable.
- `FMILIB_EXPORT fmi2_variable_alias_kind_enumeration_t fmi2_import_get_variable_alias_kind (fmi2_import_variable_t *v)`
Get the variable alias kind.
- `FMILIB_EXPORT size_t fmi2_import_get_variable_original_order (fmi2_import_variable_t *v)`
Get the original index in xml of the variable.

9.21.1 Detailed Description

Public interface to the FMI import C-library. Handling of model variables.

Definition in file [fmi2_import_variable.h](#).

9.22 [src/Import/include/FMI2/fmi2_import_variable_list.h](#) File Reference

Public interface to the FMI XML C-library. Handling of variable lists.

```
#include "fmi2_import_variable.h"
```

Functions

- [fmi2_import_variable_list_t](#) * [fmi2_import_alloc_variable_list](#) ([fmi2_import_t](#) *fmu, [size_t](#) size)
Allocate an empty list.
- [FMILIB_EXPORT](#) void [fmi2_import_free_variable_list](#) ([fmi2_import_variable_list_t](#) *vl)
Free a variable list. Note that variable lists are allocated dynamically and must be freed when not needed any longer.
- [FMILIB_EXPORT](#) [fmi2_import_variable_list_t](#) * [fmi2_import_clone_variable_list](#) ([fmi2_import_variable_list_t](#) *vl)
Make a copy of the list.
- [FMILIB_EXPORT](#) [size_t](#) [fmi2_import_get_variable_list_size](#) ([fmi2_import_variable_list_t](#) *vl)
Get number of variables in a list.
- [FMILIB_EXPORT](#) const [fmi2_value_reference_t](#) * [fmi2_import_get_value_references_list](#) ([fmi2_import_variable_list_t](#) *vl)
Get a pointer to the list of the value references for all the variables.
- [FMILIB_EXPORT](#) [fmi2_import_variable_t](#) * [fmi2_import_get_variable](#) ([fmi2_import_variable_list_t](#) *vl, [size_t](#) index)
Get a single variable from the list.

Operations on variable lists. Every operation creates a new list.

- `typedef int(* fmi2\_import\_variable\_filter\_function\_ft )(fmi2\_import\_variable\_t *vl, void *data)`
Callback function typedef for the [fmiFilterVariables](#).
- [FMILIB_EXPORT](#) [fmi2_import_variable_list_t](#) * [fmi2_import_get_sublist](#) ([fmi2_import_variable_list_t](#) *vl, [size_t](#) fromIndex, [size_t](#) toIndex)
Select sub-lists.

- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_filter_variables (fmi2_import_variable_list_t *vl, fmi2_import_variable_filter_function_ft filter, void *context)`
Call the provided 'filter' function on every variable in the list and create a new list.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_join_var_list (fmi2_import_variable_list_t *a, fmi2_import_variable_list_t *b)`
Create a new variable list by concatenating two lists.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_append_to_var_list (fmi2_import_variable_list_t *vl, fmi2_import_variable_t *v)`
Append a variable to the variable list.
- `FMILIB_EXPORT fmi2_import_variable_list_t * fmi2_import_prepend_to_var_list (fmi2_import_variable_list_t *vl, fmi2_import_variable_t *v)`
Prepend a variable to the variable list.
- `FMILIB_EXPORT jm_status_enu_t fmi2_import_var_list_push_back (fmi2_import_variable_list_t *vl, fmi2_import_variable_t *v)`
Add a variable to a variable list.

9.22.1 Detailed Description

Public interface to the FMI XML C-library. Handling of variable lists.

Definition in file [fmi2_import_variable_list.h](#).

9.23 src/Util/include/FMI/fmi_util.h File Reference

Some low-level utility functions suitable for all standards.

```
#include <fmilib_config.h> #include <JM/jm_callbacks.h>
```

Functions

- `FMILIB_EXPORT char * fmi_construct_dll_dir_name (jm_callbacks *callbacks, const char *fmu_unzipped_path)`
Given directory name fmu_unzipped_path and construct the directory path for Dll/so.
- `FMILIB_EXPORT char * fmi_construct_dll_file_name (jm_callbacks *callbacks, const char *dll_dir_name, const char *model_identifier)`
Given model_identifier construct the dll/so name by adding platform suffix.

9.23.1 Detailed Description

Some low-level utility functions suitable for all standards.

Definition in file [fmi_util.h](#).

9.24 src/Util/include/FMI/fmi_version.h File Reference

Enum defining supported FMI versions.

```
#include <fmilib_config.h>
```

Enumerations

- enum [fmi_version_enumeration_t](#) { [fmi_version_unknown_enumeration_t](#) = 0, [fmi_version_1_enumeration_t](#), [fmi_version_2_0_enumeration_t](#), [fmi_version_unsupported_enumeration_t](#) }

Supported versions of FMI standard.

Functions

- [FMILIB_EXPORT](#) const char * [fmi_version_to_string](#) ([fmi_version_enumeration_t](#) v)

9.24.1 Detailed Description

Enum defining supported FMI versions.

Definition in file [fmi_version.h](#).

9.25 src/Util/include/FMI1/fmi1_enums.h File Reference

Definitions the enum types used with FMI 1.0 libs.

```
#include <fmilib_config.h>
```

Typedefs

- typedef enum [fmi1_variable_naming_convention_enumeration_t](#) [fmi1_variable_naming_convention_enumeration_t](#)
Naming convention for the variables in XML file.
- typedef enum [fmi1_fmu_kind_enumeration_t](#) [fmi1_fmu_kind_enumeration_t](#)
FMU 1.0 kinds.
- typedef enum [fmi1_variability_enumeration_t](#) [fmi1_variability_enumeration_t](#)
Variability property for variables.
- typedef enum [fmi1_causality_enumeration_t](#) [fmi1_causality_enumeration_t](#)
Causality property for variables.
- typedef enum [fmi1_variable_alias_kind_enumeration_t](#) [fmi1_variable_alias_kind_enumeration_t](#)
Alias property for variables.
- typedef enum [fmi1_base_type_enumeration_t](#) [fmi1_base_type_enumeration_t](#)
Base types used in type definitions.

Enumerations

- enum [fmi1_variable_naming_convention_enumeration_t](#) { [fmi1_naming_enumeration_flat](#), [fmi1_naming_enumeration_structured](#), [fmi1_naming_enumeration_unknown](#) }
Naming convention for the variables in XML file.
- enum [fmi1_fmu_kind_enumeration_t](#) { [fmi1_fmu_kind_enumeration_me](#) = 0, [fmi1_fmu_kind_enumeration_cs_standalone](#), [fmi1_fmu_kind_enumeration_cs_tool](#), [fmi1_fmu_kind_enumeration_unknown](#) }
FMU 1.0 kinds.
- enum [fmi1_variability_enumeration_t](#) { [fmi1_variability_enumeration_constant](#), [fmi1_variability_enumeration_parameter](#), [fmi1_variability_enumeration_discrete](#), [fmi1_variability_enumeration_continuous](#), [fmi1_variability_enumeration_unknown](#) }
Variability property for variables.
- enum [fmi1_causality_enumeration_t](#) { [fmi1_causality_enumeration_input](#), [fmi1_causality_enumeration_output](#), [fmi1_causality_enumeration_internal](#), [fmi1_causality_enumeration_none](#), [fmi1_causality_enumeration_unknown](#) }
Causality property for variables.
- enum [fmi1_variable_alias_kind_enumeration_t](#) { [fmi1_variable_is_negated_alias](#) = -1, [fmi1_variable_is_not_alias](#) = 0, [fmi1_variable_is_alias](#) = 1 }
Alias property for variables.
- enum [fmi1_base_type_enumeration_t](#) { [fmi1_base_type_real](#), [fmi1_base_type_int](#), [fmi1_base_type_bool](#), [fmi1_base_type_str](#), [fmi1_base_type_enum](#) }
Base types used in type definitions.

Functions

- [FMILIB_EXPORT](#) const char * [fmi1_naming_convention_to_string](#) ([fmi1_variable_naming_convention_enumeration_t](#) convention)
Convert a [fmi1_variable_naming_convention_enumeration_t](#) constant into string.
- [FMILIB_EXPORT](#) const char * [fmi1_fmu_kind_to_string](#) ([fmi1_fmu_kind_enumeration_t](#) kind)
Convert a [fmi1_fmu_kind_enumeration_t](#) constant into string.
- [FMILIB_EXPORT](#) const char * [fmi1_variability_to_string](#) ([fmi1_variability_enumeration_t](#) v)
Convert a [fmi1_variability_enumeration_t](#) constant into string.
- [FMILIB_EXPORT](#) const char * [fmi1_causality_to_string](#) ([fmi1_causality_enumeration_t](#) c)
Convert a [fmi1_causality_enumeration_t](#) constant into string.
- [FMILIB_EXPORT](#) const char * [fmi1_base_type_to_string](#) ([fmi1_base_type_enumeration_t](#) bt)
Convert base type constant to string.

9.25.1 Detailed Description

Definitions the enum types used with FMI 1.0 libs.

Definition in file [fmi1_enums.h](#).

9.26 src/Util/include/FMI1/fmi1_functions.h File Reference

```
#include <fmilib_config.h> #include "fmi1_types.h" #include
<string.h>
```

Data Structures

- struct [fmi1_me_callback_functions_t](#)
- struct [fmi1_callback_functions_t](#)
- struct [fmi1_event_info_t](#)

Typedefs

- typedef void(* [fmi1_callback_logger_ft](#))(fmi1_component_t c, fmi1_string_t instanceName, [fmi1_status_t](#) status, fmi1_string_t category, fmi1_string_t message,...)
- typedef void(* [fmi1_callback_allocate_memory_ft](#))(size_t nobj, [size_t](#) size)
- typedef void(* [fmi1_callback_free_memory_ft](#))(void *obj)
- typedef void(* [fmi1_step_finished_ft](#))(fmi1_component_t c, [fmi1_status_t](#) status)
- typedef const char *(* [fmi1_get_version_ft](#))(void)
- typedef [fmi1_status_t](#)(* [fmi1_set_debug_logging_ft](#))(fmi1_component_t c, [fmi1_boolean_t](#) loggingOn)
- typedef [fmi1_status_t](#)(* [fmi1_set_real_ft](#))(fmi1_component_t c, const [fmi1_value_reference_t](#) vr[], [size_t](#) nvr, const [fmi1_real_t](#) value[])
- typedef [fmi1_status_t](#)(* [fmi1_set_integer_ft](#))(fmi1_component_t c, const [fmi1_value_reference_t](#) vr[], [size_t](#) nvr, const [fmi1_integer_t](#) value[])
- typedef [fmi1_status_t](#)(* [fmi1_set_boolean_ft](#))(fmi1_component_t c, const [fmi1_value_reference_t](#) vr[], [size_t](#) nvr, const [fmi1_boolean_t](#) value[])
- typedef [fmi1_status_t](#)(* [fmi1_set_string_ft](#))(fmi1_component_t c, const [fmi1_value_reference_t](#) vr[], [size_t](#) nvr, const [fmi1_string_t](#) value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_real_ft](#))(fmi1_component_t c, const [fmi1_value_reference_t](#) vr[], [size_t](#) nvr, [fmi1_real_t](#) value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_integer_ft](#))(fmi1_component_t c, const [fmi1_value_reference_t](#) vr[], [size_t](#) nvr, [fmi1_integer_t](#) value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_boolean_ft](#))(fmi1_component_t c, const [fmi1_value_reference_t](#) vr[], [size_t](#) nvr, [fmi1_boolean_t](#) value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_string_ft](#))(fmi1_component_t c, const [fmi1_value_reference_t](#) vr[], [size_t](#) nvr, [fmi1_string_t](#) value[])
- typedef const char *(* [fmi1_get_model_typesPlatform_ft](#))(void)
- typedef [fmi1_component_t](#)(* [fmi1_instantiate_model_ft](#))(fmi1_string_t instanceName, [fmi1_string_t](#) GUID, [fmi1_me_callback_functions_t](#) functions, [fmi1_boolean_t](#) loggingOn)
- typedef void(* [fmi1_free_model_instance_ft](#))(fmi1_component_t c)
- typedef [fmi1_status_t](#)(* [fmi1_set_time_ft](#))(fmi1_component_t c, [fmi1_real_t](#) time)

- typedef [fmi1_status_t](#)(* [fmi1_set_continuous_states_ft](#))(fmi1_component_t c, const fmi1_real_t x[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_completed_integrator_step_ft](#))(fmi1_component_t c, fmi1_boolean_t *callEventUpdate)
- typedef [fmi1_status_t](#)(* [fmi1_initialize_ft](#))(fmi1_component_t c, fmi1_boolean_t toleranceControlled, fmi1_real_t relativeTolerance, [fmi1_event_info_t](#) *eventInfo)
- typedef [fmi1_status_t](#)(* [fmi1_get_derivatives_ft](#))(fmi1_component_t c, fmi1_real_t derivatives[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_get_event_indicators_ft](#))(fmi1_component_t c, fmi1_real_t eventIndicators[], [size_t](#) ni)
- typedef [fmi1_status_t](#)(* [fmi1_event_update_ft](#))(fmi1_component_t c, fmi1_boolean_t intermediateResults, [fmi1_event_info_t](#) *eventInfo)
- typedef [fmi1_status_t](#)(* [fmi1_get_continuous_states_ft](#))(fmi1_component_t c, fmi1_real_t states[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_get_nominal_continuousStates_ft](#))(fmi1_component_t c, fmi1_real_t x_nominal[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_get_state_valueReferences_ft](#))(fmi1_component_t c, fmi1_value_reference_t vrx[], [size_t](#) nx)
- typedef [fmi1_status_t](#)(* [fmi1_terminate_ft](#))(fmi1_component_t c)
- typedef const [char](#) *(* [fmi1_get_types_platform_ft](#))(void)
- typedef fmi1_component_t(* [fmi1_instantiate_slave_ft](#))(fmi1_string_t instanceName, fmi1_string_t fmuGUID, fmi1_string_t fmuLocation, fmi1_string_t mimeType, fmi1_real_t timeout, fmi1_boolean_t visible, fmi1_boolean_t interactive, [fmi1_callback_functions_t](#) functions, fmi1_boolean_t loggingOn)
- typedef [fmi1_status_t](#)(* [fmi1_initialize_slave_ft](#))(fmi1_component_t c, fmi1_real_t tStart, fmi1_boolean_t StopTimeDefined, fmi1_real_t tStop)
- typedef [fmi1_status_t](#)(* [fmi1_terminate_slave_ft](#))(fmi1_component_t c)
- typedef [fmi1_status_t](#)(* [fmi1_reset_slave_ft](#))(fmi1_component_t c)
- typedef void(* [fmi1_free_slave_instance_ft](#))(fmi1_component_t c)
- typedef [fmi1_status_t](#)(* [fmi1_set_real_inputDerivatives_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], [size_t](#) nvr, const fmi1_integer_t order[], const fmi1_real_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_get_real_outputDerivatives_ft](#))(fmi1_component_t c, const fmi1_value_reference_t vr[], [size_t](#) nvr, const fmi1_integer_t order[], fmi1_real_t value[])
- typedef [fmi1_status_t](#)(* [fmi1_cancel_step_ft](#))(fmi1_component_t c)
- typedef [fmi1_status_t](#)(* [fmi1_do_step_ft](#))(fmi1_component_t c, fmi1_real_t currentCommunicationPoint, fmi1_real_t communicationStepSize, fmi1_boolean_t newStep)
- typedef [fmi1_status_t](#)(* [fmi1_get_status_ft](#))(fmi1_component_t c, const [fmi1_status_kind_t](#) s, [fmi1_status_t](#) *value)
- typedef [fmi1_status_t](#)(* [fmi1_get_real_status_ft](#))(fmi1_component_t c, const [fmi1_status_kind_t](#) s, fmi1_real_t *value)
- typedef [fmi1_status_t](#)(* [fmi1_get_integer_status_ft](#))(fmi1_component_t c, const [fmi1_status_kind_t](#) s, fmi1_integer_t *value)
- typedef [fmi1_status_t](#)(* [fmi1_get_boolean_status_ft](#))(fmi1_component_t c, const [fmi1_status_kind_t](#) s, fmi1_boolean_t *value)
- typedef [fmi1_status_t](#)(* [fmi1_get_string_status_ft](#))(fmi1_component_t c, const [fmi1_status_kind_t](#) s, fmi1_string_t *value)

Enumerations

- enum `fmi1_status_t` { `fmi1_status_ok`, `fmi1_status_warning`, `fmi1_status_discard`, `fmi1_status_error`, `fmi1_status_fatal`, `fmi1_status_pending` }
- enum `fmi1_status_kind_t` { `fmi1_do_step_status`, `fmi1_pending_status`, `fmi1_last_successful_time` }

Functions

- `FMILIB_EXPORT` const char * `fmi1_status_to_string` (`fmi1_status_t` status)

9.26.1 Detailed Description

Mapping for the standard FMI 1.0 functions into `fmi1_` namespace.

Definition in file [fmi1_functions.h](#).

9.27 src/Util/include/FMI1/fmi1_types.h File Reference

```
#include <FMI1/fmiPlatformTypes.h>
```

Defines

Renaming of typedefs

- #define `fmiComponent` `fmi1_component_t`
- #define `fmiValueReference` `fmi1_value_reference_t`
- #define `fmiReal` `fmi1_real_t`
- #define `fmiInteger` `fmi1_integer_t`
- #define `fmiBoolean` `fmi1_boolean_t`
- #define `fmiString` `fmi1_string_t`

Typedefs

- typedef enum `fmi1_value_reference_enumeration_t` `fmi1_value_reference_enumeration_t`

Enumerations

- enum `fmi1_boolean_enumeration_t` { `fmi1_true` = `fmiTrue`, `fmi1_false` = `fmiFalse` }
- enum `fmi1_value_reference_enumeration_t` { `fmi1_undefined_value_reference` = (int)`fmiUndefinedValueReference` }

Functions

- static const char * `fmi1_get_platform` (void)

9.27.1 Detailed Description

Transformation of the standard FMI type names into fmi1_ prefixed.

Definition in file [fmi1_types.h](#).

9.28 src/Util/include/FMI2/fmi2_enums.h File Reference

Definitions the enum types used with FMI 2.0 libs.

```
#include <fmilib_config.h>
```

Defines

- #define [FMI2_ME_CAPABILITIES](#)(H)
List of capability flags for ModelExchange.
- #define [FMI2_CS_CAPABILITIES](#)(H)
List of capability flags for CoSimulation.
- #define [FMI2_EXPAND_ME_CAPABILITIES_ENU](#)(c) fmi2_me_ ## c,
- #define [FMI2_EXPAND_CS_CAPABILITIES_ENU](#)(c) fmi2_cs_ ## c,
- #define [FMI2_SI_BASE_UNITS](#)(H) H(kg) H(m) H(s) H(A) H(K) H(mol) H(cd) -
H(rad)
List of SI base units used in Unit definitions.
- #define [FMI2_EXPAND_SI_BASE_UNIT_ENU](#)(c) fmi2_SI_base_unit_ ## c,

Typedefs

- typedef enum [fmi2_variable_naming_convension_enu_t](#) fmi2_variable_naming-
[_convension_enu_t](#)
Naming convention for the variables in XML file.
- typedef enum [fmi2_fmu_kind_enu_t](#) fmi2_fmu_kind_enu_t
FMU 2.0 kinds.
- typedef enum [fmi2_variability_enu_t](#) fmi2_variability_enu_t
Variability property for variables.
- typedef enum [fmi2_causality_enu_t](#) fmi2_causality_enu_t
Causality property for variables.
- typedef enum [fmi2_initial_enu_t](#) fmi2_initial_enu_t
Initial property for variables.
- typedef enum [fmi2_variable_alias_kind_enu_t](#) fmi2_variable_alias_kind_enu_t
Alias property for variables.
- typedef enum [fmi2_base_type_enu_t](#) fmi2_base_type_enu_t
Base types used in type definitions.
- typedef enum [fmi2_capabilities_enu_t](#) fmi2_capabilities_enu_t
Capability flags for ModelExchange and CoSimulation.

- typedef enum [fmi2_SI_base_units_enu_t](#) [fmi2_SI_base_units_enu_t](#)
SI base units used in Unit definitions.
- typedef enum [fmi2_dependency_factor_kind_enu_t](#) [fmi2_dependency_factor_kind_enu_t](#)
Dependency factor kinds are used as part of ModelStructure definition.

Enumerations

- enum [fmi2_variable_naming_convension_enu_t](#) { [fmi2_naming_enu_flat](#), [fmi2_naming_enu_structured](#), [fmi2_naming_enu_unknown](#) }
Naming convention for the variables in XML file.
- enum [fmi2_fmu_kind_enu_t](#) { [fmi2_fmu_kind_unknown](#) = 0, [fmi2_fmu_kind_me](#) = 1, [fmi2_fmu_kind_cs](#) = 2, [fmi2_fmu_kind_me_and_cs](#) = 3 }
FMU 2.0 kinds.
- enum [fmi2_variability_enu_t](#) { [fmi2_variability_enu_constant](#) = 0, [fmi2_variability_enu_fixed](#) = 1, [fmi2_variability_enu_tunable](#) = 2, [fmi2_variability_enu_discrete](#) = 3, [fmi2_variability_enu_continuous](#) = 4, [fmi2_variability_enu_unknown](#) = 5 }
Variability property for variables.
- enum [fmi2_causality_enu_t](#) { [fmi2_causality_enu_parameter](#) = 0, [fmi2_causality_enu_calculated_parameter](#) = 1, [fmi2_causality_enu_input](#) = 2, [fmi2_causality_enu_output](#) = 3, [fmi2_causality_enu_local](#) = 4, [fmi2_causality_enu_independent](#) = 5, [fmi2_causality_enu_unknown](#) = 6 }
Causality property for variables.
- enum [fmi2_initial_enu_t](#) { [fmi2_initial_enu_exact](#), [fmi2_initial_enu_approx](#), [fmi2_initial_enu_calculated](#), [fmi2_initial_enu_unknown](#) }
Initial property for variables.
- enum [fmi2_variable_alias_kind_enu_t](#) { [fmi2_variable_is_not_alias](#) = 0, [fmi2_variable_is_alias](#) = 1 }
Alias property for variables.
- enum [fmi2_base_type_enu_t](#) { [fmi2_base_type_real](#), [fmi2_base_type_int](#), [fmi2_base_type_bool](#), [fmi2_base_type_str](#), [fmi2_base_type_enum](#) }
Base types used in type definitions.
- enum [fmi2_capabilities_enu_t](#)
Capability flags for ModelExchange and CoSimulation.
- enum [fmi2_SI_base_units_enu_t](#)
SI base units used in Unit definitions.
- enum [fmi2_dependency_factor_kind_enu_t](#) { [fmi2_dependency_factor_kind_dependent](#) = 0, [fmi2_dependency_factor_kind_constant](#), [fmi2_dependency_factor_kind_fixed](#), [fmi2_dependency_factor_kind_tunable](#), [fmi2_dependency_factor_kind_discrete](#), [fmi2_dependency_factor_kind_num](#) }
Dependency factor kinds are used as part of ModelStructure definition.

Functions

- `FMILIB_EXPORT const char * fmi2_naming_convention_to_string (fmi2_variable_naming_convention_enumeration_t convention)`
Convert a `fmi2_variable_naming_convention_enumeration_t` constant into string.
- `FMILIB_EXPORT const char * fmi2_fmu_kind_to_string (fmi2_fmu_kind_enumeration_t kind)`
Convert a `fmi2_fmu_kind_enumeration_t` constant into string.
- `FMILIB_EXPORT const char * fmi2_variability_to_string (fmi2_variability_enumeration_t v)`
Convert a `fmi2_variability_enumeration_t` constant into string.
- `FMILIB_EXPORT const char * fmi2_causality_to_string (fmi2_causality_enumeration_t c)`
Convert a `fmi2_causality_enumeration_t` constant into string.
- `FMILIB_EXPORT const char * fmi2_initial_to_string (fmi2_initial_enumeration_t c)`
Convert a `fmi2_initial_enumeration_t` constant into string.
- `FMILIB_EXPORT fmi2_initial_enumeration_t fmi2_get_default_initial (fmi2_variability_enumeration_t v, fmi2_causality_enumeration_t c)`
Get default initial attribute value for the given variability and causality combination.
- `FMILIB_EXPORT fmi2_initial_enumeration_t fmi2_get_valid_initial (fmi2_variability_enumeration_t v, fmi2_causality_enumeration_t c, fmi2_initial_enumeration_t i)`
Check if the combination of variability, causality and initial is valid.
- `FMILIB_EXPORT const char * fmi2_base_type_to_string (fmi2_base_type_enumeration_t bt)`
Convert base type constant to string.
- `FMILIB_EXPORT const char * fmi2_capability_to_string (fmi2_capabilities_enumeration_t id)`
Convert capability flag to a string.
- `FMILIB_EXPORT const char * fmi2_SI_base_unit_to_string (fmi2_SI_base_units_enumeration_t id)`
Convert SI base unit ID a string.
- `FMILIB_EXPORT size_t fmi2_SI_base_unit_exp_to_string (const int exp[fmi2_SI_base_units_enumeration_t], size_t bufSize, char buf[])`
Convert a list of SI base unit exponents (corresponding to the IDs from `fmi2_SI_base_units_enumeration_t`) to a string of the form $kg \cdot m^2 / s^2$. Prints '-' if all the exponents are zero.
- `FMILIB_EXPORT const char * fmi2_dependency_factor_kind_to_string (fmi2_dependency_factor_kind_enumeration_t fc)`
Convert dependency factor kind constant to string.

9.28.1 Detailed Description

Definitions the enum types used with FMI 2.0 libs.

Definition in file `fmi2_enums.h`.

9.28.2 Define Documentation

9.28.2.1 `#define FMI2_EXPAND_ME_CAPABILITIES_ENU( c ) fmi2_me_ ## c,`

Definition at line 159 of file `fmi2_enums.h`.

9.28.2.2 `#define FMI2_EXPAND_CS_CAPABILITIES_ENU( c ) fmi2_cs_ ## c,`

Definition at line 160 of file `fmi2_enums.h`.

9.28.2.3 `#define FMI2_EXPAND_SI_BASE_UNIT_ENU( c ) fmi2_SI_base_unit_ ## c,`

Definition at line 178 of file `fmi2_enums.h`.

9.29 src/Util/include/FMI2/fmi2_functions.h File Reference

```
#include <string.h> #include <fmilib_config.h> #include
"fmi2_types.h"
```

Data Structures

- struct [fmi2_callback_functions_t](#)
- struct [fmi2_event_info_t](#)

Typedefs

- typedef void(* [fmi2_callback_logger_ft](#))(fmi2_component_environment_t env, fmi2_string_t instanceName, [fmi2_status_t](#) status, fmi2_string_t category, fmi2_string_t message,...)
- typedef void(* [fmi2_callback_allocate_memory_ft](#))(size_t nobj, size_t size)
- typedef void(* [fmi2_callback_free_memory_ft](#))(void *obj)
- typedef void(* [fmi2_step_finished_ft](#))(fmi2_component_environment_t env, [fmi2_status_t](#) status)
- typedef const char *(* [fmi2_get_types_platform_ft](#))()
- typedef const char *(* [fmi2_get_version_ft](#))()
- typedef [fmi2_status_t](#)(* [fmi2_set_debug_logging_ft](#))(fmi2_component_t, fmi2_boolean_t, size_t nCategories, const fmi2_string_t categories[])
- typedef fmi2_component_t(* [fmi2_instantiate_ft](#))(fmi2_string_t, [fmi2_type_t](#), fmi2_string_t, fmi2_string_t, const [fmi2_callback_functions_t](#) *, fmi2_boolean_t, fmi2_boolean_t)
- typedef void(* [fmi2_free_instance_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_setup_experiment_ft](#))(fmi2_component_t, fmi2_boolean_t, fmi2_real_t, fmi2_real_t, fmi2_boolean_t, fmi2_real_t)
- typedef [fmi2_status_t](#)(* [fmi2_enter_initialization_mode_ft](#))(fmi2_component_t)

- typedef [fmi2_status_t](#)(* [fmi2_exit_initialization_mode_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_terminate_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_reset_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_get_real_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, fmi2_real_t[])
- typedef [fmi2_status_t](#)(* [fmi2_get_integer_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, fmi2_integer_t[])
- typedef [fmi2_status_t](#)(* [fmi2_get_boolean_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, fmi2_boolean_t[])
- typedef [fmi2_status_t](#)(* [fmi2_get_string_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, fmi2_string_t[])
- typedef [fmi2_status_t](#)(* [fmi2_set_real_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, const fmi2_real_t[])
- typedef [fmi2_status_t](#)(* [fmi2_set_integer_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, const fmi2_integer_t[])
- typedef [fmi2_status_t](#)(* [fmi2_set_boolean_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, const fmi2_boolean_t[])
- typedef [fmi2_status_t](#)(* [fmi2_set_string_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, const fmi2_string_t[])
- typedef [fmi2_status_t](#)(* [fmi2_get_fmu_state_ft](#))(fmi2_component_t, fmi2_FMU_state_t *)
- typedef [fmi2_status_t](#)(* [fmi2_set_fmu_state_ft](#))(fmi2_component_t, fmi2_FMU_state_t)
- typedef [fmi2_status_t](#)(* [fmi2_free_fmu_state_ft](#))(fmi2_component_t, fmi2_FMU_state_t *)
- typedef [fmi2_status_t](#)(* [fmi2_serialized_fmu_state_size_ft](#))(fmi2_component_t, fmi2_FMU_state_t, size_t *)
- typedef [fmi2_status_t](#)(* [fmi2_serialize_fmu_state_ft](#))(fmi2_component_t, fmi2_FMU_state_t, fmi2_byte_t[], size_t)
- typedef [fmi2_status_t](#)(* [fmi2_de_serialize_fmu_state_ft](#))(fmi2_component_t, const fmi2_byte_t[], size_t, fmi2_FMU_state_t *)
- typedef [fmi2_status_t](#)(* [fmi2_get_directional_derivative_ft](#))(fmi2_component_t, const fmi2_value_reference_t[], size_t, const fmi2_value_reference_t[], size_t, const fmi2_real_t[], fmi2_real_t[])
- typedef [fmi2_status_t](#)(* [fmi2_enter_event_mode_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_new_discrete_states_ft](#))(fmi2_component_t, fmi2_event_info_t *)
- typedef [fmi2_status_t](#)(* [fmi2_enter_continuous_time_mode_ft](#))(fmi2_component_t)
- typedef [fmi2_status_t](#)(* [fmi2_completed_integrator_step_ft](#))(fmi2_component_t, fmi2_boolean_t, fmi2_boolean_t *, fmi2_boolean_t *)
- typedef [fmi2_status_t](#)(* [fmi2_set_time_ft](#))(fmi2_component_t, fmi2_real_t)
- typedef [fmi2_status_t](#)(* [fmi2_set_continuous_states_ft](#))(fmi2_component_t, const fmi2_real_t[], size_t)
- typedef [fmi2_status_t](#)(* [fmi2_get_derivatives_ft](#))(fmi2_component_t, fmi2_real_t[], size_t)
- typedef [fmi2_status_t](#)(* [fmi2_get_event_indicators_ft](#))(fmi2_component_t, fmi2_real_t[], size_t)

- typedef `fmi2_status_t`(* `fmi2_get_continuous_states_ft`)(fmi2_component_t, fmi2_real_t[], size_t)
- typedef `fmi2_status_t`(* `fmi2_get_nominals_of_continuous_states_ft`)(fmi2_component_t, fmi2_real_t[], size_t)
- typedef `fmi2_status_t`(* `fmi2_set_real_input_derivatives_ft`)(fmi2_component_t, const fmi2_value_reference_t[], size_t, const fmi2_integer_t[], const fmi2_real_t[])
- typedef `fmi2_status_t`(* `fmi2_get_real_output_derivatives_ft`)(fmi2_component_t, const fmi2_value_reference_t[], size_t, const fmi2_integer_t[], fmi2_real_t[])
- typedef `fmi2_status_t`(* `fmi2_do_step_ft`)(fmi2_component_t, fmi2_real_t, fmi2_real_t, fmi2_boolean_t)
- typedef `fmi2_status_t`(* `fmi2_cancel_step_ft`)(fmi2_component_t)
- typedef `fmi2_status_t`(* `fmi2_get_status_ft`)(fmi2_component_t, const `fmi2_status_kind_t`, `fmi2_status_t` *)
- typedef `fmi2_status_t`(* `fmi2_get_real_status_ft`)(fmi2_component_t, const `fmi2_status_kind_t`, fmi2_real_t *)
- typedef `fmi2_status_t`(* `fmi2_get_integer_status_ft`)(fmi2_component_t, const `fmi2_status_kind_t`, fmi2_integer_t *)
- typedef `fmi2_status_t`(* `fmi2_get_boolean_status_ft`)(fmi2_component_t, const `fmi2_status_kind_t`, fmi2_boolean_t *)
- typedef `fmi2_status_t`(* `fmi2_get_string_status_ft`)(fmi2_component_t, const `fmi2_status_kind_t`, fmi2_string_t *)

Enumerations

- enum `fmi2_status_t` { `fmi2_status_ok`, `fmi2_status_warning`, `fmi2_status_discard`, `fmi2_status_error`, `fmi2_status_fatal`, `fmi2_status_pending` }
- enum `fmi2_type_t` { `fmi2_model_exchange`, `fmi2_cosimulation` }
- enum `fmi2_status_kind_t` { `fmi2_do_step_status`, `fmi2_pending_status`, `fmi2_last_successful_time`, `fmi2_terminated` }

Functions

- `FMLIB_EXPORT` const char * `fmi2_status_to_string` (fmi2_status_t status)

9.29.1 Detailed Description

Mapping for the standard FMI 2.0 functions into `fmi2_` namespace.

Definition in file `fmi2_functions.h`.

9.30 src/Util/include/FMI2/fmi2_types.h File Reference

```
#include <FMI2/fmi2TypesPlatform.h>
```

Defines

Renaming of typedefs

- #define [fmi2Component](#) fmi2_component_t
- #define [fmi2ComponentEnvironment](#) fmi2_component_environment_t
- #define [fmi2FMUState](#) fmi2_FMU_state_t
- #define [fmi2ValueReference](#) fmi2_value_reference_t
- #define [fmi2Real](#) fmi2_real_t
- #define [fmi2Integer](#) fmi2_integer_t
- #define [fmi2Boolean](#) fmi2_boolean_t
- #define [fmi2Char](#) fmi2_char_t
- #define [fmi2String](#) fmi2_string_t
- #define [fmi2Byte](#) fmi2_byte_t

Enumerations

- enum [fmi2_boolean_enumeration](#) { [fmi2_true](#) = fmi2True, [fmi2_false](#) = fmi2False }

Functions

- static const [char](#) * [fmi2_get_types_platform](#) (void)

9.30.1 Detailed Description

Transformation of the standard FMI type names into fmi2_ prefixed.

Definition in file [fmi2_types.h](#).

9.31 src/Util/include/FMI2/fmi2_xml_callbacks.h File Reference

```
#include <fmilib_config.h>
```

Data Structures

- struct [fmi2_xml_callbacks](#)

XML callbacks are used to process parts of XML that are not handled by the library.

Typedefs

- typedef struct [fmi2_xml_callbacks](#) [fmi2_xml_callbacks_t](#)

XML handling callbacks

- typedef int(* [fmi2_xml_element_start_handle_ft](#))(void *context, const [char](#) *toolName, void *parent, const [char](#) *elm, const [char](#) **attr)

Handle start of an XML element within tool annotation in a SAX parser.

- typedef int(* [fmi2_xml_element_data_handle_ft](#))(void *context, const char *s, int len)

Handle data of an XML element within tool annotation in a SAX parser.

- typedef int(* [fmi2_xml_element_end_handle_ft](#))(void *context, const char *elm)

Handle end of an XML element within tool annotation in a SAX parser.

9.31.1 Detailed Description

Definition of [fmi2_xml_callbacks_t](#) and supporting functions

Definition in file [fmi2_xml_callbacks.h](#).

9.32 src/Util/include/JM/jm_callbacks.h File Reference

```
#include <stddef.h> #include <stdarg.h> #include <fmilib-  
_config.h> #include "jm_types.h"
```

Data Structures

- struct [jm_callbacks](#)

The callbacks struct is sent to all the modules in the library.

Defines

- #define [JM_MAX_ERROR_MESSAGE_SIZE](#) 2000

Maximum message size that can be stored in the [jm_callbacks](#) struct.

Typedefs

- typedef struct [jm_callbacks](#) [jm_callbacks](#)
- typedef void(* [jm_logger_f](#))([jm_callbacks](#) *c, [jm_string](#) module, [jm_log_level_enu_t](#) log_level, [jm_string](#) message)

Logger callback type.

Memory management callbacks

[jm_malloc_f](#), [jm_realloc_f](#), [jm_calloc_f](#), [jm_free_f](#) function types correspond to the standard C memory management functions

- typedef [jm_voidp](#)(* [jm_malloc_f](#))([size_t](#) size)
- typedef [jm_voidp](#)(* [jm_realloc_f](#))(void *ptr, [size_t](#) size)

Re-allocation function type.

- typedef `jm_voidp`(* `jm_malloc_f`)(`size_t` numitems, `size_t` itemsize)
Zero-initialized allocation function type.
- typedef void(* `jm_free_f`)(`jm_voidp` p)
Free memory function type.

Functions

- static `jm_string` `jm_get_last_error` (`jm_callbacks` *cb)
Get the last log message produced by the library.
- static void `jm_clear_last_error` (`jm_callbacks` *cb)
Clear the last generated log message.
- `FMILIB_EXPORT` void `jm_set_default_callbacks` (`jm_callbacks` *c)
Set the structure to be returned by `jm_get_default_callbacks()`.
- `FMILIB_EXPORT` `jm_callbacks` * `jm_get_default_callbacks` (void)
Get default callbacks. The function never returns NULL.
- `FMILIB_EXPORT` void `jm_default_logger` (`jm_callbacks` *c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)
The default logger implementation prints messages to stderr.
- `FMILIB_EXPORT` void `jm_log` (`jm_callbacks` *cb, const `char` *module, `jm_log_level_enu_t` log_level, const `char` *fmt,...)
Send a message to the logger function.
- `FMILIB_EXPORT` void `jm_log_v` (`jm_callbacks` *cb, const `char` *module, `jm_log_level_enu_t` log_level, const `char` *fmt, va_list ap)
Send a message to the logger function.
- `FMILIB_EXPORT` void `jm_log_fatal_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)
Send a fatal error message to the logger function. See `jm_log()` for details.
- `FMILIB_EXPORT` void `jm_log_fatal` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)
Send a fatal error message to the logger function. See `jm_log()` for details.
- `FMILIB_EXPORT` void `jm_log_error_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)
Send a error message to the logger function. See `jm_log()` for details.
- `FMILIB_EXPORT` void `jm_log_error` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)
Send a error message to the logger function. See `jm_log()` for details.
- `FMILIB_EXPORT` void `jm_log_warning_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)
Send a warning message to the logger function. See `jm_log()` for details.
- `FMILIB_EXPORT` void `jm_log_warning` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)
Send a warning message to the logger function. See `jm_log()` for details.
- `FMILIB_EXPORT` void `jm_log_info_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)
Send an info message to the logger function. See `jm_log()` for details.

- **FMILIB_EXPORT** void `jm_log_info` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)

Send an info message to the logger function. See `jm_log()` for details.

- **FMILIB_EXPORT** void `jm_log_verbose_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)

Send a verbose message to the logger function. See `jm_log()` for details.

- **FMILIB_EXPORT** void `jm_log_verbose` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)

Send a verbose message to the logger function. See `jm_log()` for details.

- static void `jm_log_debug_v` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt, va_list ap)

Send a debug message to the logger function. See `jm_log()` for details.

- static void `jm_log_debug` (`jm_callbacks` *cb, const `char` *module, const `char` *fmt,...)

Send a debug message to the logger function. See `jm_log()` for details.

9.32.1 Detailed Description

Definition of `jm_callbacks` and supporting functions

Definition in file `jm_callbacks.h`.

9.33 src/Util/include/JM/jm_named_ptr.h File Reference

```
#include "jm_vector.h" #include "jm_callbacks.h"
```

Data Structures

- struct `jm_named_ptr`

Name and object pointer pair.

Defines

- #define `jm_diff_named`(a, b) strcmp(a.name,b.name)

Typedefs

- typedef struct `jm_named_ptr` `jm_named_ptr`

Name and object pointer pair.

Functions

- `jm_named_ptr jm_named_alloc (jm_string name, size_t size, size_t nameoffset, jm_callbacks *c)`
Allocate memory for the object and the name string and sets pointer to it packed together with the name pointer.
- `jm_named_ptr jm_named_alloc_v (jm_vector(char)*name, size_t size, size_t nameoffset, jm_callbacks *c)`
Same as `jm_named_alloc()` but name is given as a `jm_vector(char)` pointer.
- `static void jm_named_free (jm_named_ptr np, jm_callbacks *c)`
Free the memory allocated for the object pointed by `jm_named_ptr`.
- `jm_vector_declare_template (jm_named_ptr) jm_define_comp_f(jm_compare_named)`
Helper to construct comparison operation.
- `static jm_diff_named void jm_named_vector_free_data (jm_vector(jm_named_ptr)*v)`
Release the data allocated by the items in a vector and then clears the memory used by the vector as well.
- `static void jm_named_vector_free (jm_vector(jm_named_ptr)*v)`
Release the data allocated by the items in a vector and then clears the memory used by the vector as well.

9.33.1 Detailed Description

Definition of `jm_named_ptr` and supporting functions

Definition in file `jm_named_ptr.h`.

9.33.2 Define Documentation

9.33.2.1 `#define jm_diff_named( a, b ) strcmp(a.name,b.name)`

9.34 src/Util/include/JM/jm_portability.h File Reference

```
#include <fmilib_config.h>      #include "jm_callbacks.h" ×
#include <dlfcn.h> #include "jm_types.h"
```

Defines

- `#define DLL_HANDLE void*`

Typedefs

- `typedef void * jm_dll_function_ptr`
A function pointer as returned when DLL symbol is loaded.

Functions

- `DLL_HANDLE jm_portability_load_dll_handle` (const char *dll_file_path)
Load a dll/so library into the process and return a handle.
- `jm_status_enu_t jm_portability_free_dll_handle` (DLL_HANDLE dll_handle)
Unload a Dll and release the handle.
- `jm_status_enu_t jm_portability_load_dll_function` (DLL_HANDLE dll_handle, char *dll_function_name, jm_dll_function_ptr *dll_function_ptrptr)
Find a function in the Dll and return a function pointer.
- `char * jm_portability_get_last_dll_error` (void)
Return error associated with Dll handling.
- `jm_status_enu_t jm_portability_get_current_working_directory` (char *buffer, size_t len)
Get current working directory name.
- `jm_status_enu_t jm_portability_set_current_working_directory` (const char *cwd)
Set current working directory.
- `const char * jm_get_system_temp_dir` ()
Get system-wide temporary directory.
- `char * jm_mktemp` (char *tmpl)
Create a unique file name.
- `char * jm_get_dir_abspath` (jm_callbacks *cb, const char *dir, char *outPath, size_t len)
Get absolute path to an existing directory.
- `char * jm_mk_temp_dir` (jm_callbacks *cb, const char *systemTempDir, const char *tempPrefix)
Create a unique temporary directory.
- `char * jm_create_URL_from_abs_path` (jm_callbacks *cb, const char *absPath)
Create a file:// URL from absolute path.
- `jm_status_enu_t jm_mkdir` (jm_callbacks *cb, const char *dir)
Make a directory.
- `jm_status_enu_t jm_rmdir` (jm_callbacks *cb, const char *dir)
Remove directory and all it contents.
- `FMILIB_EXPORT int jm_vsnprintf` (char *str, size_t size, const char *fmt, va_list al)
C89 compatible implementation of C99 vsnprintf.
- `FMILIB_EXPORT int jm_snprintf` (char *str, size_t size, const char *fmt,...)
C89 compatible implementation of C99 snprintf.

9.34.1 Detailed Description

Handling platform specific defines and functions.

Definition in file `jm_portability.h`.

9.34.2 Define Documentation

9.34.2.1 #define DLL_HANDLE void*

Definition at line 26 of file jm_portability.h.

9.35 src/Util/include/JM/jm_stack.h File Reference

```
#include "jm_vector.h"
```

Defines

- #define [jm_stack\(T\)](#) [jm_mangle\(jm_stack, T\)](#)
A basic stack of items.
- #define [jm_stack_alloc\(T\)](#) [jm_mangle\(jm_stack_alloc, T\)](#)
Allocates a stack with the given reserved memory.
- #define [jm_stack_free\(T\)](#) [jm_mangle\(jm_stack_free, T\)](#)
Release memory allocated for a stack.
- #define [jm_stack_init\(T\)](#) [jm_mangle\(jm_stack_init, T\)](#)
Initializes a [jm_stack](#) allocated on stack.
- #define [jm_stack_free_data\(T\)](#) [jm_mangle\(jm_stack_free_data, T\)](#)
Releases memory allocated for stack data.
- #define [jm_stack_get_size\(T\)](#) [jm_mangle\(jm_stack_get_size, T\)](#)
Get the number of elements in the stack.
- #define [jm_stack_reserve\(T\)](#) [jm_mangle\(jm_stack_reserve, T\)](#)
Preallocate memory for the stack (to speed up consequent push).
- #define [jm_stack_push\(T\)](#) [jm_mangle\(jm_stack_push, T\)](#)
Put an element on the stack.
- #define [jm_stack_is_empty\(T\)](#) [jm_mangle\(jm_stack_is_empty, T\)](#)
- #define [jm_stack_pop\(T\)](#) [jm_mangle\(jm_stack_pop, T\)](#)
- #define [jm_stack_top\(T\)](#) [jm_mangle\(jm_stack_top, T\)](#)
- #define [jm_stack_foreach\(T\)](#) [jm_mangle\(jm_stack_foreach, T\)](#)
- #define [JM_STACK_MINIMAL_CAPACITY](#) [JM_VECTOR_MINIMAL_CAPACITY](#)
- #define [JM_STACK_MAX_MEMORY_CHUNK](#) [JM_VECTOR_MAX_MEMORY_CHUNK](#)
- #define [jm_stack_declare_template\(T\)](#)

9.36 src/Util/include/JM/jm_string_set.h File Reference

```
#include <string.h> #include "jm_types.h" #include "jm_vector.h"
```

Typedefs

- typedef struct jm_vector_jm_string [jm_string_set](#)
Set of string is based on a vector.

Functions

- static [jm_string](#) [jm_string_set_find](#) ([jm_string_set](#) *s, [jm_string](#) str)
Find a string in a set.
- static [jm_string](#) [jm_string_set_put](#) ([jm_string_set](#) *s, [jm_string](#) str)
Put an element in the set if it is not there yet.

9.36.1 Detailed Description

Definition of [jm_string_set](#) and supporting functions

Definition in file [jm_string_set.h](#).

9.37 src/Util/include/JM/jm_types.h File Reference

```
#include <fmilib_config.h>
```

Data Structures

- struct [jm_name_ID_map_t](#)
Mapping between a string and an integer ID.

Typedefs

- typedef const [char](#) * [jm_string](#)
A constant string.
- typedef void * [jm_voidp](#)
A void pointer.
- typedef struct [jm_name_ID_map_t](#) [jm_name_ID_map_t](#)
Mapping between a string and an integer ID.

Enumerations

- enum [jm_status_enu_t](#) { [jm_status_error](#) = -1, [jm_status_success](#) = 0, [jm_status-warning](#) = 1 }
Return status codes.

- enum `jm_log_level_enu_t` { `jm_log_level_nothing` = 0, `jm_log_level_fatal`, `jm_log_level_error`, `jm_log_level_warning`, `jm_log_level_info`, `jm_log_level_verbose`, `jm_log_level_debug`, `jm_log_level_all` }

Log levels supported via the logger functions in [jm_callbacks](#).

Functions

- `FMILIB_EXPORT` const char * `jm_log_level_to_string` (`jm_log_level_enu_t` level)

Convert log level into a string.

9.37.1 Detailed Description

Types used in the Utils module.

Definition in file [jm_types.h](#).

9.38 src/Util/include/JM/jm_vector.h File Reference

```
#include <assert.h>  #include <string.h>  #include "jm_callbacks.h"
```

Defines

- #define `jm_mangle_ex`(name, type) name##_ ##type
jm_mange macro is used to construct names for the template instances Extra level (jm_mange_ex) is needed to force argument expansion (pre-scan)
- #define `jm_mangle`(name, type) `jm_mangle_ex`(name,type)
- #define `jm_vector`(T) `jm_mangle`(`jm_vector`, T)
jm_vector(T) is the type name (i.e., to be used as `jm_vector(int) vi`;)
- #define `JM_VECTOR_MINIMAL_CAPACITY` 16
- #define `JM_VECTOR_MAX_MEMORY_CHUNK` 1024
- #define `jm_vector_declare_template`(T)
- #define `jm_diff_name`(a, b) strcmp(a.name,b.name)

Functions

- `jm_vector_declare_template` (char) static `jm_string` `jm_vector_char2string`(`jm_vector`(char)*v)
- `jm_vector_declare_template` (int) `jm_vector_declare_template`(double) `jm_vector_declare_template`(`jm_voidp`) `jm_vector_declare_template`(size_t) `jm_vector_declare_template`(`jm_string`) `jm_vector_declare_template`(`jm_name_ID_map_t`) `jm_define_comp_f`(`jm_compare_voidp`)

- int [jm_diff](#) [jm_define_comp_f](#) (jm_compare_int, int, [jm_diff](#)) [jm_define_comp_f](#)(jm_compare_char
- int [jm_diff](#) [jm_diff](#) [jm_define_comp_f](#) (jm_compare_double, double, [jm_diff](#)) [jm_define_comp_f](#)(jm_compare_size_t
- int [jm_diff](#) [jm_diff](#) [jm_diff](#) [jm_define_comp_f](#) (jm_compare_string, [jm_string](#), strcmp) [jm_define_comp_f](#)(jm_compare_name

Variables

- int [jm_diff](#) char
- int [jm_diff](#) [jm_diff](#) size_t
- int [jm_diff](#) [jm_diff](#) [jm_diff](#) [jm_name_ID_map_t](#)
- int [jm_diff](#) [jm_diff](#) [jm_diff](#) [jm_diff_name](#)

Vector handling functions.

Allocates a vector on heap with the specified size and specified number of preallocated items (can be larger than size).

extern [jm_vector](#)(T)* [jm_vector_alloc](#)(T)(size_t size, size_t capacity, jm_callbacks*c);
Note that there is no need to call [jm_vector_init](#) for a vector allocated with this function.

Parameters

<i>size</i>	- initial size of the vector, can be 0
<i>capacity</i>	- initial capacity of the vector, can be 0. At least <code>initSize</code> elements are allocated.
<i>c</i>	- jm_callbacks callbacks, can be zero

Returns

Newly allocated vector

- #define [jm_vector_alloc](#)(T) [jm_mangle](#)([jm_vector_alloc](#), T)
 - #define [jm_vector_free](#)(T) [jm_mangle](#)([jm_vector_free](#), T)
 - #define [jm_vector_init](#)(T) [jm_mangle](#)([jm_vector_init](#), T)
- jm_vector_init initializes a vector allocated on stack.*
- #define [jm_vector_free_data](#)(T) [jm_mangle](#)([jm_vector_free_data](#), T)
 - #define [jm_vector_get_size](#)(T) [jm_mangle](#)([jm_vector_get_size](#), T)
 - #define [jm_vector_get_item](#)(T) [jm_mangle](#)([jm_vector_get_item](#), T)
 - #define [jm_vector_get_itemp](#)(T) [jm_mangle](#)([jm_vector_get_itemp](#), T)
 - #define [jm_vector_get_last](#)(T) [jm_mangle](#)([jm_vector_get_last](#), T)
 - #define [jm_vector_get_lastp](#)(T) [jm_mangle](#)([jm_vector_get_lastp](#), T)
 - #define [jm_define_comp_f](#)(F, T, COMPAR_OP)
- A convenience macro for comparison function definition.*
- #define [jm_diff](#)(first, second) (int)(first-second)
 - #define [jm_vector_find](#)(T) [jm_mangle](#)([jm_vector_find](#), T)

jm_vector_find functions use linear search to find items in a vector. *JM_COMPAR_OP* is used for comparison.

- #define [jm_vector_find_index](#)(T) [jm_mangle](#)([jm_vector_find_index](#), T)
- #define [jm_vector_qsort](#)(T) [jm_mangle](#)([jm_vector_qsort](#), T)
- #define [jm_vector_bsearch](#)(T) [jm_mangle](#)([jm_vector_bsearch](#), T)
- #define [jm_vector_bsearch_index](#)(T) [jm_mangle](#)([jm_vector_bsearch_index](#), T)
- #define [jm_vector_set_item](#)(T) [jm_mangle](#)([jm_vector_set_item](#), T)
- #define [jm_vector_zero](#)(T) [jm_mangle](#)([jm_vector_zero](#), T)
- #define [jm_vector_resize](#)(T) [jm_mangle](#)([jm_vector_resize](#), T)
- #define [jm_vector_reserve](#)(T) [jm_mangle](#)([jm_vector_reserve](#), T)
- #define [jm_vector_copy](#)(T) [jm_mangle](#)([jm_vector_copy](#), T)
- #define [jm_vector_clone](#)(T) [jm_mangle](#)([jm_vector_clone](#), T)
- #define [jm_vector_append](#)(T) [jm_mangle](#)([jm_vector_append](#), T)
- #define [jm_vector_insert](#)(T) [jm_mangle](#)([jm_vector_insert](#), T)
- #define [jm_vector_remove_item](#)(T) [jm_mangle](#)([jm_vector_remove_item](#), T)
- #define [jm_vector_resize1](#)(T) [jm_mangle](#)([jm_vector_resize1](#), T)
- #define [jm_vector_push_back](#)(T) [jm_mangle](#)([jm_vector_push_back](#), T)
- #define [jm_vector_foreach](#)(T) [jm_mangle](#)([jm_vector_foreach](#), T)
- #define [jm_vector_foreach_c](#)(T) [jm_mangle](#)([jm_vector_foreach_c](#), T)
- typedef int(* [jm_compare_ft](#))(const void *, const void *)

Function type for item comparison. Can be generated with [jm_define_comp_f](#).

9.38.1 Detailed Description

Definition of [jm_vector](#) and supporting functions

Definition in file [jm_vector.h](#).

9.38.2 Define Documentation

9.38.2.1 #define [jm_diff_name](#)(a, b) strcmp(a.name,b.name)

9.39 src/Util/include/JM/jm_vector_template.h File Reference

Vector template definition.

```
#include <stdlib.h>  #include <string.h>  #include "jm_
vector.h"
```

Defines

- #define [jm_vector_ptr2index](#)(T) [jm_mangle](#)([jm_vector_ptr2index](#), T)

Functions

- `jm_vector` (JM_TEMPLATE_INSTANCE_TYPE)*`jm_vector_alloc`(JM_TEMPLATE_INSTANCE_TYPE)(`size_t` size

Variables

- `size_t` capacity

9.39.1 Detailed Description

Vector template definition. This file is supposed to be included into a C-file that instantiate the template. `jm_vector.h` must be included before this file. It expects JM_TEMPLATE_INSTANCE_TYPE to be defined to the template type to be instantiated.

Definition in file `jm_vector_template.h`.

9.39.2 Define Documentation

9.39.2.1 `#define jm_vector_ptr2index( T ) jm_mangle(jm_vector_ptr2index, T)`

9.40 Test/compress_test_fmu_zip.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <JM/jm_
types.h> #include <JM/jm_callbacks.h> #include <FMI/fmi_
zip_zip.h>
```

Functions

- static void `importlogger` (`jm_callbacks` *c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)
- int `main` (int argc, `char` *argv[])

9.40.1 Function Documentation

9.40.1.1 static void `importlogger` (`jm_callbacks` * c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message) [static]

Definition at line 22 of file `compress_test_fmu_zip.c`.

9.40.1.2 int `main` (int argc, `char` * argv[])

Definition at line 27 of file `compress_test_fmu_zip.c`.

9.41 Test/FMI1/compress_test_fmu_zip.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <JM/jm_
types.h> #include <JM/jm_callbacks.h> #include <FMI/fmi_
zip_zip.h>
```

Functions

- static void `importlogger` (`jm_callbacks` *c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)
- int `main` (int argc, `char` *argv[])

9.41.1 Function Documentation

9.41.1.1 static void `importlogger` (`jm_callbacks` * c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message) [static]

Definition at line 22 of file `compress_test_fmu_zip.c`.

9.41.1.2 int `main` (int argc, `char` * argv[])

Definition at line 27 of file `compress_test_fmu_zip.c`.

9.42 Test/FMI1/fmi1_capi_cs_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.-
h> #include <string.h> #include <errno.h> #include <-
JM/jm_types.h> #include <JM/jm_portability.h> #include
<FMI1/fmil_types.h> #include <FMI1/fmil_functions.h> x
#include <FMI1/fmil_capi.h> #include <JM/jm_callbacks.-
h> #include <fmu_dummy/fmul_model_defines.h> #include
"config_test.h"
```

Defines

- #define `MODEL_IDENTIFIER` FMU_DUMMY_CS_MODEL_IDENTIFIER
- #define `INSTANCE_NAME` "Test Model"

Functions

- void `importlogger` (`jm_callbacks` *c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)

- void [fmi1logger](#) (fmi1_component_t c, fmi1_string_t instanceName, [fmi1_status_t](#) status, fmi1_string_t category, fmi1_string_t message,...)
- void [do_exit](#) (int code)
- int [test_create_dllfmu](#) ()
Tests fmi1_capi_create_dllfmu.
- int [test_load_dll](#) ()
Tests fmi1_capi_load_dll.
- int [test_load_dll_fcn](#) ()
Tests fmi1_capi_load_fcn.
- int [test_fmi_get_version](#) ()
Tests fmi1_capi_get_version.
- int [test_get_types_platform](#) ()
Tests fmi1_capi_get_types_platform.
- int [test_instantiate_slave](#) ()
Tests fmi1_capi_instantiate_slave.
- int [test_initialize_slave](#) ()
Tests fmi1_capi_initialize_slave.
- int [test_set_debug_logging](#) ()
Tests fmi1_capi_set_debug_logging.
- int [test_cancel_step](#) ()
Tests fmi1_capi_cancel_step.
- int [test_do_step](#) ()
Tests fmi1_capi_do_step.
- int [test_get_status](#) ()
Tests fmi1_capi_get_status.
- int [test_get_real_status](#) ()
Tests fmi1_capi_get_real_status.
- int [test_get_integer_status](#) ()
Tests fmi1_capi_get_integer_status.
- int [test_get_boolean_status](#) ()
Tests fmi1_capi_get_boolean_status.
- int [test_get_string_status](#) ()
Tests fmi1_capi_get_string_status.
- int [test_set_get_string](#) ()
Tests fmi1_capi_set_string and fmi1_capi_get_string Some values are set with fmi1_capi_set_string. The same values are retrived with fmi1_capi_get_string and tested to be the same as thoughts that were set.
- int [test_set_get_integer](#) ()
Tests fmi1_capi_set_integer and fmi1_capi_get_integer Some values are set with fmi1_capi_set_integer. The same values are retrived with fmi1_capi_get_integer and tested to be the same as thoughts that were set.
- int [test_set_get_boolean](#) ()
Tests fmi1_capi_set_boolean and fmi1_capi_get_boolean Some values are set with fmi1_capi_set_boolean. The same values are retrived with fmi1_capi_get_boolean and tested to be the same as thoughts that were set.

- int [test_set_get_real](#) ()
Tests fmi1_capi_set_real and fmi1_capi_get_real Some values are set with fmi1_capi_set_real. The same values are retrived with fmi1_capi_get_real and tested to be the same as thoughts that were set.
- int [test_reset_slave](#) ()
Tests fmi1_capi_reset_slave.
- int [test_set_real_input_derivatives](#) ()
Tests fmi1_capi_set_real_input_derivatives fmi1_capi_set_real_input_derivatives returns fmiError if wrong values are set. The values that are set are tested inside the DLL.
- int [test_get_real_output_derivatives](#) ()
Tests fmi1_capi_get_real_output_derivatives The output values from fmi1_capi_get_real_output_derivatives is expected to have a special value that is coded in the DLL.
- int [test_terminate_slave](#) ()
Tests fmi1_capi_terminate_slave.
- int [test_free_slave_instance](#) ()
Tests fmi1_capi_free_slave_instance.
- int [test_free_dll](#) ()
Tests fmi1_capi_free_dll.
- int [test_destroy_dllfmu](#) ()
Tests fmi1_capi_destroy_dllfmu.
- int [main](#) (int argc, char *argv[])
Tests the C-API for FMI 1.0 Co-Simulation. The tests are performed using a test-dll. The functions are called and the values are set or returned are validated either in the test function(output functions) or inside the dll(input functions). If any error occures, the program exits.

Variables

- fmi1_capi_t * [fmu](#)
- [jm_callbacks](#) * [callbacks](#) = 0

9.42.1 Define Documentation

9.42.1.1 #define MODEL_IDENTIFIER FMU_DUMMY_CS_MODEL_IDENTIFIER

Definition at line 32 of file fmi1_capi_cs_test.c.

9.42.1.2 #define INSTANCE_NAME "Test Model"

Definition at line 35 of file fmi1_capi_cs_test.c.

9.42.2 Function Documentation

9.42.2.1 `void importlogger (jm_callbacks * c, jm_string module,
jm_log_level_enu_t log_level, jm_string message)`

Definition at line 40 of file fmi1_capi_cs_test.c.

9.42.2.2 `void fmilogger (fmi1_component_t c, fmi1_string_t instanceName, fmi1_status_t
status, fmi1_string_t category, fmi1_string_t message, ...)`

Definition at line 46 of file fmi1_capi_cs_test.c.

9.42.2.3 `void do_exit ( int code )`

Definition at line 60 of file fmi1_capi_cs_test.c.

9.42.2.4 `int test_create_dllfmu ( )`

Tests fmi1_capi_create_dllfmu.

Definition at line 73 of file fmi1_capi_cs_test.c.

9.42.2.5 `int test_load_dll ( )`

Tests fmi1_capi_load_dll.

Definition at line 107 of file fmi1_capi_cs_test.c.

9.42.2.6 `int test_load_dll_fcn ( )`

Tests fmi1_capi_load_fcn.

Definition at line 126 of file fmi1_capi_cs_test.c.

9.42.2.7 `int test_fmi_get_version ( )`

Tests fmi1_capi_get_version.

Definition at line 146 of file fmi1_capi_cs_test.c.

9.42.2.8 `int test_get_types_platform ( )`

Tests fmi1_capi_get_types_platform.

Definition at line 164 of file fmi1_capi_cs_test.c.

9.42.2.9 int test_instantiate_slave ()

Tests fmi1_capi_instantiate_slave.

Definition at line 182 of file fmi1_capi_cs_test.c.

9.42.2.10 int test_initialize_slave ()

Tests fmi1_capi_initialize_slave.

Definition at line 211 of file fmi1_capi_cs_test.c.

9.42.2.11 int test_set_debug_logging ()

Tests fmi1_capi_set_debug_logging.

Definition at line 236 of file fmi1_capi_cs_test.c.

9.42.2.12 int test_cancel_step ()

Tests fmi1_capi_cancel_step.

Definition at line 253 of file fmi1_capi_cs_test.c.

9.42.2.13 int test_do_step ()

Tests fmi1_capi_do_step.

Definition at line 270 of file fmi1_capi_cs_test.c.

9.42.2.14 int test_get_status ()

Tests fmi1_capi_get_status.

Definition at line 294 of file fmi1_capi_cs_test.c.

9.42.2.15 int test_get_real_status ()

Tests fmi1_capi_get_real_status.

Definition at line 315 of file fmi1_capi_cs_test.c.

9.42.2.16 int test_get_integer_status ()

Tests fmi1_capi_get_integer_status.

Definition at line 335 of file fmi1_capi_cs_test.c.

9.42.2.17 int test_get_boolean_status ()

Tests fmi1_capi_get_boolean_status.

Definition at line 355 of file fmi1_capi_cs_test.c.

9.42.2.18 int test_get_string_status ()

Tests fmi1_capi_get_string_status.

Definition at line 375 of file fmi1_capi_cs_test.c.

9.42.2.19 int test_set_get_string ()

Tests fmi1_capi_set_string and fmi1_capi_get_string Some values are set with fmi1_capi_set_string. The same values are retrieved with fmi1_capi_get_string and tested to be the same as thoughts that were set.

Definition at line 395 of file fmi1_capi_cs_test.c.

9.42.2.20 int test_set_get_integer ()

Tests fmi1_capi_set_integer and fmi1_capi_get_integer Some values are set with fmi1_capi_set_integer. The same values are retrieved with fmi1_capi_get_integer and tested to be the same as thoughts that were set.

Definition at line 440 of file fmi1_capi_cs_test.c.

9.42.2.21 int test_set_get_boolean ()

Tests fmi1_capi_set_boolean and fmi1_capi_get_boolean Some values are set with fmi1_capi_set_boolean. The same values are retrieved with fmi1_capi_get_boolean and tested to be the same as thoughts that were set.

Definition at line 485 of file fmi1_capi_cs_test.c.

9.42.2.22 int test_set_get_real ()

Tests fmi1_capi_set_real and fmi1_capi_get_real Some values are set with fmi1_capi_set_real. The same values are retrieved with fmi1_capi_get_real and tested to be the same as thoughts that were set.

Definition at line 530 of file fmi1_capi_cs_test.c.

9.42.2.23 int test_reset_slave ()

Tests fmi1_capi_reset_slave.

Definition at line 575 of file fmi1_capi_cs_test.c.

9.42.2.24 int test_set_real_input_derivatives ()

Tests fmi1_capi_set_real_input_derivatives fmi1_capi_set_real_input_derivatives returns fmiError if wrong values are set. The values that are set are tested inside the DLL.

Definition at line 592 of file fmi1_capi_cs_test.c.

9.42.2.25 int test_get_real_output_derivatives ()

Tests fmi1_capi_get_real_output_derivatives The output values from fmi1_capi_get_real_output_derivatives is expected to have a special value that is coded in the DLL.

Definition at line 623 of file fmi1_capi_cs_test.c.

9.42.2.26 int test_terminate_slave ()

Tests fmi1_capi_terminate_slave.

Definition at line 670 of file fmi1_capi_cs_test.c.

9.42.2.27 int test_free_slave_instance ()

Tests fmi1_capi_free_slave_instance.

Definition at line 687 of file fmi1_capi_cs_test.c.

9.42.2.28 int test_free_dll ()

Tests fmi1_capi_free_dll.

Definition at line 698 of file fmi1_capi_cs_test.c.

9.42.2.29 int test_destroy_dllfmu ()

Tests fmi1_capi_destroy_dllfmu.

Definition at line 709 of file fmi1_capi_cs_test.c.

9.42.2.30 int main (int argc, char * argv[])

Tests the C-API for FMI 1.0 Co-Simulation. The tests are performed using a test-dll. The functions are called and the values are set or returned are validated either in the test function(output functions) or inside the dll(input functions). If any error occurs, the program exits.

Definition at line 721 of file fmi1_capi_cs_test.c.

9.42.3 Variable Documentation

9.42.3.1 fmi1_capi_t* fmu

Definition at line 37 of file fmi1_capi_cs_test.c.

9.42.3.2 jm_callbacks* callbacks = 0

Definition at line 67 of file fmi1_capi_cs_test.c.

9.43 Test/FMI1/fmi1_capi_me_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.-
h> #include <string.h> #include <errno.h> #include <fmilib.-
h> #include "config_test.h" #include <FMI1/fmil_capi.h> ×
#include <fmu_dummy/fmul_model_defines.h>
```

Defines

- #define [MODEL_IDENTIFIER](#) FMU_DUMMY_ME_MODEL_IDENTIFIER
- #define [INSTANCE_NAME](#) "Test Model"

Functions

- void [importlogger](#) ([jm_callbacks](#) *c, [jm_string](#) module, [jm_log_level_enu_t](#) log_level, [jm_string](#) message)
- void [fmilogger](#) ([fmi1_component_t](#) c, [fmi1_string_t](#) instanceName, [fmi1_status_t](#) status, [fmi1_string_t](#) category, [fmi1_string_t](#) message,...)
- void [do_exit](#) (int code)
- int [test_create_dllfmu](#) ()
Tests fmi1_capi_create_dllfmu.
- int [test_load_dll](#) ()
Tests fmi1_capi_load_dll.
- int [test_load_dll_fcn](#) ()
Tests fmi1_capi_load_fcn.
- int [test_fmi_get_version](#) ()
Tests fmi1_capi_get_version.
- int [test_fmi_get_model_types_platform](#) ()
Tests fmi1_capi_get_model_types_platform.
- int [test_instantiate_model](#) ()
Tests fmi1_capi_instantiate_model.
- int [test_fmi_set_time](#) ()
Tests fmi1_capi_set_time.

- int [test_set_continuous_states](#) ()
Tests fmi1_capi_set_continuous_states.
- int [test_initialize](#) ()
Tests fmi1_capi_initialize.
- int [test_completed_integrator_step](#) ()
Tests fmi1_capi_completed_integrator_step.
- int [test_get_derivatives](#) ()
Tests fmi1_capi_get_derivatives.
- int [test_get_event_indicators](#) ()
- int [test_event_update](#) ()
Tests fmi1_capi_eventUpdate.
- int [test_get_continuous_states](#) ()
Tests fmi1_capi_get_continuous_states.
- int [test_get_nominal_continuous_states](#) ()
Tests fmi1_capi_get_nominal_continuous_states.
- int [test_get_state_value_references](#) ()
Tests fmi1_capi_get_state_value_references.
- int [test_set_debug_logging](#) ()
Tests fmi1_capi_set_debug_logging.
- int [test_set_get_string](#) ()
Tests fmi1_capi_set_string and fmi1_capi_get_string Some values are set with fmi1_capi_set_string. The same values are retrived with fmi1_capi_get_string and tested to be the same as thoughts that were set.
- int [test_set_get_integer](#) ()
Tests fmi1_capi_set_integer and fmi1_capi_get_integer Some values are set with fmi1_capi_set_integer. The same values are retrived with fmi1_capi_get_integer and tested to be the same as thoughts that were set.
- int [test_set_get_boolean](#) ()
Tests fmi1_capi_set_boolean and fmi1_capi_get_boolean Some values are set with fmi1_capi_set_boolean. The same values are retrived with fmi1_capi_get_boolean and tested to be the same as thoughts that were set.
- int [test_set_get_real](#) ()
Tests fmi1_capi_set_real and fmi1_capi_get_real Some values are set with fmi1_capi_set_real. The same values are retrived with fmi1_capi_get_real and tested to be the same as thoughts that were set.
- int [test_terminate](#) ()
Tests fmi1_capi_terminate.
- int [test_free_model_instance](#) ()
Tests fmi1_capi_free_model_instance.
- int [test_free_dll](#) ()
Tests fmi1_capi_free_dll.
- int [test_destroy_dllfmu](#) ()
Tests fmi1_capi_destroy_dllfmu.
- int [main](#) (int argc, char *argv[])

Tests the C-API for FMI 1.0 Model Exchange. The tests are performed using a test-dll. The functions are called and the values are set or returned are validated either in the test function(output functions) or inside the dll(input functions). If any error occurs, the program exits.

Variables

- fmi1_capi_t * [fmu](#)
- jm_callbacks * [callbacks](#)

9.43.1 Define Documentation

9.43.1.1 **#define MODEL_IDENTIFIER FMU_DUMMY_ME_MODEL_IDENTIFIER**

Definition at line 34 of file fmi1_capi_me_test.c.

9.43.1.2 **#define INSTANCE_NAME "Test Model"**

Definition at line 37 of file fmi1_capi_me_test.c.

9.43.2 Function Documentation

9.43.2.1 **void importlogger (jm_callbacks * c, jm_string module, jm_log_level_enu_t log_level, jm_string message)**

Definition at line 42 of file fmi1_capi_me_test.c.

9.43.2.2 **void fmilogger (fmi1_component_t c, fmi1_string_t instanceName, fmi1_status_t status, fmi1_string_t category, fmi1_string_t message, ...)**

Definition at line 48 of file fmi1_capi_me_test.c.

9.43.2.3 **void do_exit (int code)**

Definition at line 62 of file fmi1_capi_me_test.c.

9.43.2.4 **int test_create_dllfmu ()**

Tests fmi1_capi_create_dllfmu.

Definition at line 75 of file fmi1_capi_me_test.c.

9.43.2.5 int test_load_dll ()

Tests fmi1_capi_load_dll.

Definition at line 108 of file fmi1_capi_me_test.c.

9.43.2.6 int test_load_dll_fcn ()

Tests fmi1_capi_load_fcn.

Definition at line 127 of file fmi1_capi_me_test.c.

9.43.2.7 int test_fmi_get_version ()

Tests fmi1_capi_get_version.

Definition at line 147 of file fmi1_capi_me_test.c.

9.43.2.8 int test_fmi_get_model_types_platform ()

Tests fmi1_capi_get_model_types_platform.

Definition at line 165 of file fmi1_capi_me_test.c.

9.43.2.9 int test_instantiate_model ()

Tests fmi1_capi_instantiate_model.

Definition at line 183 of file fmi1_capi_me_test.c.

9.43.2.10 int test_fmi_set_time ()

Tests fmi1_capi_set_time.

Definition at line 200 of file fmi1_capi_me_test.c.

9.43.2.11 int test_set_continuous_states ()

Tests fmi1_capi_set_continuous_states.

Definition at line 217 of file fmi1_capi_me_test.c.

9.43.2.12 int test_initialize ()

Tests fmi1_capi_initialize.

Definition at line 246 of file fmi1_capi_me_test.c.

9.43.2.13 int test_completed_integrator_step ()

Tests fmi1_capi_completed_integrator_step.

Definition at line 277 of file fmi1_capi_me_test.c.

9.43.2.14 int test_get_derivatives ()

Tests fmi1_capi_get_derivatives.

Definition at line 299 of file fmi1_capi_me_test.c.

9.43.2.15 int test_get_event_indicators ()

Definition at line 320 of file fmi1_capi_me_test.c.

9.43.2.16 int test_event_update ()

Tests fmi1_capi_eventUpdate.

Definition at line 345 of file fmi1_capi_me_test.c.

9.43.2.17 int test_get_continuous_states ()

Tests fmi1_capi_get_continuous_states.

Definition at line 373 of file fmi1_capi_me_test.c.

9.43.2.18 int test_get_nominal_continuous_states ()

Tests fmi1_capi_get_nominal_continuous_states.

Definition at line 398 of file fmi1_capi_me_test.c.

9.43.2.19 int test_get_state_value_references ()

Tests fmi1_capi_get_state_value_references.

Definition at line 423 of file fmi1_capi_me_test.c.

9.43.2.20 int test_set_debug_logging ()

Tests fmi1_capi_set_debug_logging.

Definition at line 448 of file fmi1_capi_me_test.c.

9.43.2.21 int test_set_get_string ()

Tests fmi1_capi_set_string and fmi1_capi_get_string Some values are set with fmi1_capi_set_string. The same values are retrieved with fmi1_capi_get_string and tested to be the same as thoughts that were set.

Definition at line 465 of file fmi1_capi_me_test.c.

9.43.2.22 int test_set_get_integer ()

Tests fmi1_capi_set_integer and fmi1_capi_get_integer Some values are set with fmi1_capi_set_integer. The same values are retrieved with fmi1_capi_get_integer and tested to be the same as thoughts that were set.

Definition at line 510 of file fmi1_capi_me_test.c.

9.43.2.23 int test_set_get_boolean ()

Tests fmi1_capi_set_boolean and fmi1_capi_get_boolean Some values are set with fmi1_capi_set_boolean. The same values are retrieved with fmi1_capi_get_boolean and tested to be the same as thoughts that were set.

Definition at line 555 of file fmi1_capi_me_test.c.

9.43.2.24 int test_set_get_real ()

Tests fmi1_capi_set_real and fmi1_capi_get_real Some values are set with fmi1_capi_set_real. The same values are retrieved with fmi1_capi_get_real and tested to be the same as thoughts that were set.

Definition at line 600 of file fmi1_capi_me_test.c.

9.43.2.25 int test_terminate ()

Tests fmi1_capi_terminate.

Definition at line 646 of file fmi1_capi_me_test.c.

9.43.2.26 int test_free_model_instance ()

Tests fmi1_capi_free_model_instance.

Definition at line 664 of file fmi1_capi_me_test.c.

9.43.2.27 int test_free_dll ()

Tests fmi1_capi_free_dll.

Definition at line 675 of file fmi1_capi_me_test.c.

9.43.2.28 `int test_destroy_dllfmu ( )`

Tests `fmi1_capi_destroy_dllfmu`.

Definition at line 686 of file `fmi1_capi_me_test.c`.

9.43.2.29 `int main ( int argc, char * argv[] )`

Tests the C-API for FMI 1.0 Model Exchange. The tests are performed using a test-dll. The functions are called and the values are set or returned are validated either in the test function(output functions) or inside the dll(input functions). If any error occurs, the program exits.

Definition at line 698 of file `fmi1_capi_me_test.c`.

9.43.3 Variable Documentation

9.43.3.1 `fmi1_capi_t* fmu`

Definition at line 39 of file `fmi1_capi_me_test.c`.

9.43.3.2 `jm_callbacks* callbacks`

Definition at line 69 of file `fmi1_capi_me_test.c`.

9.44 Test/FMI1/fmi1_import_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h> #include <config_test.h> #include <fmilib.h> #include <JM/jm_portability.h>
```

Defines

- `#define BUFFER 1000`

Functions

- static void `fmi1logger` (`fmi1_component_t c`, `fmi1_string_t instanceName`, `fmi1_status_t status`, `fmi1_string_t category`, `fmi1_string_t message`,...)
- int `fmi1_test` (`fmi_import_context_t *context`, const `char *dirPath`)

9.44.1 Define Documentation

9.44.1.1 #define BUFFER 1000

Definition at line 24 of file fmi1_import_test.c.

9.44.2 Function Documentation

9.44.2.1 static void fmi1logger (fmi1_component_t c, fmi1_string_t instanceName, fmi1_status_t status, fmi1_string_t category, fmi1_string_t message, ...) [static]

Definition at line 27 of file fmi1_import_test.c.

9.44.2.2 int fmi1_test (fmi_import_context_t * context, const char * dirPath)

Definition at line 38 of file fmi1_import_test.c.

9.45 Test/FMI1/fmi1_logger_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h>
#include <assert.h> #include "config_test.h" #include
<fmilib.h> #include <fmu_dummy/fmul_model_defines.h>
#include <JM/jm_portability.h>
```

Defines

- #define [MESSAGE_SIZE_TO_EXPAND_AND_PRINT](#) 3000 /* Using fixed size since the log message is printed to a file and compared */

Functions

- void [logger](#) ([jm_callbacks](#) *c, [jm_string](#) module, [jm_log_level_enu_t](#) log_level, [jm_string](#) message)
- void [do_exit](#) (int code)
- int [test_logger](#) ([fmi1_import_t](#) *fmu)
- int [main](#) (int argc, [char](#) *argv[])

Variables

- FILE * [logFile](#)

9.45.1 Define Documentation

9.45.1.1 `#define MESSAGE_SIZE_TO_EXPAND_AND_PRINT 3000` /* Using fixed size since the log message is printed to a file and compared */

9.45.2 Function Documentation

9.45.2.1 `void logger ( jm_callbacks * c, jm_string module, jm_log_level_enu_t log_level, jm_string message )`

Definition at line 28 of file fmi1_logger_test.c.

9.45.2.2 `void do_exit ( int code )`

Definition at line 45 of file fmi1_logger_test.c.

9.45.2.3 `int test_logger ( fmi1_import_t * fmu )`

Definition at line 51 of file fmi1_logger_test.c.

9.45.2.4 `int main ( int argc, char * argv[] )`

Definition at line 142 of file fmi1_logger_test.c.

9.45.3 Variable Documentation

9.45.3.1 `FILE* logFile`

Definition at line 26 of file fmi1_logger_test.c.

9.46 Test/FMI1/fmi2_import_xml_test.cc File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h> #include <assert.h> #include <time.h> #include "config_test.h" #include <FMI/fmi_import_context.h> #include <FMI2/fmi2_import.h>
```

Functions

- `void mylogger (jm_callbacks *c, jm_string module, jm_log_level_enu_t log_level, jm_string message)`
- `int annotation_start_handle (void *context, const char *elm, const char **attr)`
- `int annotation_data_handle (void *context, const char *s, int len)`

- int [annotation_end_handle](#) (void *context, const char *elm)
- void [do_exit](#) (int code)
- void [print_int](#) (int i, void *data)
- void [print_dbl](#) (double d, void *data)
- void [printTypeInfo](#) (fmi2_import_variable_typedef_t *vt)
- void [testVariableSearch](#) (fmi2_import_t *fmu, fmi2_import_variable_t *v)
- void [printVariableInfo](#) (fmi2_import_t *fmu, fmi2_import_variable_t *v)
- void [printCapabilitiesInfo](#) (fmi2_import_t *fmu)
- void [printDependenciesInfo](#) (fmi2_import_t *fmu, fmi2_import_variable_list_t *rows, fmi2_import_variable_list_t *cols, size_t *start, size_t *dep, char *factor)
- int [main](#) (int argc, char *argv[])

Variables

- jm_xml_callbacks_t [annotation_callbacks](#)
XML callbacks are used to process parts of XML that are not handled by the library.

9.46.1 Function Documentation

9.46.1.1 void [mylogger](#) (jm_callbacks * c, jm_string module, jm_log_level_enu_t log_level, jm_string message)

Definition at line 29 of file fmi2_import_xml_test.cc.

9.46.1.2 int [annotation_start_handle](#) (void * context, const char * elm, const char ** attr)

Definition at line 34 of file fmi2_import_xml_test.cc.

9.46.1.3 int [annotation_data_handle](#) (void * context, const char * s, int len)

Definition at line 45 of file fmi2_import_xml_test.cc.

9.46.1.4 int [annotation_end_handle](#) (void * context, const char * elm)

Definition at line 52 of file fmi2_import_xml_test.cc.

9.46.1.5 void [do_exit](#) (int code)

Definition at line 64 of file fmi2_import_xml_test.cc.

9.46.1.6 void [print_int](#) (int i, void * data)

Definition at line 71 of file fmi2_import_xml_test.cc.

9.46.1.7 void **print_dbl** (double *d*, void * *data*)

Definition at line 75 of file fmi2_import_xml_test.cc.

9.46.1.8 void **printTypeInfo** (fmi2_import_variable_typedef_t * *vt*)

Definition at line 79 of file fmi2_import_xml_test.cc.

9.46.1.9 void **testVariableSearch** (fmi2_import_t * *fmu*, fmi2_import_variable_t * *v*)

Definition at line 168 of file fmi2_import_xml_test.cc.

9.46.1.10 void **printVariableInfo** (fmi2_import_t * *fmu*, fmi2_import_variable_t * *v*)

Definition at line 198 of file fmi2_import_xml_test.cc.

9.46.1.11 void **printCapabilitiesInfo** (fmi2_import_t * *fmu*)

Definition at line 269 of file fmi2_import_xml_test.cc.

9.46.1.12 void **printDependenciesInfo** (fmi2_import_t * *fmu*,
fmi2_import_variable_list_t * *rows*, fmi2_import_variable_list_t * *cols*,
size_t * *start*, size_t * *dep*, char * *factor*)

Definition at line 279 of file fmi2_import_xml_test.cc.

9.46.1.13 int **main** (int *argc*, char * *argv*[])

Definition at line 309 of file fmi2_import_xml_test.cc.

9.46.2 Variable Documentation

9.46.2.1 jm_xml_callbacks_t **annotation_callbacks**

Initial value:

```
{
    annotation_start_handle,
    annotation_data_handle,
    annotation_end_handle, NULL}
```

XML callbacks are used to process parts of XML that are not handled by the library.

Definition at line 58 of file fmi2_import_xml_test.cc.

9.47 Test/FMI2/fmi2_import_xml_test.cc File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h>
#include <assert.h> #include <time.h> #include "config-
_test.h" #include <FMI/fmi_import_context.h> #include <F-
MI2/fmi2_import.h>
```

Functions

- void `mylogger` (`jm_callbacks` *c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)
- int `annotation_start_handle` (void *context, const char *parentName, void *parent, const char *elm, const char **attr)
- int `annotation_data_handle` (void *context, const char *s, int len)
- int `annotation_end_handle` (void *context, const char *elm)
- void `do_exit` (int code)
- void `print_int` (int i, void *data)
- void `print_dbl` (double d, void *data)
- void `printTypeInfo` (`fmi2_import_variable_typedef_t` *vt)
- void `testVariableSearch` (`fmi2_import_t` *fmu, `fmi2_import_variable_t` *v)
- void `printVariableInfo` (`fmi2_import_t` *fmu, `fmi2_import_variable_t` *v)
- void `printCapabilitiesInfo` (`fmi2_import_t` *fmu)
- void `printDependenciesInfo` (`fmi2_import_t` *fmu, `fmi2_import_variable_list_t` *rows, `fmi2_import_variable_list_t` *cols, `size_t` *start, `size_t` *dep, char *factor)
- int `main` (int argc, char *argv[])

Variables

- `fmi2_xml_callbacks_t` `annotation_callbacks`
XML callbacks are used to process parts of XML that are not handled by the library.

9.47.1 Function Documentation

9.47.1.1 void `mylogger` (`jm_callbacks` * c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)

Definition at line 29 of file `fmi2_import_xml_test.cc`.

9.47.1.2 int `annotation_start_handle` (void * context, const char * parentName, void * parent, const char * elm, const char ** attr)

Definition at line 34 of file `fmi2_import_xml_test.cc`.

9.47.1.3 `int annotation_data_handle ( void * context, const char * s, int len )`

Definition at line 46 of file fmi2_import_xml_test.cc.

9.47.1.4 `int annotation_end_handle ( void * context, const char * elm )`

Definition at line 53 of file fmi2_import_xml_test.cc.

9.47.1.5 `void do_exit ( int code )`

Definition at line 65 of file fmi2_import_xml_test.cc.

9.47.1.6 `void print_int ( int i, void * data )`

Definition at line 72 of file fmi2_import_xml_test.cc.

9.47.1.7 `void print_dbl ( double d, void * data )`

Definition at line 76 of file fmi2_import_xml_test.cc.

9.47.1.8 `void printTypeInfo ( fmi2_import_variable_typedef_t * vt )`

Definition at line 80 of file fmi2_import_xml_test.cc.

9.47.1.9 `void testVariableSearch ( fmi2_import_t * fmu, fmi2_import_variable_t * v )`

Definition at line 169 of file fmi2_import_xml_test.cc.

9.47.1.10 `void printVariableInfo ( fmi2_import_t * fmu, fmi2_import_variable_t * v )`

Definition at line 199 of file fmi2_import_xml_test.cc.

9.47.1.11 `void printCapabilitiesInfo ( fmi2_import_t * fmu )`

Definition at line 270 of file fmi2_import_xml_test.cc.

9.47.1.12 `void printDependenciesInfo (fmi2_import_t * fmu,
fmi2_import_variable_list_t * rows, fmi2_import_variable_list_t * cols,
size_t * start, size_t * dep, char * factor)`

Definition at line 280 of file fmi2_import_xml_test.cc.

9.47.1.13 `int main ( int argc, char * argv[] )`

Definition at line 310 of file `fmi2_import_xml_test.cc`.

9.47.2 Variable Documentation

9.47.2.1 `fmi2_xml_callbacks_t annotation_callbacks`

Initial value:

```
{
    annotation_start_handle,
    annotation_data_handle,
    annotation_end_handle, NULL}
```

XML callbacks are used to process parts of XML that are not handled by the library.

Definition at line 59 of file `fmi2_import_xml_test.cc`.

9.48 Test/FMI1/fmi_import_cs_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.-
h> #include "config_test.h" #include <fmilib.h>
```

Defines

- `#define BUFFER 1000`

Functions

- void `importlogger` (`jm_callbacks` *c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)
- void `fmilogger` (`fmi1_component_t` c, `fmi1_string_t` instanceName, `fmi1_status_t` status, `fmi1_string_t` category, `fmi1_string_t` message,...)
- void `do_exit` (int code)
- int `test_simulate_cs` (`fmi1_import_t` *fmu)
- int `main` (int argc, `char` *argv[])

9.48.1 Define Documentation

9.48.1.1 `#define BUFFER 1000`

Definition at line 25 of file `fmi_import_cs_test.c`.

9.48.2 Function Documentation

9.48.2.1 `void importlogger (jm_callbacks * c, jm_string module,
jm_log_level_enu_t log_level, jm_string message)`

Definition at line 27 of file `fmi_import_cs_test.c`.

9.48.2.2 `void fmilogger (fmi1_component_t c, fmi1_string_t instanceName, fmi1_status_t
status, fmi1_string_t category, fmi1_string_t message, ...)`

Definition at line 34 of file `fmi_import_cs_test.c`.

9.48.2.3 `void do_exit ( int code )`

Definition at line 44 of file `fmi_import_cs_test.c`.

9.48.2.4 `int test_simulate_cs ( fmi1_import_t * fmu )`

Definition at line 51 of file `fmi_import_cs_test.c`.

9.48.2.5 `int main ( int argc, char * argv[] )`

Definition at line 152 of file `fmi_import_cs_test.c`.

9.49 Test/FMI1/fmi_import_me_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.-  
h> #include "config_test.h" #include <fmilib.h>
```

Data Structures

- struct [fmu_t](#)

Defines

- #define [BUFFER](#) 1000
- #define [NUMBER_OF_TESTS](#) 150

Functions

- void [do_exit](#) (int code)
- int [test_simulate_me](#) ([fmi1_import_t](#) *fmu)

- `fmul_t load` (int argc, char *argv[])
- void `destroy` (fmul_t *fmus)
- int `main` (int argc, char *argv[])

9.49.1 Define Documentation

9.49.1.1 #define BUFFER 1000

Definition at line 23 of file fmi_import_me_test.c.

9.49.1.2 #define NUMBER_OF_TESTS 150

Definition at line 260 of file fmi_import_me_test.c.

9.49.2 Function Documentation

9.49.2.1 void do_exit (int code)

Definition at line 25 of file fmi_import_me_test.c.

9.49.2.2 int test_simulate_me (fmi1_import_t * fmu)

Definition at line 32 of file fmi_import_me_test.c.

9.49.2.3 fmul_t load (int argc, char * argv[])

Definition at line 174 of file fmi_import_me_test.c.

9.49.2.4 void destroy (fmul_t * fmus)

Definition at line 251 of file fmi_import_me_test.c.

9.49.2.5 int main (int argc, char * argv[])

Definition at line 261 of file fmi_import_me_test.c.

9.50 Test/FMI1/fmi_import_xml_test.cc File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h>
#include <assert.h> #include <time.h> #include "config-
_test.h" #include <FMI/fmi_import_context.h> #include <F-
MI1/fmi1_import.h>
```

Functions

- void `mylogger` (`jm_callbacks` *c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)
- void `do_exit` (int code)
- void `print_int` (int i, void *data)
- void `print_dbl` (double d, void *data)
- void `printTypeInfo` (`fmi1_import_variable_typedef_t` *vt)
- void `testVariableSearch` (`fmi1_import_t` *fmu, `fmi1_import_variable_t` *v)
- void `printVariableInfo` (`fmi1_import_t` *fmu, `fmi1_import_variable_t` *v)
- void `printCapabilitiesInfo` (`fmi1_import_capabilities_t` *capabilities)
- int `main` (int argc, char *argv[])

9.50.1 Function Documentation

9.50.1.1 void `mylogger` (`jm_callbacks` * c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)

Definition at line 29 of file `fmi_import_xml_test.cc`.

9.50.1.2 void `do_exit` (int code)

Definition at line 34 of file `fmi_import_xml_test.cc`.

9.50.1.3 void `print_int` (int i, void * data)

Definition at line 41 of file `fmi_import_xml_test.cc`.

9.50.1.4 void `print_dbl` (double d, void * data)

Definition at line 45 of file `fmi_import_xml_test.cc`.

9.50.1.5 void `printTypeInfo` (`fmi1_import_variable_typedef_t` * vt)

Definition at line 49 of file `fmi_import_xml_test.cc`.

9.50.1.6 void `testVariableSearch` (`fmi1_import_t` * fmu, `fmi1_import_variable_t` * v)

Definition at line 128 of file `fmi_import_xml_test.cc`.

9.50.1.7 void `printVariableInfo` (`fmi1_import_t` * fmu, `fmi1_import_variable_t` * v)

Definition at line 158 of file `fmi_import_xml_test.cc`.

9.50.1.8 void printCapabilitiesInfo (fmi1_import_capabilities_t * capabilities)

Definition at line 243 of file fmi_import_xml_test.cc.

9.50.1.9 int main (int argc, char * argv[])

Definition at line 256 of file fmi_import_xml_test.cc.

9.51 Test/FMI1/fmi_total_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include "fmi_dll-
_1_0_cs.h" #include "fmi_dll_1_0_me.h" #include "fmi_dll-
_common.h" #include "jm_types.h" #include "fmi_zip_unzip.-
h" #include <FMI1/fmil_xml_model_description.h> #include
"fmi_import_util.h"
```

Defines

- #define `PRINT_MY_DEBUG` printf("Line: %d \t File: %s \n", __LINE__, __FILE__)

Functions

- void `mylogger` (fmiComponent c, fmiString instanceName, fmiStatus status, fmiString category, fmiString message,...)
- void `do_pause` ()
- int `main` (int argc, char *argv[])

9.51.1 Define Documentation

9.51.1.1 #define `PRINT_MY_DEBUG` printf("Line: %d \t File: %s \n", __LINE__, __FILE__)

Definition at line 27 of file fmi_total_test.c.

9.51.2 Function Documentation

9.51.2.1 void `mylogger` (fmiComponent c, fmiString instanceName, fmiStatus status, fmiString category, fmiString message, ...)

Definition at line 29 of file fmi_total_test.c.

9.51.2.2 void do_pause ()

Definition at line 40 of file fmi_total_test.c.

9.51.2.3 int main (int argc, char * argv[])

Definition at line 48 of file fmi_total_test.c.

9.52 Test/FMI1/fmi_zip_unzip_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h> #include <string.h> #include <errno.h> #include <JM/jm_types.h> #include <JM/jm_callbacks.h> #include <FMI/fmi_zip_unzip.h> #include "config_test.h"
```

Functions

- void [do_exit](#) (int code)
- void [importlogger](#) (jm_callbacks *c, jm_string module, jm_log_level_enu_t log_level, jm_string message)
- int [main](#) (int argc, char *argv[])

Unzip test. Tests the fmi_zip_unzip function by uncompressing some file.

9.52.1 Function Documentation

9.52.1.1 void do_exit (int code)

Definition at line 28 of file fmi_zip_unzip_test.c.

9.52.1.2 void importlogger (jm_callbacks * c, jm_string module, jm_log_level_enu_t log_level, jm_string message)

Definition at line 36 of file fmi_zip_unzip_test.c.

9.52.1.3 int main (int argc, char * argv[])

Unzip test. Tests the fmi_zip_unzip function by uncompressing some file.

Definition at line 45 of file fmi_zip_unzip_test.c.

9.53 Test/fmi_zip_unzip_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h>
#include <string.h> #include <errno.h> #include <JM/jm_types.h>
#include <JM/jm_callbacks.h> #include <FMI/fmi_zip_unzip.h>
#include "config_test.h"
```

Functions

- void [do_exit](#) (int code)
- void [importlogger](#) (jm_callbacks *c, jm_string module, jm_log_level_enu_t log_level, jm_string message)
- int [main](#) (int argc, char *argv[])

Unzip test. Tests the fmi_zip_unzip function by uncompressing some file.

9.53.1 Function Documentation

9.53.1.1 void do_exit (int code)

Definition at line 28 of file fmi_zip_unzip_test.c.

9.53.1.2 void importlogger (jm_callbacks * c, jm_string module, jm_log_level_enu_t log_level, jm_string message)

Definition at line 36 of file fmi_zip_unzip_test.c.

9.53.1.3 int main (int argc, char * argv[])

Unzip test. Tests the fmi_zip_unzip function by uncompressing some file.

Definition at line 45 of file fmi_zip_unzip_test.c.

9.54 Test/FMI1/fmi_zip_zip_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <JM/jm_types.h>
#include <JM/jm_callbacks.h> #include <FMI/fmi_zip_zip.h>
#include "config_test.h"
```

Defines

- #define [PRINT_MY_DEBUG](#) printf("Line: %d \t File: %s \n", __LINE__, __FILE__)

Functions

- void `do_exit` (int code)
- void `importlogger` (`jm_callbacks` *c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)
- int `main` (int argc, `char` *argv[])

Zip test. Tests the fmi_zip_zip function by compressing some file.

9.54.1 Define Documentation

9.54.1.1 `#define PRINT_MY_DEBUG printf("Line: %d \t File: %s \n", __LINE__, __FILE__)`

Definition at line 24 of file `fmi_zip_zip_test.c`.

9.54.2 Function Documentation

9.54.2.1 void `do_exit` (int code)

Definition at line 28 of file `fmi_zip_zip_test.c`.

9.54.2.2 void `importlogger` (`jm_callbacks` * c, `jm_string` module, `jm_log_level_enu_t` log_level, `jm_string` message)

Definition at line 36 of file `fmi_zip_zip_test.c`.

9.54.2.3 int `main` (int argc, `char` * argv[])

Zip test. Tests the fmi_zip_zip function by compressing some file.

Definition at line 45 of file `fmi_zip_zip_test.c`.

9.55 Test/fmi_zip_zip_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <JM/jm_
types.h> #include <JM/jm_callbacks.h> #include <FMI/fmi_
zip_zip.h> #include "config_test.h"
```

Defines

- `#define PRINT_MY_DEBUG printf("Line: %d \t File: %s \n", __LINE__, __FILE__)`

Functions

- void `do_exit` (int code)
- void `importlogger` (jm_callbacks *c, jm_string module, jm_log_level_enu_t log_level, jm_string message)
- int `main` (int argc, char *argv[])

Zip test. Tests the fmi_zip_zip function by compressing some file.

9.55.1 Define Documentation

9.55.1.1 `#define PRINT_MY_DEBUG` `printf("Line: %d \t File: %s \n", __LINE__, __FILE__)`

Definition at line 24 of file fmi_zip_zip_test.c.

9.55.2 Function Documentation

9.55.2.1 void `do_exit` (int code)

Definition at line 28 of file fmi_zip_zip_test.c.

9.55.2.2 void `importlogger` (jm_callbacks * c, jm_string module, jm_log_level_enu_t log_level, jm_string message)

Definition at line 36 of file fmi_zip_zip_test.c.

9.55.2.3 int `main` (int argc, char * argv[])

Zip test. Tests the fmi_zip_zip function by compressing some file.

Definition at line 45 of file fmi_zip_zip_test.c.

9.56 Test/FMI1/fmu_dummy/fmu1_model.c File Reference

```
#include <stdio.h>  #include <string.h>  #include <fmu_
dummy/fmu1_model.h>
```

Defines

- `#define FMI_TEST_LOGGER_TEST_RESULT_FILE` "C:\\P510-JModelica\\FMIToolbox\\trunk\\external\\FMIL\\build\\testfolder\\"
- `#define FMI_TEST_LOGGER_TEST_SOURCE_FILE` "C:\\P510-JModelica\\FMIToolbox\\trunk\\external\\FMIL\\build\\testfolder\\"

Functions

- static int `calc_initialize` (`component_ptr_t` comp)
- static int `calc_get_derivatives` (`component_ptr_t` comp)
- static int `calc_get_event_indicators` (`component_ptr_t` comp)
- static int `calc_event_update` (`component_ptr_t` comp)
- const `char` * `fmi_get_version` ()
- `fmiStatus` `fmi_set_debug_logging` (`fmiComponent` c, `fmiBoolean` loggingOn)
- `fmiStatus` `fmi_get_real` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, `fmiReal` value[])
- `fmiStatus` `fmi_get_integer` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, `fmiInteger` value[])
- `fmiStatus` `fmi_get_boolean` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, `fmiBoolean` value[])
- `fmiStatus` `fmi_get_string` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, `fmiString` value[])
- `fmiStatus` `fmi_set_real` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, const `fmiReal` value[])
- `fmiStatus` `fmi_set_integer` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, const `fmiInteger` value[])
- `fmiStatus` `fmi_set_boolean` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, const `fmiBoolean` value[])
- `fmiStatus` `fmi_set_string` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, const `fmiString` value[])
- const `char` * `fmi_get_model_types_platform` ()
- `fmiComponent` `fmi_instantiate_model` (`fmiString` instanceName, `fmiString` GUID, `fmiCallbackFunctions` functions, `fmiBoolean` loggingOn)
- void `fmi_free_model_instance` (`fmiComponent` c)
- `fmiStatus` `fmi_set_time` (`fmiComponent` c, `fmiReal` fmitime)
- `fmiStatus` `fmi_set_continuous_states` (`fmiComponent` c, const `fmiReal` x[], `size_t` nx)
- `fmiStatus` `fmi_completed_integrator_step` (`fmiComponent` c, `fmiBoolean` *call-EventUpdate)
- `fmiStatus` `fmi_initialize` (`fmiComponent` c, `fmiBoolean` toleranceControlled, `fmiReal` relativeTolerance, `fmiEventInfo` *eventInfo)
- `fmiStatus` `fmi_get_derivatives` (`fmiComponent` c, `fmiReal` derivatives[], `size_t` nx)
- `fmiStatus` `fmi_get_event_indicators` (`fmiComponent` c, `fmiReal` eventIndicators[], `size_t` ni)
- `fmiStatus` `fmi_event_update` (`fmiComponent` c, `fmiBoolean` intermediateResults, `fmiEventInfo` *eventInfo)
- `fmiStatus` `fmi_get_continuous_states` (`fmiComponent` c, `fmiReal` states[], `size_t` nx)
- `fmiStatus` `fmi_get_nominal_continuousstates` (`fmiComponent` c, `fmiReal` x_nominal[], `size_t` nx)
- `fmiStatus` `fmi_get_state_value_references` (`fmiComponent` c, `fmiValueReference` vrx[], `size_t` nx)
- `fmiStatus` `fmi_terminate` (`fmiComponent` c)
- const `char` * `fmi_get_types_platform` ()

- `fmiComponent fmi_instantiate_slave` (`fmiString` instanceName, `fmiString` fmuGUID, `fmiString` fmuLocation, `fmiString` mimeType, `fmiReal` timeout, `fmiBoolean` visible, `fmiBoolean` interactive, `fmiCallbackFunctions` functions, `fmiBoolean` logging-On)
- `fmiStatus fmi_initialize_slave` (`fmiComponent` c, `fmiReal` tStart, `fmiBoolean` Stop-TimeDefined, `fmiReal` tStop)
- `fmiStatus fmi_terminate_slave` (`fmiComponent` c)
- `fmiStatus fmi_reset_slave` (`fmiComponent` c)
- `void fmi_free_slave_instance` (`fmiComponent` c)
- `fmiStatus fmi_set_real_input_derivatives` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, const `fmiInteger` order[], const `fmiReal` value[])
- `fmiStatus fmi_get_real_output_derivatives` (`fmiComponent` c, const `fmiValueReference` vr[], `size_t` nvr, const `fmiInteger` order[], `fmiReal` value[])
- `fmiStatus fmi_cancel_step` (`fmiComponent` c)
- `fmiStatus fmi_do_step` (`fmiComponent` c, `fmiReal` currentCommunicationPoint, `fmiReal` communicationStepSize, `fmiBoolean` newStep)
- `fmiStatus fmi_get_status` (`fmiComponent` c, const `fmiStatusKind` s, `fmiStatus` *value)
- `fmiStatus fmi_get_real_status` (`fmiComponent` c, const `fmiStatusKind` s, `fmiReal` *value)
- `fmiStatus fmi_get_integer_status` (`fmiComponent` c, const `fmiStatusKind` s, `fmiInteger` *value)
- `fmiStatus fmi_get_boolean_status` (`fmiComponent` c, const `fmiStatusKind` s, `fmiBoolean` *value)
- `fmiStatus fmi_get_string_status` (`fmiComponent` c, const `fmiStatusKind` s, `fmiString` *value)

9.56.1 Define Documentation

9.56.1.1 `#define FMI_TEST_LOGGER_TEST_RESULT_FILE "C:\\P510-JModelica\\FMIToolbox\\trunk\\external\\FMIL\\build\\testfolder\\"`

Definition at line 247 of file `fmu1_model.c`.

9.56.1.2 `#define FMI_TEST_LOGGER_TEST_SOURCE_FILE "C:\\P510-JModelica\\FMIToolbox\\trunk\\external\\FMIL\\build\\testfolder\\"`

Definition at line 248 of file `fmu1_model.c`.

9.56.2 Function Documentation

9.56.2.1 `static int calc_initialize ( component_ptr_t comp )` `[static]`

Definition at line 23 of file `fmu1_model.c`.

9.56.2.2 `static int calc_get_derivatives ( component_ptr_t comp ) [static]`

Definition at line 43 of file `fmu1_model.c`.

9.56.2.3 `static int calc_get_event_indicators (component_ptr_t comp)
[static]`

Definition at line 50 of file `fmu1_model.c`.

9.56.2.4 `static int calc_event_update ( component_ptr_t comp ) [static]`

Definition at line 57 of file `fmu1_model.c`.

9.56.2.5 `const char* fmi_get_version ( )`

Definition at line 77 of file `fmu1_model.c`.

9.56.2.6 `fmiStatus fmi_set_debug_logging (fmiComponent c, fmiBoolean loggingOn
)`

Definition at line 82 of file `fmu1_model.c`.

9.56.2.7 `fmiStatus fmi_get_real (fmiComponent c, const fmiValueReference vr[],
size_t nvr, fmiReal value[])`

Definition at line 93 of file `fmu1_model.c`.

9.56.2.8 `fmiStatus fmi_get_integer (fmiComponent c, const fmiValueReference vr[],
size_t nvr, fmiInteger value[])`

Definition at line 113 of file `fmu1_model.c`.

9.56.2.9 `fmiStatus fmi_get_boolean (fmiComponent c, const fmiValueReference
vr[], size_t nvr, fmiBoolean value[])`

Definition at line 127 of file `fmu1_model.c`.

9.56.2.10 `fmiStatus fmi_get_string (fmiComponent c, const fmiValueReference vr[],
size_t nvr, fmiString value[])`

Definition at line 141 of file `fmu1_model.c`.

9.56.2.11 `fmiStatus fmi_set_real (fmiComponent c, const fmiValueReference vr[],
size_t nvr, const fmiReal value[])`

Definition at line 155 of file `fmu1_model.c`.

9.56.2.12 `fmiStatus fmi_set_integer (fmiComponent c, const fmiValueReference vr[],
size_t nvr, const fmiInteger value[])`

Definition at line 175 of file `fmu1_model.c`.

9.56.2.13 `fmiStatus fmi_set_boolean (fmiComponent c, const fmiValueReference
vr[], size_t nvr, const fmiBoolean value[])`

Definition at line 189 of file `fmu1_model.c`.

9.56.2.14 `fmiStatus fmi_set_string (fmiComponent c, const fmiValueReference vr[],
size_t nvr, const fmiString value[])`

Definition at line 203 of file `fmu1_model.c`.

9.56.2.15 `const char* fmi_get_model_types_platform ( )`

Definition at line 242 of file `fmu1_model.c`.

9.56.2.16 `fmiComponent fmi_instantiate_model (fmiString instanceName,
fmiString GUID, fmiCallbackFunctions functions, fmiBoolean loggingOn)`

Definition at line 254 of file `fmu1_model.c`.

9.56.2.17 `void fmi_free_model_instance ( fmiComponent c )`

Definition at line 302 of file `fmu1_model.c`.

9.56.2.18 `fmiStatus fmi_set_time ( fmiComponent c, fmiReal fmitime )`

Definition at line 313 of file `fmu1_model.c`.

9.56.2.19 `fmiStatus fmi_set_continuous_states (fmiComponent c, const fmiReal x[],
size_t nx)`

Definition at line 324 of file `fmu1_model.c`.

9.56.2.20 **fmiStatus fmi_completed_integrator_step** (**fmiComponent** *c*, **fmiBoolean** * *callEventUpdate*)

Definition at line 338 of file `fmu1_model.c`.

9.56.2.21 **fmiStatus fmi_initialize** (**fmiComponent** *c*, **fmiBoolean** *toleranceControlled*, **fmiReal** *relativeTolerance*, **fmiEventInfo** * *eventInfo*)

Definition at line 349 of file `fmu1_model.c`.

9.56.2.22 **fmiStatus fmi_get_derivatives** (**fmiComponent** *c*, **fmiReal** *derivatives*[], **size_t** *nx*)

Definition at line 374 of file `fmu1_model.c`.

9.56.2.23 **fmiStatus fmi_get_event_indicators** (**fmiComponent** *c*, **fmiReal** *eventIndicators*[], **size_t** *ni*)

Definition at line 391 of file `fmu1_model.c`.

9.56.2.24 **fmiStatus fmi_event_update** (**fmiComponent** *c*, **fmiBoolean** *intermediateResults*, **fmiEventInfo** * *eventInfo*)

Definition at line 408 of file `fmu1_model.c`.

9.56.2.25 **fmiStatus fmi_get_continuous_states** (**fmiComponent** *c*, **fmiReal** *states*[], **size_t** *nx*)

Definition at line 421 of file `fmu1_model.c`.

9.56.2.26 **fmiStatus fmi_get_nominal_continuousstates** (**fmiComponent** *c*, **fmiReal** *x_nominal*[], **size_t** *nx*)

Definition at line 436 of file `fmu1_model.c`.

9.56.2.27 **fmiStatus fmi_get_state_value_references** (**fmiComponent** *c*, **fmiValueReference** *vr*[], **size_t** *nx*)

Definition at line 450 of file `fmu1_model.c`.

9.56.2.28 **fmiStatus fmi_terminate** (**fmiComponent** *c*)

Definition at line 464 of file `fmu1_model.c`.

9.56.2.29 `const char* fmi_get_types_platform ( )`

Definition at line 475 of file `fmu1_model.c`.

9.56.2.30 `fmiComponent fmi_instantiate_slave ( fmiString instanceName, fmiString fmuGUID, fmiString fmuLocation, fmiString mimeType, fmiReal timeout, fmiBoolean visible, fmiBoolean interactive, fmiCallbackFunctions functions, fmiBoolean loggingOn )`

Definition at line 480 of file `fmu1_model.c`.

9.56.2.31 `fmiStatus fmi_initialize_slave ( fmiComponent c, fmiReal tStart, fmiBoolean StopTimeDefined, fmiReal tStop )`

Definition at line 499 of file `fmu1_model.c`.

9.56.2.32 `fmiStatus fmi_terminate_slave ( fmiComponent c )`

Definition at line 517 of file `fmu1_model.c`.

9.56.2.33 `fmiStatus fmi_reset_slave ( fmiComponent c )`

Definition at line 522 of file `fmu1_model.c`.

9.56.2.34 `void fmi_free_slave_instance ( fmiComponent c )`

Definition at line 527 of file `fmu1_model.c`.

9.56.2.35 `fmiStatus fmi_set_real_input_derivatives ( fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiInteger order[], const fmiReal value[] )`

Definition at line 532 of file `fmu1_model.c`.

9.56.2.36 `fmiStatus fmi_get_real_output_derivatives ( fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiInteger order[], fmiReal value[] )`

Definition at line 548 of file `fmu1_model.c`.

9.56.2.37 `fmiStatus fmi_cancel_step ( fmiComponent c )`

Definition at line 560 of file `fmu1_model.c`.

9.56.2.38 `fmiStatus fmi_do_step (fmiComponent c, fmiReal
currentCommunicationPoint, fmiReal communicationStepSize, fmiBoolean
newStep)`

Definition at line 565 of file `fmu1_model.c`.

9.56.2.39 `fmiStatus fmi_get_status (fmiComponent c, const fmiStatusKind s,
fmiStatus * value)`

Definition at line 662 of file `fmu1_model.c`.

9.56.2.40 `fmiStatus fmi_get_real_status (fmiComponent c, const fmiStatusKind s,
fmiReal * value)`

Definition at line 674 of file `fmu1_model.c`.

9.56.2.41 `fmiStatus fmi_get_integer_status (fmiComponent c, const fmiStatusKind
s, fmiInteger * value)`

Definition at line 686 of file `fmu1_model.c`.

9.56.2.42 `fmiStatus fmi_get_boolean_status (fmiComponent c, const fmiStatusKind
s, fmiBoolean * value)`

Definition at line 694 of file `fmu1_model.c`.

9.56.2.43 `fmiStatus fmi_get_string_status (fmiComponent c, const fmiStatusKind s,
fmiString * value)`

Definition at line 702 of file `fmu1_model.c`.

9.57 Test/FMI1/fmu_dummy/fmu1_model.h File Reference

```
#include <fmu_dummy/fmu1_model_defines.h>
```

Data Structures

- struct [component_t](#)

Typedefs

- typedef [component_t](#) * [component_ptr_t](#)

Functions

- const char * fmi_get_version ()
- fmiStatus fmi_set_debug_logging (fmiComponent c, fmiBoolean loggingOn)
- fmiStatus fmi_get_real (fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiReal value[])
- fmiStatus fmi_get_integer (fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiInteger value[])
- fmiStatus fmi_get_boolean (fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiBoolean value[])
- fmiStatus fmi_get_string (fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiString value[])
- fmiStatus fmi_set_real (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiReal value[])
- fmiStatus fmi_set_integer (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiInteger value[])
- fmiStatus fmi_set_boolean (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiBoolean value[])
- fmiStatus fmi_set_string (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiString value[])
- const char * fmi_get_model_types_platform ()
- fmiComponent fmi_instantiate_model (fmiString instanceName, fmiString GUID, fmiCallbackFunctions functions, fmiBoolean loggingOn)
- void fmi_free_model_instance (fmiComponent c)
- fmiStatus fmi_set_time (fmiComponent c, fmiReal fmtime)
- fmiStatus fmi_set_continuous_states (fmiComponent c, const fmiReal x[], size_t nx)
- fmiStatus fmi_completed_integrator_step (fmiComponent c, fmiBoolean *call-EventUpdate)
- fmiStatus fmi_initialize (fmiComponent c, fmiBoolean toleranceControlled, fmiReal relativeTolerance, fmiEventInfo *eventInfo)
- fmiStatus fmi_get_derivatives (fmiComponent c, fmiReal derivatives[], size_t nx)
- fmiStatus fmi_get_event_indicators (fmiComponent c, fmiReal eventIndicators[], size_t ni)
- fmiStatus fmi_event_update (fmiComponent c, fmiBoolean intermediateResults, fmiEventInfo *eventInfo)
- fmiStatus fmi_get_continuous_states (fmiComponent c, fmiReal states[], size_t nx)
- fmiStatus fmi_get_nominal_continuousstates (fmiComponent c, fmiReal x_nominal[], size_t nx)
- fmiStatus fmi_get_state_value_references (fmiComponent c, fmiValueReference vr[], size_t nx)
- fmiStatus fmi_terminate (fmiComponent c)

9.57.1 Typedef Documentation

9.57.1.1 typedef component_t* component_ptr_t

Definition at line 75 of file fmu1_model.h.

9.57.2 Function Documentation

9.57.2.1 `const char* fmi_get_version ( )`

Definition at line 77 of file `fmu1_model.c`.

9.57.2.2 `fmiStatus fmi_set_debug_logging ( fmiComponent c, fmiBoolean loggingOn )`

Definition at line 82 of file `fmu1_model.c`.

9.57.2.3 `fmiStatus fmi_get_real ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiReal value[] )`

Definition at line 93 of file `fmu1_model.c`.

9.57.2.4 `fmiStatus fmi_get_integer ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiInteger value[] )`

Definition at line 113 of file `fmu1_model.c`.

9.57.2.5 `fmiStatus fmi_get_boolean ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiBoolean value[] )`

Definition at line 127 of file `fmu1_model.c`.

9.57.2.6 `fmiStatus fmi_get_string ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiString value[] )`

Definition at line 141 of file `fmu1_model.c`.

9.57.2.7 `fmiStatus fmi_set_real ( fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiReal value[] )`

Definition at line 155 of file `fmu1_model.c`.

9.57.2.8 `fmiStatus fmi_set_integer ( fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiInteger value[] )`

Definition at line 175 of file `fmu1_model.c`.

9.57.2.9 **fmiStatus fmi_set_boolean** (**fmiComponent** *c*, **const fmiValueReference** *vr[]*, **size_t** *nvr*, **const fmiBoolean** *value[]*)

Definition at line 189 of file fmu1_model.c.

9.57.2.10 **fmiStatus fmi_set_string** (**fmiComponent** *c*, **const fmiValueReference** *vr[]*, **size_t** *nvr*, **const fmiString** *value[]*)

Definition at line 203 of file fmu1_model.c.

9.57.2.11 **const char*** **fmi_get_model_types_platform** ()

Definition at line 242 of file fmu1_model.c.

9.57.2.12 **fmiComponent** **fmi_instantiate_model** (**fmiString** *instanceName*, **fmiString** *GUID*, **fmiCallbackFunctions** *functions*, **fmiBoolean** *loggingOn*)

Definition at line 254 of file fmu1_model.c.

9.57.2.13 **void** **fmi_free_model_instance** (**fmiComponent** *c*)

Definition at line 302 of file fmu1_model.c.

9.57.2.14 **fmiStatus fmi_set_time** (**fmiComponent** *c*, **fmiReal** *fmitime*)

Definition at line 313 of file fmu1_model.c.

9.57.2.15 **fmiStatus fmi_set_continuous_states** (**fmiComponent** *c*, **const fmiReal** *x[]*, **size_t** *nx*)

Definition at line 324 of file fmu1_model.c.

9.57.2.16 **fmiStatus fmi_completed_integrator_step** (**fmiComponent** *c*, **fmiBoolean** * *callEventUpdate*)

Definition at line 338 of file fmu1_model.c.

9.57.2.17 **fmiStatus fmi_initialize** (**fmiComponent** *c*, **fmiBoolean** *toleranceControlled*, **fmiReal** *relativeTolerance*, **fmiEventInfo** * *eventInfo*)

Definition at line 349 of file fmu1_model.c.

9.57.2.18 `fmiStatus fmi_get_derivatives ( fmiComponent c, fmiReal derivatives[], size_t nx )`

Definition at line 374 of file `fmu1_model.c`.

9.57.2.19 `fmiStatus fmi_get_event_indicators ( fmiComponent c, fmiReal eventIndicators[], size_t ni )`

Definition at line 391 of file `fmu1_model.c`.

9.57.2.20 `fmiStatus fmi_event_update ( fmiComponent c, fmiBoolean intermediateResults, fmiEventInfo * eventInfo )`

Definition at line 408 of file `fmu1_model.c`.

9.57.2.21 `fmiStatus fmi_get_continuous_states ( fmiComponent c, fmiReal states[], size_t nx )`

Definition at line 421 of file `fmu1_model.c`.

9.57.2.22 `fmiStatus fmi_get_nominal_continuousstates ( fmiComponent c, fmiReal x_nominal[], size_t nx )`

Definition at line 436 of file `fmu1_model.c`.

9.57.2.23 `fmiStatus fmi_get_state_value_references ( fmiComponent c, fmiValueReference vr[], size_t nx )`

Definition at line 450 of file `fmu1_model.c`.

9.57.2.24 `fmiStatus fmi_terminate ( fmiComponent c )`

Definition at line 464 of file `fmu1_model.c`.

9.58 Test/FMI1/fmu_dummy/fmu1_model_cs.c File Reference

```
#include <string.h> #include <FMI1/fmiPlatformTypes.h> ×
#include <FMI1/fmiFunctions.h> #include <fmu_dummy/fmu1_
model.h> #include "config_test.h" #include "fmu1_model.c"
```

Defines

- `#define MODEL_IDENTIFIER FMU_DUMMY_CS_MODEL_IDENTIFIER`

Functions

- DllExport const char * fmiGetVersion ()
- DllExport fmiStatus fmiSetDebugLogging (fmiComponent c, fmiBoolean loggingOn)
- DllExport fmiStatus fmiGetReal (fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiReal value[])
- DllExport fmiStatus fmiGetInteger (fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiInteger value[])
- DllExport fmiStatus fmiGetBoolean (fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiBoolean value[])
- DllExport fmiStatus fmiGetString (fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiString value[])
- DllExport fmiStatus fmiSetReal (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiReal value[])
- DllExport fmiStatus fmiSetInteger (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiInteger value[])
- DllExport fmiStatus fmiSetBoolean (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiBoolean value[])
- DllExport fmiStatus fmiSetString (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiString value[])
- DllExport const char * fmiGetTypesPlatform ()
- DllExport fmiComponent fmiInstantiateSlave (fmiString instanceName, fmiString fmuGUID, fmiString fmuLocation, fmiString mimeType, fmiReal timeout, fmiBoolean visible, fmiBoolean interactive, fmiCallbackFunctions functions, fmiBoolean loggingOn)
- DllExport fmiStatus fmiInitializeSlave (fmiComponent c, fmiReal tStart, fmiBoolean StopTimeDefined, fmiReal tStop)
- DllExport fmiStatus fmiTerminateSlave (fmiComponent c)
- DllExport fmiStatus fmiResetSlave (fmiComponent c)
- DllExport void fmiFreeSlaveInstance (fmiComponent c)
- DllExport fmiStatus fmiSetRealInputDerivatives (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiInteger order[], const fmiReal value[])
- DllExport fmiStatus fmiGetRealOutputDerivatives (fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiInteger order[], fmiReal value[])
- DllExport fmiStatus fmiCancelStep (fmiComponent c)
- DllExport fmiStatus fmiDoStep (fmiComponent c, fmiReal currentCommunicationPoint, fmiReal communicationStepSize, fmiBoolean newStep)
- DllExport fmiStatus fmiGetStatus (fmiComponent c, const fmiStatusKind s, fmiStatus *value)
- DllExport fmiStatus fmiGetRealStatus (fmiComponent c, const fmiStatusKind s, fmiReal *value)
- DllExport fmiStatus fmiGetIntegerStatus (fmiComponent c, const fmiStatusKind s, fmiInteger *value)
- DllExport fmiStatus fmiGetBooleanStatus (fmiComponent c, const fmiStatusKind s, fmiBoolean *value)
- DllExport fmiStatus fmiGetStringStatus (fmiComponent c, const fmiStatusKind s, fmiString *value)

9.58.1 Define Documentation

9.58.1.1 `#define MODEL_IDENTIFIER FMU_DUMMY_CS_MODEL_IDENTIFIER`

Definition at line 32 of file `fmu1_model_cs.c`.

9.58.2 Function Documentation

9.58.2.1 `DllExport const char* fmiGetVersion ( )`

Definition at line 39 of file `fmu1_model_cs.c`.

9.58.2.2 `DllExport fmiStatus fmiSetDebugLogging ( fmiComponent c, fmiBoolean loggingOn )`

Definition at line 44 of file `fmu1_model_cs.c`.

9.58.2.3 `DllExport fmiStatus fmiGetReal ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiReal value[] )`

Definition at line 49 of file `fmu1_model_cs.c`.

9.58.2.4 `DllExport fmiStatus fmiGetInteger ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiInteger value[] )`

Definition at line 54 of file `fmu1_model_cs.c`.

9.58.2.5 `DllExport fmiStatus fmiGetBoolean ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiBoolean value[] )`

Definition at line 59 of file `fmu1_model_cs.c`.

9.58.2.6 `DllExport fmiStatus fmiGetString ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiString value[] )`

Definition at line 64 of file `fmu1_model_cs.c`.

9.58.2.7 `DllExport fmiStatus fmiSetReal ( fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiReal value[] )`

Definition at line 69 of file `fmu1_model_cs.c`.

9.58.2.8 DllExport fmiStatus fmiSetInteger (fmiComponent *c*, const fmiValueReference *vr[]*, size_t *nvr*, const fmiInteger *value[]*)

Definition at line 74 of file fmu1_model_cs.c.

9.58.2.9 DllExport fmiStatus fmiSetBoolean (fmiComponent *c*, const fmiValueReference *vr[]*, size_t *nvr*, const fmiBoolean *value[]*)

Definition at line 79 of file fmu1_model_cs.c.

9.58.2.10 DllExport fmiStatus fmiSetString (fmiComponent *c*, const fmiValueReference *vr[]*, size_t *nvr*, const fmiString *value[]*)

Definition at line 84 of file fmu1_model_cs.c.

9.58.2.11 DllExport const char* fmiGetTypesPlatform ()

Definition at line 90 of file fmu1_model_cs.c.

9.58.2.12 DllExport fmiComponent fmiInstantiateSlave (fmiString *instanceName*, fmiString *fmuGUID*, fmiString *fmuLocation*, fmiString *mimeType*, fmiReal *timeout*, fmiBoolean *visible*, fmiBoolean *interactive*, fmiCallbackFunctions *functions*, fmiBoolean *loggingOn*)

Definition at line 95 of file fmu1_model_cs.c.

9.58.2.13 DllExport fmiStatus fmiInitializeSlave (fmiComponent *c*, fmiReal *tStart*, fmiBoolean *StopTimeDefined*, fmiReal *tStop*)

Definition at line 100 of file fmu1_model_cs.c.

9.58.2.14 DllExport fmiStatus fmiTerminateSlave (fmiComponent *c*)

Definition at line 105 of file fmu1_model_cs.c.

9.58.2.15 DllExport fmiStatus fmiResetSlave (fmiComponent *c*)

Definition at line 110 of file fmu1_model_cs.c.

9.58.2.16 DllExport void fmiFreeSlaveInstance (fmiComponent *c*)

Definition at line 115 of file fmu1_model_cs.c.

9.58.2.17 **DllExport fmiStatus fmiSetRealInputDerivatives (fmiComponent *c*, const fmiValueReference *vr*[], size_t *nvr*, const fmiInteger *order*[], const fmiReal *value*[])**

Definition at line 120 of file `fmu1_model_cs.c`.

9.58.2.18 **DllExport fmiStatus fmiGetRealOutputDerivatives (fmiComponent *c*, const fmiValueReference *vr*[], size_t *nvr*, const fmiInteger *order*[], fmiReal *value*[])**

Definition at line 125 of file `fmu1_model_cs.c`.

9.58.2.19 **DllExport fmiStatus fmiCancelStep (fmiComponent *c*)**

Definition at line 130 of file `fmu1_model_cs.c`.

9.58.2.20 **DllExport fmiStatus fmiDoStep (fmiComponent *c*, fmiReal *currentCommunicationPoint*, fmiReal *communicationStepSize*, fmiBoolean *newStep*)**

Definition at line 135 of file `fmu1_model_cs.c`.

9.58.2.21 **DllExport fmiStatus fmiGetStatus (fmiComponent *c*, const fmiStatusKind *s*, fmiStatus * *value*)**

Definition at line 140 of file `fmu1_model_cs.c`.

9.58.2.22 **DllExport fmiStatus fmiGetRealStatus (fmiComponent *c*, const fmiStatusKind *s*, fmiReal * *value*)**

Definition at line 145 of file `fmu1_model_cs.c`.

9.58.2.23 **DllExport fmiStatus fmiGetIntegerStatus (fmiComponent *c*, const fmiStatusKind *s*, fmiInteger * *value*)**

Definition at line 150 of file `fmu1_model_cs.c`.

9.58.2.24 **DllExport fmiStatus fmiGetBooleanStatus (fmiComponent *c*, const fmiStatusKind *s*, fmiBoolean * *value*)**

Definition at line 155 of file `fmu1_model_cs.c`.

9.58.2.25 `DllExport fmiStatus fmiGetStringStatus ( fmiComponent c, const fmiStatusKind s, fmiString * value )`

Definition at line 160 of file `fmu1_model_cs.c`.

9.59 Test/FMI1/fmu_dummy/fmu1_model_defines.h File Reference

Defines

- `#define STRINGIFY(a) #a`
- `#define STRINGIFY2(a) STRINGIFY(a)`
- `#define MODEL_IDENTIFIER_STR STRINGIFY2(MODEL_IDENTIFIER)`
- `#define BUFFER 1024`
- `#define MAGIC_TEST_VALUE 13.0 /* A test value for some functions */`
- `#define VAR_R_HIGHT 0`
- `#define VAR_R_HIGHT_SPEED 1`
- `#define VAR_R_GRATIVY 2`
- `#define VAR_R_BOUNCE_CONF 3`
- `#define EVENT_HIGHT 0`
- `#define VAR_S_LOGGER_TEST 0`
- `#define N_STATES 2`
- `#define N_EVENT_INDICATORS 1`
- `#define N_REAL 4`
- `#define N_INTEGER 4`
- `#define N_BOOLEAN 4`
- `#define N_STRING 4`
- `#define N_INPUT_REAL 2 /* CS only */`
- `#define N_INPUT_REAL_MAX_ORDER 2 /* CS only */`
- `#define N_OUTPUT_REAL 2 /* CS only */`
- `#define N_OUTPUT_REAL_MAX_ORDER 2 /* CS only */`
- `#define FMI_VERSION "1.0"`
- `#define FMI_GUID "123"`

9.59.1 Define Documentation

9.59.1.1 `#define STRINGIFY( a ) #a`

Definition at line 21 of file `fmu1_model_defines.h`.

9.59.1.2 `#define STRINGIFY2( a ) STRINGIFY(a)`

Definition at line 22 of file `fmu1_model_defines.h`.

9.59.1.3 **#define MODEL_IDENTIFIER_STR STRINGIFY2(MODEL_IDENTIFIER)**

Definition at line 23 of file fmu1_model_defines.h.

9.59.1.4 **#define BUFFER 1024**

Definition at line 25 of file fmu1_model_defines.h.

9.59.1.5 **#define MAGIC_TEST_VALUE 13.0 /* A test value for some functions */**

Definition at line 26 of file fmu1_model_defines.h.

9.59.1.6 **#define VAR_R_HIGHT 0**

Definition at line 31 of file fmu1_model_defines.h.

9.59.1.7 **#define VAR_R_HIGHT_SPEED 1**

Definition at line 32 of file fmu1_model_defines.h.

9.59.1.8 **#define VAR_R_GRATIVY 2**

Definition at line 34 of file fmu1_model_defines.h.

9.59.1.9 **#define VAR_R_BOUNCE_CONF 3**

Definition at line 35 of file fmu1_model_defines.h.

9.59.1.10 **#define EVENT_HIGHT 0**

Definition at line 38 of file fmu1_model_defines.h.

9.59.1.11 **#define VAR_S_LOGGER_TEST 0**

Definition at line 41 of file fmu1_model_defines.h.

9.59.1.12 **#define N_STATES 2**

Definition at line 44 of file fmu1_model_defines.h.

9.59.1.13 #define N_EVENT_INDICATORS 1

Definition at line 45 of file fmu1_model_defines.h.

9.59.1.14 #define N_REAL 4

Definition at line 46 of file fmu1_model_defines.h.

9.59.1.15 #define N_INTEGER 4

Definition at line 47 of file fmu1_model_defines.h.

9.59.1.16 #define N_BOOLEAN 4

Definition at line 48 of file fmu1_model_defines.h.

9.59.1.17 #define N_STRING 4

Definition at line 49 of file fmu1_model_defines.h.

9.59.1.18 #define N_INPUT_REAL 2 /* CS only */

Definition at line 51 of file fmu1_model_defines.h.

9.59.1.19 #define N_INPUT_REAL_MAX_ORDER 2 /* CS only */

Definition at line 52 of file fmu1_model_defines.h.

9.59.1.20 #define N_OUTPUT_REAL 2 /* CS only */

Definition at line 53 of file fmu1_model_defines.h.

9.59.1.21 #define N_OUTPUT_REAL_MAX_ORDER 2 /* CS only */

Definition at line 54 of file fmu1_model_defines.h.

9.59.1.22 #define FMI_VERSION "1.0"

Definition at line 57 of file fmu1_model_defines.h.

9.59.1.23 #define FMI_GUID "123"

Definition at line 67 of file fmu1_model_defines.h.

9.60 Test/FMI1/fmu_dummy/fmu1_model_me.c File Reference

```
#include <string.h>          #include <FMI1/fmiModelTypes.h> ×
#include <FMI1/fmiModelFunctions.h> #include <fmu_dummy/fmu1-
_model.h> #include "config_test.h" #include "fmu1_model.-
c"
```

Defines

- #define [MODEL_IDENTIFIER](#) FMU_DUMMY_ME_MODEL_IDENTIFIER

Enumerations

- enum [fmiStatusKind](#) { [fmiDoStepStatus](#), [fmiPendingStatus](#), [fmiLastSuccessfulTime](#) }

Functions

- DllExport const [char](#) * [fmiGetVersion](#) ()
- DllExport [fmiStatus](#) [fmiSetDebugLogging](#) ([fmiComponent](#) c, [fmiBoolean](#) loggingOn)
- DllExport [fmiStatus](#) [fmiGetReal](#) ([fmiComponent](#) c, const [fmiValueReference](#) vr[], [size_t](#) nvr, [fmiReal](#) value[])
- DllExport [fmiStatus](#) [fmiGetInteger](#) ([fmiComponent](#) c, const [fmiValueReference](#) vr[], [size_t](#) nvr, [fmiInteger](#) value[])
- DllExport [fmiStatus](#) [fmiGetBoolean](#) ([fmiComponent](#) c, const [fmiValueReference](#) vr[], [size_t](#) nvr, [fmiBoolean](#) value[])
- DllExport [fmiStatus](#) [fmiGetString](#) ([fmiComponent](#) c, const [fmiValueReference](#) vr[], [size_t](#) nvr, [fmiString](#) value[])
- DllExport [fmiStatus](#) [fmiSetReal](#) ([fmiComponent](#) c, const [fmiValueReference](#) vr[], [size_t](#) nvr, const [fmiReal](#) value[])
- DllExport [fmiStatus](#) [fmiSetInteger](#) ([fmiComponent](#) c, const [fmiValueReference](#) vr[], [size_t](#) nvr, const [fmiInteger](#) value[])
- DllExport [fmiStatus](#) [fmiSetBoolean](#) ([fmiComponent](#) c, const [fmiValueReference](#) vr[], [size_t](#) nvr, const [fmiBoolean](#) value[])
- DllExport [fmiStatus](#) [fmiSetString](#) ([fmiComponent](#) c, const [fmiValueReference](#) vr[], [size_t](#) nvr, const [fmiString](#) value[])
- DllExport const [char](#) * [fmiGetModelTypesPlatform](#) ()
- DllExport [fmiComponent](#) [fmiInstantiateModel](#) ([fmiString](#) instanceName, [fmiString](#) GUID, [fmiCallbackFunctions](#) functions, [fmiBoolean](#) loggingOn)
- DllExport void [fmiFreeModelInstance](#) ([fmiComponent](#) c)

- DllExport fmiStatus [fmiSetTime](#) (fmiComponent c, fmiReal fmitime)
- DllExport fmiStatus [fmiSetContinuousStates](#) (fmiComponent c, const fmiReal x[], size_t nx)
- DllExport fmiStatus [fmiCompletedIntegratorStep](#) (fmiComponent c, fmiBoolean *callEventUpdate)
- DllExport fmiStatus [fmiInitialize](#) (fmiComponent c, fmiBoolean toleranceControlled, fmiReal relativeTolerance, fmiEventInfo *eventInfo)
- DllExport fmiStatus [fmiGetDerivatives](#) (fmiComponent c, fmiReal derivatives[], size_t nx)
- DllExport fmiStatus [fmiGetEventIndicators](#) (fmiComponent c, fmiReal eventIndicators[], size_t ni)
- DllExport fmiStatus [fmiEventUpdate](#) (fmiComponent c, fmiBoolean intermediateResults, fmiEventInfo *eventInfo)
- DllExport fmiStatus [fmiGetContinuousStates](#) (fmiComponent c, fmiReal states[], size_t nx)
- DllExport fmiStatus [fmiGetNominalContinuousStates](#) (fmiComponent c, fmiReal x_nominal[], size_t nx)
- DllExport fmiStatus [fmiGetStateValueReferences](#) (fmiComponent c, fmiValueReference vrx[], size_t nx)
- DllExport fmiStatus [fmiTerminate](#) (fmiComponent c)

9.60.1 Define Documentation

9.60.1.1 `#define MODEL_IDENTIFIER FMU_DUMMY_ME.MODEL_IDENTIFIER`

Definition at line 36 of file fmu1_model_me.c.

9.60.2 Enumeration Type Documentation

9.60.2.1 `enum fmiStatusKind`

Enumerator:

fmiDoStepStatus
fmiPendingStatus
fmiLastSuccessfulTime

Definition at line 30 of file fmu1_model_me.c.

9.60.3 Function Documentation

9.60.3.1 `DllExport const char* fmiGetVersion ( )`

Definition at line 39 of file fmu1_model_me.c.

9.60.3.2 `DllExport fmiStatus fmiSetDebugLogging ( fmiComponent c, fmiBoolean loggingOn )`

Definition at line 44 of file `fmu1_model_me.c`.

9.60.3.3 `DllExport fmiStatus fmiGetReal ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiReal value[] )`

Definition at line 49 of file `fmu1_model_me.c`.

9.60.3.4 `DllExport fmiStatus fmiGetInteger ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiInteger value[] )`

Definition at line 54 of file `fmu1_model_me.c`.

9.60.3.5 `DllExport fmiStatus fmiGetBoolean ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiBoolean value[] )`

Definition at line 59 of file `fmu1_model_me.c`.

9.60.3.6 `DllExport fmiStatus fmiGetString ( fmiComponent c, const fmiValueReference vr[], size_t nvr, fmiString value[] )`

Definition at line 64 of file `fmu1_model_me.c`.

9.60.3.7 `DllExport fmiStatus fmiSetReal ( fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiReal value[] )`

Definition at line 69 of file `fmu1_model_me.c`.

9.60.3.8 `DllExport fmiStatus fmiSetInteger ( fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiInteger value[] )`

Definition at line 74 of file `fmu1_model_me.c`.

9.60.3.9 `DllExport fmiStatus fmiSetBoolean ( fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiBoolean value[] )`

Definition at line 79 of file `fmu1_model_me.c`.

9.60.3.10 `DllExport fmiStatus fmiSetString ( fmiComponent c, const fmiValueReference vr[], size_t nvr, const fmiString value[] )`

Definition at line 84 of file `fmu1_model_me.c`.

9.60.3.11 DllExport const char* fmiGetModelTypesPlatform ()

Definition at line 90 of file fmu1_model_me.c.

**9.60.3.12 DllExport fmiComponent fmiInstantiateModel (fmiString *instanceName*,
fmiString *GUID*, fmiCallbackFunctions *functions*, fmiBoolean *loggingOn*)**

Definition at line 95 of file fmu1_model_me.c.

9.60.3.13 DllExport void fmiFreeModelInstance (fmiComponent *c*)

Definition at line 100 of file fmu1_model_me.c.

9.60.3.14 DllExport fmiStatus fmiSetTime (fmiComponent *c*, fmiReal *fmitime*)

Definition at line 105 of file fmu1_model_me.c.

**9.60.3.15 DllExport fmiStatus fmiSetContinuousStates (fmiComponent *c*, const
fmiReal *x[]*, size_t *nx*)**

Definition at line 110 of file fmu1_model_me.c.

**9.60.3.16 DllExport fmiStatus fmiCompletedIntegratorStep (fmiComponent *c*,
fmiBoolean * *callEventUpdate*)**

Definition at line 115 of file fmu1_model_me.c.

**9.60.3.17 DllExport fmiStatus fmiInitialize (fmiComponent *c*, fmiBoolean
toleranceControlled, fmiReal *relativeTolerance*, fmiEventInfo * *eventInfo*)**

Definition at line 120 of file fmu1_model_me.c.

**9.60.3.18 DllExport fmiStatus fmiGetDerivatives (fmiComponent *c*, fmiReal
derivatives[], size_t *nx*)**

Definition at line 125 of file fmu1_model_me.c.

**9.60.3.19 DllExport fmiStatus fmiGetEventIndicators (fmiComponent *c*, fmiReal
eventIndicators[], size_t *ni*)**

Definition at line 130 of file fmu1_model_me.c.

9.60.3.20 DllExport fmiStatus fmiEventUpdate (fmiComponent *c*, fmiBoolean *intermediateResults*, fmiEventInfo * *eventInfo*)

Definition at line 135 of file fmu1_model_me.c.

9.60.3.21 DllExport fmiStatus fmiGetContinuousStates (fmiComponent *c*, fmiReal *states*[], size_t *nx*)

Definition at line 140 of file fmu1_model_me.c.

9.60.3.22 DllExport fmiStatus fmiGetNominalContinuousStates (fmiComponent *c*, fmiReal *x_nominal*[], size_t *nx*)

Definition at line 145 of file fmu1_model_me.c.

9.60.3.23 DllExport fmiStatus fmiGetStateValueReferences (fmiComponent *c*, fmiValueReference *vr*[], size_t *nx*)

Definition at line 150 of file fmu1_model_me.c.

9.60.3.24 DllExport fmiStatus fmiTerminate (fmiComponent *c*)

Definition at line 155 of file fmu1_model_me.c.

9.61 Test/FMI1/jm_vector_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h>
#include "config_test.h" #include <JM/jm_vector.h>
#include <JM/jm_stack.h>
```

Defines

- #define TESTVAL 49
- #define VINIT_SIZE 5

Functions

- void print_int (int i, void *data)
- void print_dbl (double d, void *data)
- int compar_int (int *a, int *b)
- void log_error (const char *fmt,...)
- int main ()

Variables

- int `return_code` = CTEST_RETURN_SUCCESS

9.61.1 Define Documentation

9.61.1.1 `#define TESTVAL 49`

Definition at line 31 of file `jm_vector_test.c`.

9.61.1.2 `#define VINIT_SIZE 5`

9.61.2 Function Documentation

9.61.2.1 `void print_int ( int i, void * data )`

Definition at line 9 of file `jm_vector_test.c`.

9.61.2.2 `void print_dbl ( double d, void * data )`

Definition at line 13 of file `jm_vector_test.c`.

9.61.2.3 `int compar_int ( int * a, int * b )`

Definition at line 17 of file `jm_vector_test.c`.

9.61.2.4 `void log_error ( const char * fmt, ... )`

Definition at line 23 of file `jm_vector_test.c`.

9.61.2.5 `int main ( )`

Definition at line 33 of file `jm_vector_test.c`.

9.61.3 Variable Documentation

9.61.3.1 `int return_code = CTEST_RETURN_SUCCESS`

Definition at line 21 of file `jm_vector_test.c`.

9.62 Test/jm_vector_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.-  
h> #include "config_test.h" #include <JM/jm_vector.h> ×  
#include <JM/jm_stack.h>
```

Defines

- #define [TESTVAL](#) 49
- #define [VINIT_SIZE](#) 5

Functions

- void [print_int](#) (int i, void *data)
- void [print_dbl](#) (double d, void *data)
- int [compar_int](#) (int *a, int *b)
- void [log_error](#) (const [char](#) *fmt,...)
- int [main](#) ()

Variables

- int [return_code](#) = CTEST_RETURN_SUCCESS

9.62.1 Define Documentation

9.62.1.1 #define TESTVAL 49

Definition at line 31 of file jm_vector_test.c.

9.62.1.2 #define VINIT_SIZE 5

9.62.2 Function Documentation

9.62.2.1 void print_int (int i, void * data)

Definition at line 9 of file jm_vector_test.c.

9.62.2.2 void print_dbl (double d, void * data)

Definition at line 13 of file jm_vector_test.c.

9.62.2.3 `int compar_int ( int * a, int * b )`

Definition at line 17 of file `jm_vector_test.c`.

9.62.2.4 `void log_error ( const char * fmt, ... )`

Definition at line 23 of file `jm_vector_test.c`.

9.62.2.5 `int main ( )`

Definition at line 33 of file `jm_vector_test.c`.

9.62.3 Variable Documentation

9.62.3.1 `int return_code = CTEST_RETURN_SUCCESS`

Definition at line 21 of file `jm_vector_test.c`.

9.63 Test/FMI2/fmi2_import_cs_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h>
#include "config_test.h" #include <fmilib.h> #include
<JM/jm_portability.h>
```

Defines

- `#define BUFFER 1000`

Functions

- `void importlogger (jm_callbacks *c, jm_string module, jm_log_level_enumeration_t log_level, jm_string message)`
- `void fmilogger (fmi2_component_t c, fmi2_string_t instanceName, fmi2_status_t status, fmi2_string_t category, fmi2_string_t message,...)`
- `void do_exit (int code)`
- `int test_simulate_cs (fmi2_import_t *fmu)`
- `int main (int argc, char *argv[])`

9.63.1 Define Documentation

9.63.1.1 `#define BUFFER 1000`

Definition at line 26 of file `fmi2_import_cs_test.c`.

9.63.2 Function Documentation

9.63.2.1 `void importlogger ( jm_callbacks * c, jm_string module, jm_log_level_enu_t log_level, jm_string message )`

Definition at line 28 of file fmi2_import_cs_test.c.

9.63.2.2 `void fmilogger ( fmi2_component_t c, fmi2_string_t instanceName, fmi2_status_t status, fmi2_string_t category, fmi2_string_t message, ... )`

Definition at line 35 of file fmi2_import_cs_test.c.

9.63.2.3 `void do_exit ( int code )`

Definition at line 46 of file fmi2_import_cs_test.c.

9.63.2.4 `int test_simulate_cs ( fmi2_import_t * fmu )`

Definition at line 53 of file fmi2_import_cs_test.c.

9.63.2.5 `int main ( int argc, char * argv[] )`

Definition at line 165 of file fmi2_import_cs_test.c.

9.64 Test/FMI2/fmi2_import_me_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h>
#include "config_test.h" #include <fmilib.h>
```

Defines

- `#define BUFFER 1000`

Functions

- `void do_exit (int code)`
- `void do_event_iteration (fmi2_import_t *fmu, fmi2_event_info_t *eventInfo)`
- `int test_simulate_me (fmi2_import_t *fmu)`
- `int main (int argc, char *argv[])`

9.64.1 Define Documentation

9.64.1.1 `#define BUFFER 1000`

Definition at line 23 of file `fmi2_import_me_test.c`.

9.64.2 Function Documentation

9.64.2.1 `void do_exit ( int code )`

Definition at line 25 of file `fmi2_import_me_test.c`.

9.64.2.2 `void do_event_iteration ( fmi2_import_t * fmu, fmi2_event_info_t * eventInfo )`

Definition at line 32 of file `fmi2_import_me_test.c`.

9.64.2.3 `int test_simulate_me ( fmi2_import_t * fmu )`

Definition at line 41 of file `fmi2_import_me_test.c`.

9.64.2.4 `int main ( int argc, char * argv[] )`

Definition at line 196 of file `fmi2_import_me_test.c`.

9.65 Test/FMI2/fmi2_import_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.h>
#include <config_test.h> #include <fmilib.h> #include
<JM/jm_portability.h>
```

Defines

- `#define BUFFER 1000`

Functions

- static void `fmi2logger` (`fmi2_component_environment_t` env, `fmi2_string_t` instanceName, `fmi2_status_t` status, `fmi2_string_t` category, `fmi2_string_t` message,...)
- static void `stepFinished` (`fmi2_component_environment_t` env, `fmi2_status_t` status)
- int `fmi2_test` (`fmi_import_context_t` *context, const `char` *dirPath)

9.65.1 Define Documentation

9.65.1.1 #define BUFFER 1000

Definition at line 24 of file fmi2_import_test.c.

9.65.2 Function Documentation

9.65.2.1 static void fmi2logger (fmi2_component_environment_t env, fmi2_string_t instanceName, fmi2_status_t status, fmi2_string_t category, fmi2_string_t message, ...) [static]

Definition at line 27 of file fmi2_import_test.c.

9.65.2.2 static void stepFinished (fmi2_component_environment_t env, fmi2_status_t status) [static]

Definition at line 37 of file fmi2_import_test.c.

9.65.2.3 int fmi2_test (fmi_import_context_t * context, const char * dirPath)

Definition at line 41 of file fmi2_import_test.c.

9.66 Test/FMI2/fmu_dummy/fmu2_model.c File Reference

```
#include <stdio.h>  #include <string.h>  #include <fmu_
dummy/fmu2_model.h>
```

Functions

- static int [calc_initialize](#) (component_ptr_t comp)
- static int [calc_get_derivatives](#) (component_ptr_t comp)
- static int [calc_get_event_indicators](#) (component_ptr_t comp)
- static int [calc_event_update](#) (component_ptr_t comp)
- const char * [fmi_get_version](#) ()
- fmi2Status [fmi_set_debug_logging](#) (fmi2Component c, fmi2Boolean loggingOn)
- fmi2Status [fmi_get_real](#) (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Real value[])
- fmi2Status [fmi_get_integer](#) (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Integer value[])
- fmi2Status [fmi_get_boolean](#) (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Boolean value[])
- fmi2Status [fmi_get_string](#) (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2String value[])

- fmi2Status `fmi_set_real` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Real value[])
- fmi2Status `fmi_set_integer` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer value[])
- fmi2Status `fmi_set_boolean` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Boolean value[])
- fmi2Status `fmi_set_string` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2String value[])
- const char * `fmi_get_model_types_platform` ()
- fmi2Component `fmi_instantiate` (fmi2String instanceName, fmi2Type fmuType, fmi2String fmuGUID, fmi2String fmuLocation, const fmi2CallbackFunctions *functions, fmi2Boolean visible, fmi2Boolean loggingOn)
- void `fmi_free_instance` (fmi2Component c)
- fmi2Status `fmi_setup_experiment` (fmi2Component c, fmi2Boolean toleranceDefined, fmi2Real tolerance, fmi2Real startTime, fmi2Boolean stopTimeDefined, fmi2Real stopTime)
- fmi2Status `fmi_enter_initialization_mode` (fmi2Component c)
- fmi2Status `fmi_exit_initialization_mode` (fmi2Component c)
- fmi2Status `fmi_enter_event_mode` (fmi2Component c)
- fmi2Status `fmi_new_discrete_states` (fmi2Component c, fmi2EventInfo *eventInfo)
- fmi2Status `fmi_enter_continuous_time_mode` (fmi2Component c)
- fmi2Status `fmi_set_time` (fmi2Component c, fmi2Real fmiTime)
- fmi2Status `fmi_set_continuous_states` (fmi2Component c, const fmi2Real x[], size_t nx)
- fmi2Status `fmi_completed_integrator_step` (fmi2Component c, fmi2Boolean noSetFMUStatePriorToCurrentPoint, fmi2Boolean *enterEventMode, fmi2Boolean *terminateSimulation)
- fmi2Status `fmi_get_derivatives` (fmi2Component c, fmi2Real derivatives[], size_t nx)
- fmi2Status `fmi_get_event_indicators` (fmi2Component c, fmi2Real eventIndicators[], size_t ni)
- fmi2Status `fmi_get_continuous_states` (fmi2Component c, fmi2Real states[], size_t nx)
- fmi2Status `fmi_get_nominals_of_continuousstates` (fmi2Component c, fmi2Real x_nominal[], size_t nx)
- fmi2Status `fmi_terminate` (fmi2Component c)
- const char * `fmi_get_types_platform` ()
- fmi2Status `fmi_reset` (fmi2Component c)
- fmi2Status `fmi_set_real_input_derivatives` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer order[], const fmi2Real value[])
- fmi2Status `fmi_get_real_output_derivatives` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer order[], fmi2Real value[])
- fmi2Status `fmi_cancel_step` (fmi2Component c)
- fmi2Status `fmi_do_step` (fmi2Component c, fmi2Real currentCommunicationPoint, fmi2Real communicationStepSize, fmi2Boolean newStep)
- fmi2Status `fmi_get_status` (fmi2Component c, const fmi2StatusKind s, fmi2Status *value)

- fmi2Status [fmi_get_real_status](#) (fmi2Component *c*, const fmi2StatusKind *s*, [fmi2Real](#) *value)
- fmi2Status [fmi_get_integer_status](#) (fmi2Component *c*, const fmi2StatusKind *s*, [fmi2Integer](#) *value)
- fmi2Status [fmi_get_boolean_status](#) (fmi2Component *c*, const fmi2StatusKind *s*, [fmi2Boolean](#) *value)
- fmi2Status [fmi_get_string_status](#) (fmi2Component *c*, const fmi2StatusKind *s*, [fmi2String](#) *value)

9.66.1 Function Documentation

9.66.1.1 `static int calc_initialize ( component_ptr_t comp ) [static]`

Definition at line 23 of file `fmu2_model.c`.

9.66.1.2 `static int calc_get_derivatives ( component_ptr_t comp ) [static]`

Definition at line 43 of file `fmu2_model.c`.

9.66.1.3 `static int calc_get_event_indicators ( component_ptr_t comp ) [static]`

Definition at line 50 of file `fmu2_model.c`.

9.66.1.4 `static int calc_event_update ( component_ptr_t comp ) [static]`

Definition at line 57 of file `fmu2_model.c`.

9.66.1.5 `const char* fmi_get_version ( )`

Definition at line 78 of file `fmu2_model.c`.

9.66.1.6 `fmi2Status fmi_set_debug_logging ( fmi2Component c, fmi2Boolean loggingOn )`

Definition at line 83 of file `fmu2_model.c`.

9.66.1.7 `fmi2Status fmi_get_real ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Real value[] )`

Definition at line 94 of file `fmu2_model.c`.

9.66.1.8 `fmi2Status fmi_get_integer ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Integer value[] )`

Definition at line 118 of file fmu2_model.c.

9.66.1.9 `fmi2Status fmi_get_boolean ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Boolean value[] )`

Definition at line 132 of file fmu2_model.c.

9.66.1.10 `fmi2Status fmi_get_string ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2String value[] )`

Definition at line 146 of file fmu2_model.c.

9.66.1.11 `fmi2Status fmi_set_real ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Real value[] )`

Definition at line 160 of file fmu2_model.c.

9.66.1.12 `fmi2Status fmi_set_integer ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer value[] )`

Definition at line 184 of file fmu2_model.c.

9.66.1.13 `fmi2Status fmi_set_boolean ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Boolean value[] )`

Definition at line 198 of file fmu2_model.c.

9.66.1.14 `fmi2Status fmi_set_string ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2String value[] )`

Definition at line 212 of file fmu2_model.c.

9.66.1.15 `const char* fmi_get_model_types_platform ( )`

Definition at line 251 of file fmu2_model.c.

9.66.1.16 **fmi2Component fmi_instantiate** (*fmi2String instanceName*,
fmi2Type fmuType, *fmi2String fmuGUID*, *fmi2String fmuLocation*, *const*
*fmi2CallbackFunctions * functions*, *fmi2Boolean visible*, *fmi2Boolean*
loggingOn)

Definition at line 260 of file `fmu2_model.c`.

9.66.1.17 **void fmi_free_instance** (*fmi2Component c*)

Definition at line 312 of file `fmu2_model.c`.

9.66.1.18 **fmi2Status fmi_setup_experiment** (*fmi2Component c*, *fmi2Boolean*
toleranceDefined, *fmi2Real tolerance*, *fmi2Real startTime*, *fmi2Boolean*
stopTimeDefined, *fmi2Real stopTime*)

Definition at line 323 of file `fmu2_model.c`.

9.66.1.19 **fmi2Status fmi_enter_initialization_mode** (*fmi2Component c*)

Definition at line 344 of file `fmu2_model.c`.

9.66.1.20 **fmi2Status fmi_exit_initialization_mode** (*fmi2Component c*)

Definition at line 354 of file `fmu2_model.c`.

9.66.1.21 **fmi2Status fmi_enter_event_mode** (*fmi2Component c*)

Definition at line 359 of file `fmu2_model.c`.

9.66.1.22 **fmi2Status fmi_new_discrete_states** (*fmi2Component c*, *fmi2EventInfo **
eventInfo)

Definition at line 364 of file `fmu2_model.c`.

9.66.1.23 **fmi2Status fmi_enter_continuous_time_mode** (*fmi2Component c*)

Definition at line 377 of file `fmu2_model.c`.

9.66.1.24 **fmi2Status fmi_set_time** (*fmi2Component c*, *fmi2Real fmitime*)

Definition at line 382 of file `fmu2_model.c`.

9.66.1.25 **fmi2Status fmi_set_continuous_states** (**fmi2Component** *c*, **const fmi2Real** *x[]*, **size_t** *nx*)

Definition at line 393 of file fmu2_model.c.

9.66.1.26 **fmi2Status fmi_completed_integrator_step** (**fmi2Component** *c*, **fmi2Boolean** *noSetFMUStatePriorToCurrentPoint*, **fmi2Boolean** * *enterEventMode*, **fmi2Boolean** * *terminateSimulation*)

Definition at line 407 of file fmu2_model.c.

9.66.1.27 **fmi2Status fmi_get_derivatives** (**fmi2Component** *c*, **fmi2Real** *derivatives[]*, **size_t** *nx*)

Definition at line 420 of file fmu2_model.c.

9.66.1.28 **fmi2Status fmi_get_event_indicators** (**fmi2Component** *c*, **fmi2Real** *eventIndicators[]*, **size_t** *ni*)

Definition at line 437 of file fmu2_model.c.

9.66.1.29 **fmi2Status fmi_get_continuous_states** (**fmi2Component** *c*, **fmi2Real** *states[]*, **size_t** *nx*)

Definition at line 454 of file fmu2_model.c.

9.66.1.30 **fmi2Status fmi_get_nominals_of_continuousstates** (**fmi2Component** *c*, **fmi2Real** *x.nominal[]*, **size_t** *nx*)

Definition at line 469 of file fmu2_model.c.

9.66.1.31 **fmi2Status fmi_terminate** (**fmi2Component** *c*)

Definition at line 483 of file fmu2_model.c.

9.66.1.32 **const char*** **fmi_get_types_platform** ()

Definition at line 494 of file fmu2_model.c.

9.66.1.33 **fmi2Status fmi_reset** (**fmi2Component** *c*)

Definition at line 499 of file fmu2_model.c.

9.66.1.34 **fmi2Status fmi_set_real_input_derivatives** (**fmi2Component** *c*, **const** **fmi2ValueReference** *vr[]*, **size_t** *nvr*, **const** **fmi2Integer** *order[]*, **const** **fmi2Real** *value[]*)

Definition at line 504 of file `fmu2_model.c`.

9.66.1.35 **fmi2Status fmi_get_real_output_derivatives** (**fmi2Component** *c*, **const** **fmi2ValueReference** *vr[]*, **size_t** *nvr*, **const** **fmi2Integer** *order[]*, **fmi2Real** *value[]*)

Definition at line 520 of file `fmu2_model.c`.

9.66.1.36 **fmi2Status fmi_cancel_step** (**fmi2Component** *c*)

Definition at line 532 of file `fmu2_model.c`.

9.66.1.37 **fmi2Status fmi_do_step** (**fmi2Component** *c*, **fmi2Real** *currentCommunicationPoint*, **fmi2Real** *communicationStepSize*, **fmi2Boolean** *newStep*)

Definition at line 537 of file `fmu2_model.c`.

9.66.1.38 **fmi2Status fmi_get_status** (**fmi2Component** *c*, **const** **fmi2StatusKind** *s*, **fmi2Status** * *value*)

Definition at line 637 of file `fmu2_model.c`.

9.66.1.39 **fmi2Status fmi_get_real_status** (**fmi2Component** *c*, **const** **fmi2StatusKind** *s*, **fmi2Real** * *value*)

Definition at line 649 of file `fmu2_model.c`.

9.66.1.40 **fmi2Status fmi_get_integer_status** (**fmi2Component** *c*, **const** **fmi2StatusKind** *s*, **fmi2Integer** * *value*)

Definition at line 661 of file `fmu2_model.c`.

9.66.1.41 **fmi2Status fmi_get_boolean_status** (**fmi2Component** *c*, **const** **fmi2StatusKind** *s*, **fmi2Boolean** * *value*)

Definition at line 669 of file `fmu2_model.c`.

9.66.1.42 `fmi2Status fmi_get_string_status ( fmi2Component c, const fmi2StatusKind s, fmi2String * value )`

Definition at line 677 of file `fmu2_model.c`.

9.67 Test/FMI2/fmu_dummy/fmu2_model.h File Reference

```
#include <FMI2/fmi2Functions.h> #include <fmu_dummy/fmu2-
_model_defines.h>
```

Data Structures

- struct `component_t`

Defines

- #define `FMI2_Export` `DllExport`

Typedefs

- typedef `component_t * component_ptr_t`

Functions

- const `char * fmi_get_version ()`
- `fmi2Status fmi_set_debug_logging (fmi2Component c, fmi2Boolean loggingOn)`
- `fmi2Component fmi_instantiate (fmi2String instanceName, fmi2Type fmuType, fmi2String fmuGUID, fmi2String fmuLocation, const fmi2CallbackFunctions *functions, fmi2Boolean visible, fmi2Boolean loggingOn)`
- void `fmi_free_instance (fmi2Component c)`
- `fmi2Status fmi_setup_experiment (fmi2Component c, fmi2Boolean tolerance-Defined, fmi2Real tolerance, fmi2Real startTime, fmi2Boolean stopTimeDefined, fmi2Real stopTime)`
- `fmi2Status fmi_enter_initialization_mode (fmi2Component c)`
- `fmi2Status fmi_exit_initialization_mode (fmi2Component c)`
- `fmi2Status fmi_terminate (fmi2Component c)`
- `fmi2Status fmi_reset (fmi2Component c)`
- `fmi2Status fmi_get_real (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Real value[])`
- `fmi2Status fmi_get_integer (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Integer value[])`
- `fmi2Status fmi_get_boolean (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Boolean value[])`

- fmi2Status `fmi_get_string` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2String value[])
- fmi2Status `fmi_set_real` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Real value[])
- fmi2Status `fmi_set_integer` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer value[])
- fmi2Status `fmi_set_boolean` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Boolean value[])
- fmi2Status `fmi_set_string` (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2String value[])
- const char * `fmi_get_model_types_platform` ()
- fmi2Status `fmi_enter_event_mode` (fmi2Component c)
- fmi2Status `fmi_new_discrete_states` (fmi2Component c, fmi2EventInfo *eventInfo)
- fmi2Status `fmi_enter_continuous_time_mode` (fmi2Component c)
- fmi2Status `fmi_set_time` (fmi2Component c, fmi2Real fmitime)
- fmi2Status `fmi_set_continuous_states` (fmi2Component c, const fmi2Real x[], size_t nx)
- fmi2Status `fmi_completed_integrator_step` (fmi2Component c, fmi2Boolean noSetFMUStatePriorToCurrentPoint, fmi2Boolean *enterEventMode, fmi2Boolean *terminateSimulation)
- fmi2Status `fmi_get_derivatives` (fmi2Component c, fmi2Real derivatives[], size_t nx)
- fmi2Status `fmi_get_event_indicators` (fmi2Component c, fmi2Real eventIndicators[], size_t ni)
- fmi2Status `fmi_get_continuous_states` (fmi2Component c, fmi2Real states[], size_t nx)
- fmi2Status `fmi_get_nominals_of_continuousstates` (fmi2Component c, fmi2Real x_nominal[], size_t nx)

9.67.1 Define Documentation

9.67.1.1 #define FMI2_Export DllExport

Definition at line 24 of file fmu2_model.h.

9.67.2 Typedef Documentation

9.67.2.1 typedef component_t* component_ptr_t

Definition at line 71 of file fmu2_model.h.

9.67.3 Function Documentation

9.67.3.1 const char* fmi_get_version ()

Definition at line 77 of file fmu1_model.c.

9.67.3.2 **fmi2Status fmi_set_debug_logging** (**fmi2Component** *c*, **fmi2Boolean** *loggingOn*)

Definition at line 83 of file fmu2_model.c.

9.67.3.3 **fmi2Component fmi_instantiate** (**fmi2String** *instanceName*, **fmi2Type** *fmuType*, **fmi2String** *fmuGUID*, **fmi2String** *fmuLocation*, **const** **fmi2CallbackFunctions** * *functions*, **fmi2Boolean** *visible*, **fmi2Boolean** *loggingOn*)

Definition at line 260 of file fmu2_model.c.

9.67.3.4 **void fmi_free_instance** (**fmi2Component** *c*)

Definition at line 312 of file fmu2_model.c.

9.67.3.5 **fmi2Status fmi_setup_experiment** (**fmi2Component** *c*, **fmi2Boolean** *toleranceDefined*, **fmi2Real** *tolerance*, **fmi2Real** *startTime*, **fmi2Boolean** *stopTimeDefined*, **fmi2Real** *stopTime*)

Definition at line 323 of file fmu2_model.c.

9.67.3.6 **fmi2Status fmi_enter_initialization_mode** (**fmi2Component** *c*)

Definition at line 344 of file fmu2_model.c.

9.67.3.7 **fmi2Status fmi_exit_initialization_mode** (**fmi2Component** *c*)

Definition at line 354 of file fmu2_model.c.

9.67.3.8 **fmi2Status fmi_terminate** (**fmi2Component** *c*)

Definition at line 483 of file fmu2_model.c.

9.67.3.9 **fmi2Status fmi_reset** (**fmi2Component** *c*)

Definition at line 499 of file fmu2_model.c.

9.67.3.10 **fmi2Status fmi_get_real** (**fmi2Component** *c*, **const** **fmi2ValueReference** *vr[]*, **size_t** *nvr*, **fmi2Real** *value[]*)

Definition at line 94 of file fmu2_model.c.

9.67.3.11 `fmi2Status fmi_get_integer ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Integer value[] )`

Definition at line 118 of file `fmu2_model.c`.

9.67.3.12 `fmi2Status fmi_get_boolean ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Boolean value[] )`

Definition at line 132 of file `fmu2_model.c`.

9.67.3.13 `fmi2Status fmi_get_string ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2String value[] )`

Definition at line 146 of file `fmu2_model.c`.

9.67.3.14 `fmi2Status fmi_set_real ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Real value[] )`

Definition at line 160 of file `fmu2_model.c`.

9.67.3.15 `fmi2Status fmi_set_integer ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer value[] )`

Definition at line 184 of file `fmu2_model.c`.

9.67.3.16 `fmi2Status fmi_set_boolean ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Boolean value[] )`

Definition at line 198 of file `fmu2_model.c`.

9.67.3.17 `fmi2Status fmi_set_string ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2String value[] )`

Definition at line 212 of file `fmu2_model.c`.

9.67.3.18 `const char* fmi_get_model_types_platform ( )`

Definition at line 242 of file `fmu1_model.c`.

9.67.3.19 `fmi2Status fmi_enter_event_mode ( fmi2Component c )`

Definition at line 359 of file `fmu2_model.c`.

9.67.3.20 **fmi2Status fmi_new_discrete_states** (**fmi2Component** *c*, **fmi2EventInfo** * *eventInfo*)

Definition at line 364 of file `fmu2_model.c`.

9.67.3.21 **fmi2Status fmi_enter_continuous_time_mode** (**fmi2Component** *c*)

Definition at line 377 of file `fmu2_model.c`.

9.67.3.22 **fmi2Status fmi_set_time** (**fmi2Component** *c*, **fmi2Real** *fmiTime*)

Definition at line 382 of file `fmu2_model.c`.

9.67.3.23 **fmi2Status fmi_set_continuous_states** (**fmi2Component** *c*, **const fmi2Real** *x*[], **size_t** *nx*)

Definition at line 393 of file `fmu2_model.c`.

9.67.3.24 **fmi2Status fmi_completed_integrator_step** (**fmi2Component** *c*, **fmi2Boolean** *noSetFMUStatePriorToCurrentPoint*, **fmi2Boolean** * *enterEventMode*, **fmi2Boolean** * *terminateSimulation*)

Definition at line 407 of file `fmu2_model.c`.

9.67.3.25 **fmi2Status fmi_get_derivatives** (**fmi2Component** *c*, **fmi2Real** *derivatives*[], **size_t** *nx*)

Definition at line 420 of file `fmu2_model.c`.

9.67.3.26 **fmi2Status fmi_get_event_indicators** (**fmi2Component** *c*, **fmi2Real** *eventIndicators*[], **size_t** *ni*)

Definition at line 437 of file `fmu2_model.c`.

9.67.3.27 **fmi2Status fmi_get_continuous_states** (**fmi2Component** *c*, **fmi2Real** *states*[], **size_t** *nx*)

Definition at line 454 of file `fmu2_model.c`.

9.67.3.28 **fmi2Status fmi_get_nominals_of_continuousstates** (**fmi2Component** *c*, **fmi2Real** *x.nominal*[], **size_t** *nx*)

Definition at line 469 of file `fmu2_model.c`.

9.68 Test/FMI2/fmu_dummy/fmu2_model_cs.c File Reference

```
#include <string.h>          #include <FMI2/fmi2Functions.h> ×
#include <fmu_dummy/fmu2_model.h> #include "config_test.-
h" #include "fmu2_model.c"
```

Functions

- [FMI2_Export](#) `const char * fmi2GetVersion ()`
- [FMI2_Export](#) `fmi2Status fmi2SetDebugLogging (fmi2Component c, fmi2Boolean loggingOn, size_t n, const fmi2String cat[])`
- [FMI2_Export](#) `fmi2Component fmi2Instantiate (fmi2String instanceName, fmi2Type fmuType, fmi2String GUID, fmi2String location, const fmi2CallbackFunctions *functions, fmi2Boolean visible, fmi2Boolean loggingOn)`
- [FMI2_Export](#) `void fmi2FreeInstance (fmi2Component c)`
- [FMI2_Export](#) `fmi2Status fmi2SetupExperiment (fmi2Component c, fmi2Boolean toleranceDefined, fmi2Real tolerance, fmi2Real startTime, fmi2Boolean stopTimeDefined, fmi2Real stopTime)`
- [FMI2_Export](#) `fmi2Status fmi2EnterInitializationMode (fmi2Component c)`
- [FMI2_Export](#) `fmi2Status fmi2ExitInitializationMode (fmi2Component c)`
- [FMI2_Export](#) `fmi2Status fmi2GetReal (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Real value[])`
- [FMI2_Export](#) `fmi2Status fmi2GetInteger (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Integer value[])`
- [FMI2_Export](#) `fmi2Status fmi2GetBoolean (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Boolean value[])`
- [FMI2_Export](#) `fmi2Status fmi2GetString (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2String value[])`
- [FMI2_Export](#) `fmi2Status fmi2SetReal (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Real value[])`
- [FMI2_Export](#) `fmi2Status fmi2SetInteger (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer value[])`
- [FMI2_Export](#) `fmi2Status fmi2SetBoolean (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Boolean value[])`
- [FMI2_Export](#) `fmi2Status fmi2SetString (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2String value[])`
- [FMI2_Export](#) `const char * fmi2GetTypesPlatform ()`
- [FMI2_Export](#) `fmi2Status fmi2Terminate (fmi2Component c)`
- [FMI2_Export](#) `fmi2Status fmi2Reset (fmi2Component c)`
- [FMI2_Export](#) `fmi2Status fmi2SetRealInputDerivatives (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer order[], const fmi2Real value[])`
- [FMI2_Export](#) `fmi2Status fmi2GetRealOutputDerivatives (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer order[], fmi2Real value[])`
- [FMI2_Export](#) `fmi2Status fmi2CancelStep (fmi2Component c)`

- **FMI2_Export** fmi2Status **fmi2DoStep** (fmi2Component c, fmi2Real currentCommunicationPoint, fmi2Real communicationStepSize, fmi2Boolean newStep)
- **FMI2_Export** fmi2Status **fmi2GetStatus** (fmi2Component c, const fmi2StatusKind s, fmi2Status *value)
- **FMI2_Export** fmi2Status **fmi2GetRealStatus** (fmi2Component c, const fmi2StatusKind s, fmi2Real *value)
- **FMI2_Export** fmi2Status **fmi2GetIntegerStatus** (fmi2Component c, const fmi2StatusKind s, fmi2Integer *value)
- **FMI2_Export** fmi2Status **fmi2GetBooleanStatus** (fmi2Component c, const fmi2StatusKind s, fmi2Boolean *value)
- **FMI2_Export** fmi2Status **fmi2GetStringStatus** (fmi2Component c, const fmi2StatusKind s, fmi2String *value)

9.68.1 Function Documentation

9.68.1.1 **FMI2_Export** const char* **fmi2GetVersion** ()

Definition at line 38 of file fmu2_model_cs.c.

9.68.1.2 **FMI2_Export** fmi2Status **fmi2SetDebugLogging** (fmi2Component c, fmi2Boolean *loggingOn*, size_t n, const fmi2String *cat[]*)

Definition at line 43 of file fmu2_model_cs.c.

9.68.1.3 **FMI2_Export** fmi2Component **fmi2Instantiate** (fmi2String *instanceName*, fmi2Type *fmuType*, fmi2String *GUID*, fmi2String *location*, const fmi2CallbackFunctions * *functions*, fmi2Boolean *visible*, fmi2Boolean *loggingOn*)

Definition at line 48 of file fmu2_model_cs.c.

9.68.1.4 **FMI2_Export** void **fmi2FreeInstance** (fmi2Component c)

Definition at line 57 of file fmu2_model_cs.c.

9.68.1.5 **FMI2_Export** fmi2Status **fmi2SetupExperiment** (fmi2Component c, fmi2Boolean *toleranceDefined*, fmi2Real *tolerance*, fmi2Real *startTime*, fmi2Boolean *stopTimeDefined*, fmi2Real *stopTime*)

Definition at line 62 of file fmu2_model_cs.c.

9.68.1.6 **FMI2_Export** fmi2Status **fmi2EnterInitializationMode** (fmi2Component c)

Definition at line 71 of file fmu2_model_cs.c.

9.68.1.7 FMI2_Export fmi2Status fmi2ExitInitializationMode (fmi2Component c)

Definition at line 76 of file fmu2_model_cs.c.

9.68.1.8 FMI2_Export fmi2Status fmi2GetReal (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Real value[])

Definition at line 81 of file fmu2_model_cs.c.

9.68.1.9 FMI2_Export fmi2Status fmi2GetInteger (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Integer value[])

Definition at line 86 of file fmu2_model_cs.c.

9.68.1.10 FMI2_Export fmi2Status fmi2GetBoolean (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Boolean value[])

Definition at line 91 of file fmu2_model_cs.c.

9.68.1.11 FMI2_Export fmi2Status fmi2GetString (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2String value[])

Definition at line 96 of file fmu2_model_cs.c.

9.68.1.12 FMI2_Export fmi2Status fmi2SetReal (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Real value[])

Definition at line 101 of file fmu2_model_cs.c.

9.68.1.13 FMI2_Export fmi2Status fmi2SetInteger (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer value[])

Definition at line 106 of file fmu2_model_cs.c.

9.68.1.14 FMI2_Export fmi2Status fmi2SetBoolean (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Boolean value[])

Definition at line 111 of file fmu2_model_cs.c.

9.68.1.15 FMI2_Export fmi2Status fmi2SetString (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2String value[])

Definition at line 116 of file fmu2_model_cs.c.

9.68.1.16 FMI2_Export `const char* fmi2GetTypesPlatform ( )`

Definition at line 122 of file `fmu2_model_cs.c`.

9.68.1.17 FMI2_Export `fmi2Status fmi2Terminate ( fmi2Component c )`

Definition at line 127 of file `fmu2_model_cs.c`.

9.68.1.18 FMI2_Export `fmi2Status fmi2Reset ( fmi2Component c )`

Definition at line 132 of file `fmu2_model_cs.c`.

9.68.1.19 FMI2_Export `fmi2Status fmi2SetRealInputDerivatives ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer order[], const fmi2Real value[] )`

Definition at line 137 of file `fmu2_model_cs.c`.

9.68.1.20 FMI2_Export `fmi2Status fmi2GetRealOutputDerivatives ( fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer order[], fmi2Real value[] )`

Definition at line 142 of file `fmu2_model_cs.c`.

9.68.1.21 FMI2_Export `fmi2Status fmi2CancelStep ( fmi2Component c )`

Definition at line 147 of file `fmu2_model_cs.c`.

9.68.1.22 FMI2_Export `fmi2Status fmi2DoStep ( fmi2Component c, fmi2Real currentCommunicationPoint, fmi2Real communicationStepSize, fmi2Boolean newStep )`

Definition at line 152 of file `fmu2_model_cs.c`.

9.68.1.23 FMI2_Export `fmi2Status fmi2GetStatus ( fmi2Component c, const fmi2StatusKind s, fmi2Status * value )`

Definition at line 157 of file `fmu2_model_cs.c`.

9.68.1.24 FMI2_Export `fmi2Status fmi2GetRealStatus ( fmi2Component c, const fmi2StatusKind s, fmi2Real * value )`

Definition at line 162 of file `fmu2_model_cs.c`.

9.68.1.25 **FMI2_Export** fmi2Status fmi2GetIntegerStatus (fmi2Component *c*, const fmi2StatusKind *s*, fmi2Integer * *value*)

Definition at line 167 of file fmu2_model_cs.c.

9.68.1.26 **FMI2_Export** fmi2Status fmi2GetBooleanStatus (fmi2Component *c*, const fmi2StatusKind *s*, fmi2Boolean * *value*)

Definition at line 172 of file fmu2_model_cs.c.

9.68.1.27 **FMI2_Export** fmi2Status fmi2GetStringStatus (fmi2Component *c*, const fmi2StatusKind *s*, fmi2String * *value*)

Definition at line 177 of file fmu2_model_cs.c.

9.69 Test/FMI2/fmu_dummy/fmu2_model_defines.h File Reference

Defines

- #define [BUFFER](#) 1024
- #define [MAGIC_TEST_VALUE](#) 13.0 /* A test value for some functions */
- #define [VAR_R_HIGHT](#) 0
- #define [VAR_R_HIGHT_SPEED](#) 1
- #define [VAR_R_GRATIVY](#) 2
- #define [VAR_R_BOUNCE_CONF](#) 3
- #define [EVENT_HIGHT](#) 0
- #define [VAR_S_LOGGER_TEST](#) 0
- #define [N_STATES](#) 2
- #define [N_EVENT_INDICATORS](#) 1
- #define [N_REAL](#) 4
- #define [N_INTEGER](#) 4
- #define [N_BOOLEAN](#) 4
- #define [N_STRING](#) 4
- #define [N_INPUT_REAL](#) 2 /* CS only */
- #define [N_INPUT_REAL_MAX_ORDER](#) 2 /* CS only */
- #define [N_OUTPUT_REAL](#) 2 /* CS only */
- #define [N_OUTPUT_REAL_MAX_ORDER](#) 2 /* CS only */
- #define [FMI_VERSION](#) "2.0"
- #define [FMI_GUID](#) "123"

9.69.1 Define Documentation

9.69.1.1 **#define** [BUFFER](#) 1024

Definition at line 25 of file fmu2_model_defines.h.

9.69.1.2 **#define MAGIC_TEST_VALUE 13.0** /* A test value for some functions */

Definition at line 26 of file fmu2_model_defines.h.

9.69.1.3 **#define VAR_R_HIGHT 0**

Definition at line 31 of file fmu2_model_defines.h.

9.69.1.4 **#define VAR_R_HIGHT_SPEED 1**

Definition at line 32 of file fmu2_model_defines.h.

9.69.1.5 **#define VAR_R_GRATIVY 2**

Definition at line 34 of file fmu2_model_defines.h.

9.69.1.6 **#define VAR_R_BOUNCE_CONF 3**

Definition at line 35 of file fmu2_model_defines.h.

9.69.1.7 **#define EVENT_HIGHT 0**

Definition at line 38 of file fmu2_model_defines.h.

9.69.1.8 **#define VAR_S_LOGGER_TEST 0**

Definition at line 41 of file fmu2_model_defines.h.

9.69.1.9 **#define N_STATES 2**

Definition at line 44 of file fmu2_model_defines.h.

9.69.1.10 **#define N_EVENT_INDICATORS 1**

Definition at line 45 of file fmu2_model_defines.h.

9.69.1.11 **#define N_REAL 4**

Definition at line 46 of file fmu2_model_defines.h.

9.69.1.12 #define N_INTEGER 4

Definition at line 47 of file fmu2_model_defines.h.

9.69.1.13 #define N_BOOLEAN 4

Definition at line 48 of file fmu2_model_defines.h.

9.69.1.14 #define N_STRING 4

Definition at line 49 of file fmu2_model_defines.h.

9.69.1.15 #define N_INPUT_REAL 2 /* CS only */

Definition at line 51 of file fmu2_model_defines.h.

9.69.1.16 #define N_INPUT_REAL_MAX_ORDER 2 /* CS only */

Definition at line 52 of file fmu2_model_defines.h.

9.69.1.17 #define N_OUTPUT_REAL 2 /* CS only */

Definition at line 53 of file fmu2_model_defines.h.

9.69.1.18 #define N_OUTPUT_REAL_MAX_ORDER 2 /* CS only */

Definition at line 54 of file fmu2_model_defines.h.

9.69.1.19 #define FMI_VERSION "2.0"

Definition at line 57 of file fmu2_model_defines.h.

9.69.1.20 #define FMI_GUID "123"

Definition at line 59 of file fmu2_model_defines.h.

9.70 Test/FMI2/fmu_dummy/fmu2_model_me.c File Reference

```
#include <string.h> #include <FMI2/fmi2TypesPlatform.h>
#include <FMI2/fmi2Functions.h> #include <fmu_dummy/fmu2-
_model.h> #include "config_test.h" #include "fmu2_model.-
c"
```

Functions

- `FMI2_Export` `const char * fmi2GetVersion ()`
- `FMI2_Export` `fmi2Status fmi2SetDebugLogging (fmi2Component c, fmi2Boolean loggingOn, size_t n, const fmi2String cat[])`
- `FMI2_Export` `fmi2Component fmi2Instantiate (fmi2String instanceName, fmi2Type fmuType, fmi2String GUID, fmi2String location, const fmi2CallbackFunctions *functions, fmi2Boolean visible, fmi2Boolean loggingOn)`
- `FMI2_Export` `void fmi2FreeInstance (fmi2Component c)`
- `FMI2_Export` `fmi2Status fmi2SetupExperiment (fmi2Component c, fmi2Boolean toleranceDefined, fmi2Real tolerance, fmi2Real startTime, fmi2Boolean stopTimeDefined, fmi2Real stopTime)`
- `FMI2_Export` `fmi2Status fmi2EnterInitializationMode (fmi2Component c)`
- `FMI2_Export` `fmi2Status fmi2ExitInitializationMode (fmi2Component c)`
- `FMI2_Export` `fmi2Status fmi2GetReal (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Real value[])`
- `FMI2_Export` `fmi2Status fmi2GetInteger (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Integer value[])`
- `FMI2_Export` `fmi2Status fmi2GetBoolean (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2Boolean value[])`
- `FMI2_Export` `fmi2Status fmi2GetString (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, fmi2String value[])`
- `FMI2_Export` `fmi2Status fmi2SetReal (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Real value[])`
- `FMI2_Export` `fmi2Status fmi2SetInteger (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Integer value[])`
- `FMI2_Export` `fmi2Status fmi2SetBoolean (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2Boolean value[])`
- `FMI2_Export` `fmi2Status fmi2SetString (fmi2Component c, const fmi2ValueReference vr[], size_t nvr, const fmi2String value[])`
- `FMI2_Export` `const char * fmi2GetTypesPlatform ()`
- `FMI2_Export` `fmi2Status fmi2EnterEventMode (fmi2Component c)`
- `FMI2_Export` `fmi2Status fmi2NewDiscreteStates (fmi2Component c, fmi2EventInfo *eventInfo)`
- `FMI2_Export` `fmi2Status fmi2EnterContinuousTimeMode (fmi2Component c)`
- `FMI2_Export` `fmi2Status fmi2SetTime (fmi2Component c, fmi2Real fmitime)`
- `FMI2_Export` `fmi2Status fmi2SetContinuousStates (fmi2Component c, const fmi2Real x[], size_t nx)`
- `FMI2_Export` `fmi2Status fmi2CompletedIntegratorStep (fmi2Component c, fmi2Boolean noSetFMUStatePriorToCurrentPoint, fmi2Boolean *enterEventMode, fmi2Boolean *terminateSimulation)`
- `FMI2_Export` `fmi2Status fmi2GetDerivatives (fmi2Component c, fmi2Real derivatives[], size_t nx)`
- `FMI2_Export` `fmi2Status fmi2GetEventIndicators (fmi2Component c, fmi2Real eventIndicators[], size_t ni)`
- `FMI2_Export` `fmi2Status fmi2GetContinuousStates (fmi2Component c, fmi2Real states[], size_t nx)`

- `FMI2_Export fmi2Status fmi2GetNominalsOfContinuousStates (fmi2Component c, fmi2Real x_nominal[], size_t nx)`
- `FMI2_Export fmi2Status fmi2Terminate (fmi2Component c)`
- `FMI2_Export fmi2Status fmi2Reset (fmi2Component c)`

9.70.1 Function Documentation

9.70.1.1 `FMI2_Export const char* fmi2GetVersion ( )`

Definition at line 35 of file `fmu2_model_me.c`.

9.70.1.2 `FMI2_Export fmi2Status fmi2SetDebugLogging ( fmi2Component c, fmi2Boolean loggingOn, size_t n, const fmi2String cat[] )`

Definition at line 40 of file `fmu2_model_me.c`.

9.70.1.3 `FMI2_Export fmi2Component fmi2Instantiate ( fmi2String instanceName, fmi2Type fmuType, fmi2String GUID, fmi2String location, const fmi2CallbackFunctions * functions, fmi2Boolean visible, fmi2Boolean loggingOn )`

Definition at line 45 of file `fmu2_model_me.c`.

9.70.1.4 `FMI2_Export void fmi2FreeInstance ( fmi2Component c )`

Definition at line 54 of file `fmu2_model_me.c`.

9.70.1.5 `FMI2_Export fmi2Status fmi2SetupExperiment ( fmi2Component c, fmi2Boolean toleranceDefined, fmi2Real tolerance, fmi2Real startTime, fmi2Boolean stopTimeDefined, fmi2Real stopTime )`

Definition at line 59 of file `fmu2_model_me.c`.

9.70.1.6 `FMI2_Export fmi2Status fmi2EnterInitializationMode ( fmi2Component c )`

Definition at line 68 of file `fmu2_model_me.c`.

9.70.1.7 `FMI2_Export fmi2Status fmi2ExitInitializationMode ( fmi2Component c )`

Definition at line 73 of file `fmu2_model_me.c`.

9.70.1.8 FMI2_Export fmi2Status fmi2GetReal (fmi2Component *c*, const fmi2ValueReference *vr*[], size_t *nvr*, fmi2Real *value*[])

Definition at line 78 of file fmu2_model_me.c.

9.70.1.9 FMI2_Export fmi2Status fmi2GetInteger (fmi2Component *c*, const fmi2ValueReference *vr*[], size_t *nvr*, fmi2Integer *value*[])

Definition at line 83 of file fmu2_model_me.c.

9.70.1.10 FMI2_Export fmi2Status fmi2GetBoolean (fmi2Component *c*, const fmi2ValueReference *vr*[], size_t *nvr*, fmi2Boolean *value*[])

Definition at line 88 of file fmu2_model_me.c.

9.70.1.11 FMI2_Export fmi2Status fmi2GetString (fmi2Component *c*, const fmi2ValueReference *vr*[], size_t *nvr*, fmi2String *value*[])

Definition at line 93 of file fmu2_model_me.c.

9.70.1.12 FMI2_Export fmi2Status fmi2SetReal (fmi2Component *c*, const fmi2ValueReference *vr*[], size_t *nvr*, const fmi2Real *value*[])

Definition at line 98 of file fmu2_model_me.c.

9.70.1.13 FMI2_Export fmi2Status fmi2SetInteger (fmi2Component *c*, const fmi2ValueReference *vr*[], size_t *nvr*, const fmi2Integer *value*[])

Definition at line 103 of file fmu2_model_me.c.

9.70.1.14 FMI2_Export fmi2Status fmi2SetBoolean (fmi2Component *c*, const fmi2ValueReference *vr*[], size_t *nvr*, const fmi2Boolean *value*[])

Definition at line 108 of file fmu2_model_me.c.

9.70.1.15 FMI2_Export fmi2Status fmi2SetString (fmi2Component *c*, const fmi2ValueReference *vr*[], size_t *nvr*, const fmi2String *value*[])

Definition at line 113 of file fmu2_model_me.c.

9.70.1.16 FMI2_Export const char* fmi2GetTypesPlatform ()

Definition at line 119 of file fmu2_model_me.c.

9.70.1.17 FMI2_Export fmi2Status fmi2EnterEventMode (fmi2Component c)

Definition at line 124 of file fmu2_model_me.c.

9.70.1.18 FMI2_Export fmi2Status fmi2NewDiscreteStates (fmi2Component c, fmi2EventInfo * eventInfo)

Definition at line 129 of file fmu2_model_me.c.

9.70.1.19 FMI2_Export fmi2Status fmi2EnterContinuousTimeMode (fmi2Component c)

Definition at line 134 of file fmu2_model_me.c.

9.70.1.20 FMI2_Export fmi2Status fmi2SetTime (fmi2Component c, fmi2Real fmitime)

Definition at line 139 of file fmu2_model_me.c.

9.70.1.21 FMI2_Export fmi2Status fmi2SetContinuousStates (fmi2Component c, const fmi2Real x[], size_t nx)

Definition at line 144 of file fmu2_model_me.c.

9.70.1.22 FMI2_Export fmi2Status fmi2CompletedIntegratorStep (fmi2Component c, fmi2Boolean noSetFMUStatePriorToCurrentPoint, fmi2Boolean * enterEventMode, fmi2Boolean * terminateSimulation)

Definition at line 149 of file fmu2_model_me.c.

9.70.1.23 FMI2_Export fmi2Status fmi2GetDerivatives (fmi2Component c, fmi2Real derivatives[], size_t nx)

Definition at line 157 of file fmu2_model_me.c.

9.70.1.24 FMI2_Export fmi2Status fmi2GetEventIndicators (fmi2Component c, fmi2Real eventIndicators[], size_t ni)

Definition at line 162 of file fmu2_model_me.c.

9.70.1.25 FMI2_Export fmi2Status fmi2GetContinuousStates (fmi2Component c, fmi2Real states[], size_t nx)

Definition at line 167 of file fmu2_model_me.c.

9.70.1.26 **FMI2_Export** fmi2Status fmi2GetNominalsOfContinuousStates (fmi2Component *c*, fmi2Real *x_nominal*[], size_t *nx*)

Definition at line 172 of file fmu2_model_me.c.

9.70.1.27 **FMI2_Export** fmi2Status fmi2Terminate (fmi2Component *c*)

Definition at line 177 of file fmu2_model_me.c.

9.70.1.28 **FMI2_Export** fmi2Status fmi2Reset (fmi2Component *c*)

Definition at line 182 of file fmu2_model_me.c.

9.71 Test/fmi_import_test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <stdarg.-
h> #include <config_test.h> #include <fmilib.h>
```

Defines

- #define **BUFFER** 1000

Functions

- int **fmi1_test** (fmi_import_context_t *context, const char *dirPath)
- int **fmi2_test** (fmi_import_context_t *context, const char *dirPath)
- void **importlogger** (jm_callbacks *c, jm_string module, jm_log_level_enu_t log_level, jm_string message)
- void **do_exit** (int code)
- int **main** (int argc, char *argv[])

9.71.1 Define Documentation

9.71.1.1 #define **BUFFER** 1000

Definition at line 23 of file fmi_import_test.c.

9.71.2 Function Documentation

9.71.2.1 int **fmi1_test** (fmi_import_context_t * *context*, const char * *dirPath*)

Definition at line 38 of file fmi1_import_test.c.

9.71.2.2 `int fmi2_test ( fmi_import_context_t * context, const char * dirPath )`

Definition at line 41 of file `fmi2_import_test.c`.

9.71.2.3 `void importlogger (jm_callbacks * c, jm_string module,
jm_log_level_enu_t log_level, jm_string message)`

Definition at line 28 of file `fmi_import_test.c`.

9.71.2.4 `void do_exit ( int code )`

Definition at line 33 of file `fmi_import_test.c`.

9.71.2.5 `int main ( int argc, char * argv[] )`

Definition at line 40 of file `fmi_import_test.c`.