

vk3d

Generated by Doxygen 1.9.4



<b>1 Data Structure Index</b>	<b>1</b>
1.1 Data Structures	1
<b>2 File Index</b>	<b>5</b>
2.1 File List	5
<b>3 Data Structure Documentation</b>	<b>7</b>
3.1 vkd3d_application_info Struct Reference	7
3.1.1 Detailed Description	7
3.1.2 Field Documentation	8
3.1.2.1 api_version	8
3.1.2.2 application_name	8
3.1.2.3 engine_name	8
3.1.2.4 engine_version	8
3.2 vkd3d_device_create_info Struct Reference	9
3.2.1 Detailed Description	9
3.2.2 Field Documentation	10
3.2.2.1 adapter_luid	10
3.2.2.2 device_extensions	10
3.2.2.3 instance	10
3.2.2.4 instance_create_info	10
3.2.2.5 minimum_feature_level	10
3.2.2.6 parent	11
3.2.2.7 vk_physical_device	11
3.3 vkd3d_host_time_domain_info Struct Reference	11
3.3.1 Detailed Description	11
3.3.2 Field Documentation	12
3.3.2.1 ticks_per_second	12
3.4 vkd3d_image_resource_create_info Struct Reference	12
3.4.1 Detailed Description	12
3.4.2 Field Documentation	12
3.4.2.1 flags	13
3.4.2.2 present_state	13
3.5 vkd3d_instance_create_info Struct Reference	13
3.5.1 Detailed Description	14
3.5.2 Field Documentation	14
3.5.2.1 instance_extensions	14
3.5.2.2 pfn_create_thread	14
3.5.2.3 pfn_join_thread	15
3.5.2.4 pfn_vkGetInstanceProcAddr	15
3.5.2.5 wchar_size	15
3.6 vkd3d_optional_device_extensions_info Struct Reference	15
3.6.1 Detailed Description	16

3.6.2 Field Documentation	16
3.6.2.1 extensions	16
3.7 vkd3d_optional_instance_extensions_info Struct Reference	16
3.7.1 Detailed Description	17
3.7.2 Field Documentation	17
3.7.2.1 extensions	17
3.8 vkd3d_shader_code Struct Reference	17
3.8.1 Detailed Description	17
3.8.2 Field Documentation	17
3.8.2.1 code	18
3.9 vkd3d_shader_combined_resource_sampler Struct Reference	18
3.9.1 Detailed Description	19
3.9.2 Field Documentation	19
3.9.2.1 resource_space	19
3.9.2.2 sampler_space	19
3.10 vkd3d_shader_combined_resource_sampler_info Struct Reference	19
3.10.1 Detailed Description	20
3.11 vkd3d_shader_compile_info Struct Reference	20
3.11.1 Detailed Description	21
3.11.2 Field Documentation	21
3.11.2.1 next	21
3.11.2.2 options	21
3.11.2.3 source_name	21
3.12 vkd3d_shader_compile_option Struct Reference	21
3.12.1 Detailed Description	22
3.12.2 Field Documentation	22
3.12.2.1 value	22
3.13 vkd3d_shader_descriptor_binding Struct Reference	22
3.13.1 Detailed Description	23
3.13.2 Field Documentation	23
3.13.2.1 count	23
3.13.2.2 set	23
3.14 vkd3d_shader_descriptor_info Struct Reference	23
3.14.1 Detailed Description	24
3.14.2 Field Documentation	24
3.14.2.1 count	24
3.15 vkd3d_shader_descriptor_offset Struct Reference	24
3.16 vkd3d_shader_descriptor_offset_info Struct Reference	24
3.16.1 Detailed Description	25
3.16.2 Field Documentation	25
3.16.2.1 binding_offsets	25
3.16.2.2 descriptor_table_offset	26

3.16.2.3 uav_counter_offsets . . . . .	26
3.17 vkd3d_shader_descriptor_range Struct Reference . . . . .	26
3.18 vkd3d_shader_descriptor_range1 Struct Reference . . . . .	27
3.19 vkd3d_shader_dxbc_desc Struct Reference . . . . .	27
3.19.1 Detailed Description . . . . .	28
3.19.2 Field Documentation . . . . .	28
3.19.2.1 tag . . . . .	28
3.19.2.2 version . . . . .	28
3.20 vkd3d_shader_dxbc_section_desc Struct Reference . . . . .	29
3.20.1 Detailed Description . . . . .	29
3.21 vkd3d_shader_hlsl_source_info Struct Reference . . . . .	30
3.21.1 Detailed Description . . . . .	30
3.21.2 Field Documentation . . . . .	30
3.21.2.1 entry_point . . . . .	31
3.22 vkd3d_shader_interface_info Struct Reference . . . . .	31
3.22.1 Detailed Description . . . . .	32
3.23 vkd3d_shader_macro Struct Reference . . . . .	32
3.23.1 Detailed Description . . . . .	32
3.23.2 Field Documentation . . . . .	32
3.23.2.1 name . . . . .	32
3.23.2.2 value . . . . .	33
3.24 vkd3d_shader_parameter Struct Reference . . . . .	33
3.24.1 Detailed Description . . . . .	33
3.25 vkd3d_shader_parameter1 Struct Reference . . . . .	34
3.25.1 Detailed Description . . . . .	34
3.26 vkd3d_shader_parameter_buffer Struct Reference . . . . .	35
3.26.1 Detailed Description . . . . .	35
3.26.2 Field Documentation . . . . .	35
3.26.2.1 set . . . . .	35
3.27 vkd3d_shader_parameter_immediate_constant Struct Reference . . . . .	36
3.27.1 Detailed Description . . . . .	36
3.27.2 Field Documentation . . . . .	36
3.27.2.1 f32 . . . . .	36
3.28 vkd3d_shader_parameter_immediate_constant1 Struct Reference . . . . .	36
3.28.1 Detailed Description . . . . .	37
3.28.2 Field Documentation . . . . .	37
3.28.2.1 f32_vec4 . . . . .	37
3.29 vkd3d_shader_parameter_info Struct Reference . . . . .	38
3.29.1 Detailed Description . . . . .	38
3.30 vkd3d_shader_parameter_specialization_constant Struct Reference . . . . .	39
3.30.1 Detailed Description . . . . .	39
3.30.2 Field Documentation . . . . .	39

3.30.2.1 id . . . . .	39
3.31 vkd3d_shader_preprocess_info Struct Reference . . . . .	39
3.31.1 Detailed Description . . . . .	40
3.31.2 Field Documentation . . . . .	40
3.31.2.1 macros . . . . .	40
3.31.2.2 pfn_close_include . . . . .	41
3.31.2.3 pfn_open_include . . . . .	41
3.32 vkd3d_shader_push_constant_buffer Struct Reference . . . . .	41
3.32.1 Detailed Description . . . . .	42
3.32.2 Field Documentation . . . . .	42
3.32.2.1 register_space . . . . .	42
3.33 vkd3d_shader_resource_binding Struct Reference . . . . .	42
3.33.1 Detailed Description . . . . .	43
3.33.2 Field Documentation . . . . .	43
3.33.2.1 register_index . . . . .	43
3.33.2.2 register_space . . . . .	44
3.34 vkd3d_shader_root_constants Struct Reference . . . . .	44
3.35 vkd3d_shader_root_descriptor Struct Reference . . . . .	44
3.36 vkd3d_shader_root_descriptor1 Struct Reference . . . . .	44
3.37 vkd3d_shader_root_descriptor_table Struct Reference . . . . .	45
3.38 vkd3d_shader_root_descriptor_table1 Struct Reference . . . . .	45
3.39 vkd3d_shader_root_parameter Struct Reference . . . . .	46
3.40 vkd3d_shader_root_parameter1 Struct Reference . . . . .	46
3.41 vkd3d_shader_root_signature_desc Struct Reference . . . . .	47
3.42 vkd3d_shader_root_signature_desc1 Struct Reference . . . . .	47
3.43 vkd3d_shader_scan_combined_resource_sampler_info Struct Reference . . . . .	48
3.43.1 Detailed Description . . . . .	49
3.44 vkd3d_shader_scan_descriptor_info Struct Reference . . . . .	49
3.44.1 Detailed Description . . . . .	50
3.45 vkd3d_shader_scan_hull_shader_tessellation_info Struct Reference . . . . .	50
3.45.1 Detailed Description . . . . .	51
3.46 vkd3d_shader_scan_signature_info Struct Reference . . . . .	51
3.46.1 Detailed Description . . . . .	52
3.47 vkd3d_shader_signature Struct Reference . . . . .	53
3.47.1 Detailed Description . . . . .	53
3.48 vkd3d_shader_signature_element Struct Reference . . . . .	53
3.48.1 Detailed Description . . . . .	54
3.48.2 Field Documentation . . . . .	54
3.48.2.1 stream_index . . . . .	54
3.48.2.2 sysval_semantic . . . . .	54
3.48.2.3 used_mask . . . . .	55
3.49 vkd3d_shader_spirv_domain_shader_target_info Struct Reference . . . . .	55

3.50 vkd3d_shader_spirv_target_info Struct Reference . . . . .	55
3.51 vkd3d_shader_static_sampler_desc Struct Reference . . . . .	56
3.52 vkd3d_shader_transform_feedback_element Struct Reference . . . . .	56
3.53 vkd3d_shader_transform_feedback_info Struct Reference . . . . .	57
3.54 vkd3d_shader_uav_counter_binding Struct Reference . . . . .	57
3.54.1 Detailed Description . . . . .	58
3.54.2 Field Documentation . . . . .	58
3.54.2.1 register_space . . . . .	58
3.55 vkd3d_shader_varying_map Struct Reference . . . . .	59
3.55.1 Detailed Description . . . . .	59
3.55.2 Field Documentation . . . . .	59
3.55.2.1 output_signature_index . . . . .	59
3.56 vkd3d_shader_varying_map_info Struct Reference . . . . .	60
3.56.1 Detailed Description . . . . .	60
3.56.2 Field Documentation . . . . .	61
3.56.2.1 varying_map . . . . .	61
3.57 vkd3d_shader_versioned_root_signature_desc Struct Reference . . . . .	61
<b>4 File Documentation</b> . . . . .	<b>63</b>
4.1 /builds/nsivov/vkd3d/include/vkd3d.h File Reference . . . . .	63
4.1.1 Detailed Description . . . . .	66
4.1.2 Macro Definition Documentation . . . . .	66
4.1.2.1 VKD3D_RESOURCE_INITIAL_STATE_TRANSITION . . . . .	66
4.1.3 Typedef Documentation . . . . .	66
4.1.3.1 PFN_vkd3d_queue_signal_on_cpu . . . . .	67
4.1.3.2 PFN_vkd3d_set_log_callback . . . . .	67
4.1.4 Enumeration Type Documentation . . . . .	67
4.1.4.1 vkd3d_structure_type . . . . .	67
4.1.5 Function Documentation . . . . .	68
4.1.5.1 vkd3d_acquire_vk_queue() . . . . .	68
4.1.5.2 vkd3d_queue_signal_on_cpu() . . . . .	69
4.1.5.3 vkd3d_release_vk_queue() . . . . .	69
4.1.5.4 vkd3d_set_log_callback() . . . . .	69
4.2 vkd3d.h . . . . .	70
4.3 /builds/nsivov/vkd3d/include/vkd3d_shader.h File Reference . . . . .	73
4.3.1 Detailed Description . . . . .	84
4.3.2 Macro Definition Documentation . . . . .	84
4.3.2.1 VKD3D_SHADER_SWIZZLE . . . . .	84
4.3.2.2 VKD3D_SHADER_SWIZZLE_MASK . . . . .	85
4.3.3 Typedef Documentation . . . . .	85
4.3.3.1 PFN_vkd3d_shader_build_varying_map . . . . .	85
4.3.3.2 PFN_vkd3d_shader_close_include . . . . .	85

4.3.3.3 PFN_vk3d_shader_free_dxbc . . . . .	85
4.3.3.4 PFN_vk3d_shader_free_scan_combined_resource_sampler_info . . . . .	86
4.3.3.5 PFN_vk3d_shader_free_scan_signature_info . . . . .	86
4.3.3.6 PFN_vk3d_shader_open_include . . . . .	86
4.3.3.7 PFN_vk3d_shader_parse_dxbc . . . . .	87
4.3.3.8 PFN_vk3d_shader_preprocess . . . . .	87
4.3.3.9 PFN_vk3d_shader_serialize_dxbc . . . . .	87
4.3.3.10 PFN_vk3d_shader_set_log_callback . . . . .	88
4.3.4 Enumeration Type Documentation . . . . .	88
4.3.4.1 vk3d_shader_api_version . . . . .	88
4.3.4.2 vk3d_shader_compile_option_backward_compatibility . . . . .	88
4.3.4.3 vk3d_shader_compile_option_buffer_uav . . . . .	89
4.3.4.4 vk3d_shader_compile_option_feature_flags . . . . .	89
4.3.4.5 vk3d_shader_compile_option_formatting_flags . . . . .	90
4.3.4.6 vk3d_shader_compile_option_fragment_coordinate_origin . . . . .	90
4.3.4.7 vk3d_shader_compile_option_name . . . . .	91
4.3.4.8 vk3d_shader_compile_option_pack_matrix_order . . . . .	93
4.3.4.9 vk3d_shader_compile_option_typed_uav . . . . .	93
4.3.4.10 vk3d_shader_component_type . . . . .	94
4.3.4.11 vk3d_shader_d3dbc_constant_register . . . . .	94
4.3.4.12 vk3d_shader_descriptor_info_flag . . . . .	95
4.3.4.13 vk3d_shader_descriptor_range_flags . . . . .	95
4.3.4.14 vk3d_shader_descriptor_type . . . . .	96
4.3.4.15 vk3d_shader_fog_fragment_mode . . . . .	96
4.3.4.16 vk3d_shader_fog_source . . . . .	97
4.3.4.17 vk3d_shader_log_level . . . . .	98
4.3.4.18 vk3d_shader_minimum_precision . . . . .	98
4.3.4.19 vk3d_shader_parameter_data_type . . . . .	99
4.3.4.20 vk3d_shader_parameter_name . . . . .	99
4.3.4.21 vk3d_shader_parameter_type . . . . .	105
4.3.4.22 vk3d_shader_parse_dxbc_flags . . . . .	106
4.3.4.23 vk3d_shader_resource_data_type . . . . .	106
4.3.4.24 vk3d_shader_resource_type . . . . .	107
4.3.4.25 vk3d_shader_source_type . . . . .	107
4.3.4.26 vk3d_shader_spirv_environment . . . . .	108
4.3.4.27 vk3d_shader_spirv_extension . . . . .	108
4.3.4.28 vk3d_shader_structure_type . . . . .	109
4.3.4.29 vk3d_shader_sysval_semantic . . . . .	110
4.3.4.30 vk3d_shader_target_type . . . . .	111
4.3.4.31 vk3d_shader_visibility . . . . .	112
4.3.5 Function Documentation . . . . .	113
4.3.5.1 vk3d_shader_build_varying_map() . . . . .	113

4.3.5.2 vkd3d_shader_compile()	113
4.3.5.3 vkd3d_shader_convert_root_signature()	115
4.3.5.4 vkd3d_shader_find_signature_element()	115
4.3.5.5 vkd3d_shader_free_dxbc()	116
4.3.5.6 vkd3d_shader_free_messages()	116
4.3.5.7 vkd3d_shader_free_root_signature()	116
4.3.5.8 vkd3d_shader_free_scan_combined_resource_sampler_info()	117
4.3.5.9 vkd3d_shader_free_scan_descriptor_info()	117
4.3.5.10 vkd3d_shader_free_scan_signature_info()	117
4.3.5.11 vkd3d_shader_free_shader_code()	118
4.3.5.12 vkd3d_shader_free_shader_signature()	118
4.3.5.13 vkd3d_shader_get_supported_source_types()	118
4.3.5.14 vkd3d_shader_get_supported_target_types()	119
4.3.5.15 vkd3d_shader_get_version()	119
4.3.5.16 vkd3d_shader_parse_dxbc()	120
4.3.5.17 vkd3d_shader_parse_input_signature()	120
4.3.5.18 vkd3d_shader_parse_root_signature()	122
4.3.5.19 vkd3d_shader_preprocess()	123
4.3.5.20 vkd3d_shader_scan()	123
4.3.5.21 vkd3d_shader_serialize_dxbc()	124
4.3.5.22 vkd3d_shader_serialize_root_signature()	125
4.3.5.23 vkd3d_shader_set_log_callback()	126
4.4 vkd3d_shader.h	126
4.5 /builds/nsivov/vkd3d/include/vkd3d_types.h File Reference	141
4.5.1 Detailed Description	142
4.5.2 Enumeration Type Documentation	142
4.5.2.1 vkd3d_result	142
4.6 vkd3d_types.h	143
4.7 /builds/nsivov/vkd3d/include/vkd3d_utils.h File Reference	143
4.7.1 Detailed Description	145
4.7.2 Function Documentation	145
4.7.2.1 D3DCompile2()	146
4.7.2.2 D3DCompile2VKD3D()	146
4.7.2.3 D3DDisassemble()	147
4.7.2.4 D3DGetBlobPart()	147
4.7.2.5 D3DGetDebugInfo()	147
4.7.2.6 D3DGetInputAndOutputSignatureBlob()	148
4.7.2.7 D3DGetInputSignatureBlob()	148
4.7.2.8 D3DGetOutputSignatureBlob()	148
4.7.2.9 D3DReflect()	148
4.7.2.10 D3DStripShader()	149
4.7.2.11 vkd3d_utils_set_log_callback()	149

<a href="#">4.8 vkd3d_utils.h</a> . . . . .	149
---	-----

# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">vkd3d_application_info</a>	
A chained structure to specify application information . . . . .	7
<a href="#">vkd3d_device_create_info</a>	
A chained structure containing device creation parameters . . . . .	9
<a href="#">vkd3d_host_time_domain_info</a>	
A chained structure to specify the host time domain . . . . .	11
<a href="#">vkd3d_image_resource_create_info</a>	
A chained structure containing the parameters to create a D3D12 resource backed by a Vulkan image . . . . .	12
<a href="#">vkd3d_instance_create_info</a>	
A chained structure containing instance creation parameters . . . . .	13
<a href="#">vkd3d_optional_device_extensions_info</a>	
A chained structure to specify optional device extensions . . . . .	15
<a href="#">vkd3d_optional_instance_extensions_info</a>	
A chained structure to specify optional instance extensions . . . . .	16
<a href="#">vkd3d_shader_code</a>	
A generic structure containing a GPU shader, in text or byte-code format . . . . .	17
<a href="#">vkd3d_shader_combined_resource_sampler</a>	
Describes the mapping of a Direct3D resource-sampler pair to a combined sampler (i.e . . . . .	18
<a href="#">vkd3d_shader_combined_resource_sampler_info</a>	
This structure describes a single resource-sampler pair . . . . .	19
<a href="#">vkd3d_shader_compile_info</a>	
A chained structure containing compilation parameters . . . . .	20
<a href="#">vkd3d_shader_compile_option</a>	
Various settings which may affect shader compilation or scanning, passed as part of struct <a href="#">vkd3d_shader_compile_info</a> . . . . .	21
<a href="#">vkd3d_shader_descriptor_binding</a>	
A common structure describing the bind point of a descriptor or descriptor array in the target environment . . . . .	22
<a href="#">vkd3d_shader_descriptor_info</a>	
Describes a single shader descriptor; returned as part of struct <a href="#">vkd3d_shader_scan_descriptor_info</a> . . . . .	23
<a href="#">vkd3d_shader_descriptor_offset</a> . . . . .	24
<a href="#">vkd3d_shader_descriptor_offset_info</a>	
A chained structure containing descriptor offsets . . . . .	24

<a href="#">vkd3d_shader_descriptor_range</a>	26
<a href="#">vkd3d_shader_descriptor_range1</a>	27
<a href="#">vkd3d_shader_dxbc_desc</a>	
A description of a DXBC blob, as returned by <a href="#">vkd3d_shader_parse_dxbc()</a>	27
<a href="#">vkd3d_shader_dxbc_section_desc</a>	
A description of a DXBC section	29
<a href="#">vkd3d_shader_hlsl_source_info</a>	
A chained structure containing HLSL compilation parameters	30
<a href="#">vkd3d_shader_interface_info</a>	
A chained structure describing the interface between a compiled shader and the target environment	31
<a href="#">vkd3d_shader_macro</a>	
A single preprocessor macro, passed as part of struct <a href="#">vkd3d_shader_preprocess_info</a>	32
<a href="#">vkd3d_shader_parameter</a>	
An individual shader parameter	33
<a href="#">vkd3d_shader_parameter1</a>	
An individual shader parameter	34
<a href="#">vkd3d_shader_parameter_buffer</a>	
The linkage of a parameter specified through a uniform buffer, used in struct <a href="#">vkd3d_shader_parameter1</a>	35
<a href="#">vkd3d_shader_parameter_immediate_constant</a>	
The value of an immediate constant parameter, used in struct <a href="#">vkd3d_shader_parameter</a>	36
<a href="#">vkd3d_shader_parameter_immediate_constant1</a>	
The value of an immediate constant parameter, used in struct <a href="#">vkd3d_shader_parameter1</a>	36
<a href="#">vkd3d_shader_parameter_info</a>	
Interface information regarding a builtin shader parameter	38
<a href="#">vkd3d_shader_parameter_specialization_constant</a>	
The linkage of a specialization constant parameter, used in struct <a href="#">vkd3d_shader_parameter</a> and struct <a href="#">vkd3d_shader_parameter1</a>	39
<a href="#">vkd3d_shader_preprocess_info</a>	
A chained structure containing preprocessing parameters	39
<a href="#">vkd3d_shader_push_constant_buffer</a>	
Describes the mapping of a Direct3D constant buffer to a range of push constants in the target environment	41
<a href="#">vkd3d_shader_resource_binding</a>	
Describes the mapping of a single resource or resource array to its binding point in the target environment	42
<a href="#">vkd3d_shader_root_constants</a>	44
<a href="#">vkd3d_shader_root_descriptor</a>	44
<a href="#">vkd3d_shader_root_descriptor1</a>	44
<a href="#">vkd3d_shader_root_descriptor_table</a>	45
<a href="#">vkd3d_shader_root_descriptor_table1</a>	45
<a href="#">vkd3d_shader_root_parameter</a>	46
<a href="#">vkd3d_shader_root_parameter1</a>	46
<a href="#">vkd3d_shader_root_signature_desc</a>	47
<a href="#">vkd3d_shader_root_signature_desc1</a>	47
<a href="#">vkd3d_shader_scan_combined_resource_sampler_info</a>	
A chained structure describing the resource-sampler pairs used by a shader	48
<a href="#">vkd3d_shader_scan_descriptor_info</a>	
A chained structure enumerating the descriptors declared by a shader	49
<a href="#">vkd3d_shader_scan_hull_shader_tessellation_info</a>	
A chained structure describing the tessellation information in a hull shader	50
<a href="#">vkd3d_shader_scan_signature_info</a>	
A chained structure containing descriptions of shader inputs and outputs	51
<a href="#">vkd3d_shader_signature</a>	
Description of a shader input or output signature	53
<a href="#">vkd3d_shader_signature_element</a>	
A single shader varying, returned as part of struct <a href="#">vkd3d_shader_signature</a>	53

<a href="#">vkd3d_shader_spirv_domain_shader_target_info</a>	55
<a href="#">vkd3d_shader_spirv_target_info</a>	55
<a href="#">vkd3d_shader_static_sampler_desc</a>	56
<a href="#">vkd3d_shader_transform_feedback_element</a>	56
<a href="#">vkd3d_shader_transform_feedback_info</a>	57
<a href="#">vkd3d_shader_uav_counter_binding</a>	
Describes the mapping of a single Direct3D UAV counter	57
<a href="#">vkd3d_shader_varying_map</a>	
Describes the mapping of a output varying register in a shader stage, to an input varying register in the following shader stage	59
<a href="#">vkd3d_shader_varying_map_info</a>	
A chained structure which describes how output varyings in this shader stage should be mapped to input varyings in the next stage	60
<a href="#">vkd3d_shader_versioned_root_signature_desc</a>	61



## Chapter 2

# File Index

### 2.1 File List

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

<a href="#">/builds/nsivov/vkd3d/include/vkd3d.h</a>	
This file contains definitions for the vkd3d library . . . . .	63
<a href="#">/builds/nsivov/vkd3d/include/vkd3d_shader.h</a>	
This file contains definitions for the vkd3d-shader library . . . . .	73
<a href="#">/builds/nsivov/vkd3d/include/vkd3d_types.h</a>	
This file contains definitions for basic types used by vkd3d libraries . . . . .	141
<a href="#">/builds/nsivov/vkd3d/include/vkd3d_utils.h</a>	
This file contains definitions for the vkd3d-utils library . . . . .	143



## Chapter 3

# Data Structure Documentation

### 3.1 vkd3d\_application\_info Struct Reference

A chained structure to specify application information.

```
#include <vkd3d.h>
```

#### Data Fields

- enum [vkd3d\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_STRUCTURE\_TYPE\_APPLICATION\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- const char \* [application\\_name](#)  
*The application's name, to be passed to the Vulkan implementation.*
- uint32\_t **application\_version**  
*The application's version, to be passed to the Vulkan implementation.*
- const char \* [engine\\_name](#)  
*The engine name, to be passed to the Vulkan implementation.*
- uint32\_t [engine\\_version](#)  
*The engine version, to be passed to the Vulkan implementation.*
- enum vkd3d\_api\_version [api\\_version](#)  
*The vkd3d API version to use, to guarantee backward compatibility of the shared library.*

#### 3.1.1 Detailed Description

A chained structure to specify application information.

This structure extends [vkd3d\\_instance\\_create\\_info](#).

Since

1.2

### 3.1.2 Field Documentation

#### 3.1.2.1 `api_version`

```
enum vkd3d_api_version vkd3d_application_info::api_version
```

The vkd3d API version to use, to guarantee backward compatibility of the shared library.

If this chained structure is not used then VKD3D\_API\_VERSION\_1\_0 is used.

#### 3.1.2.2 `application_name`

```
const char* vkd3d_application_info::application_name
```

The application's name, to be passed to the Vulkan implementation.

If it is NULL, a name is computed from the process executable filename. If that cannot be done, the empty string is used.

#### 3.1.2.3 `engine_name`

```
const char* vkd3d_application_info::engine_name
```

The engine name, to be passed to the Vulkan implementation.

If it is NULL, "vkd3d" is used.

#### 3.1.2.4 `engine_version`

```
uint32_t vkd3d_application_info::engine_version
```

The engine version, to be passed to the Vulkan implementation.

If it is 0, the version is computed from the vkd3d library version.

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

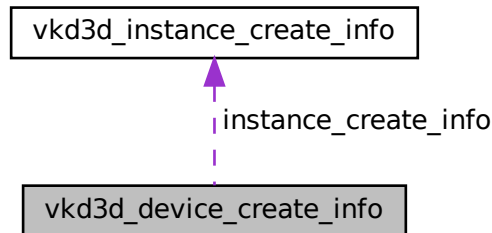
- [/builds/nsivov/vkd3d/include/vkd3d.h](#)

## 3.2 vkd3d\_device\_create\_info Struct Reference

A chained structure containing device creation parameters.

```
#include <vkd3d.h>
```

Collaboration diagram for vkd3d\_device\_create\_info:



### Data Fields

- enum [vkd3d\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_STRUCTURE\_TYPE\_DEVICE\_CREATE\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- D3D\_FEATURE\_LEVEL [minimum\\_feature\\_level](#)  
*The minimum feature level to request.*
- struct vkd3d\_instance \* [instance](#)  
*The vkd3d instance to use to create a device.*
- const struct [vkd3d\\_instance\\_create\\_info](#) \* [instance\\_create\\_info](#)  
*The parameters used to create an instance, which is then used to create a device.*
- VkPhysicalDevice [vk\\_physical\\_device](#)  
*The Vulkan physical device to use.*
- const char \*const \* [device\\_extensions](#)  
*A list of Vulkan device extensions to request.*
- uint32\_t **device\_extension\_count**  
*The number of elements in the device\_extensions array.*
- IUnknown \* [parent](#)  
*An object to be set as the device parent.*
- LUID [adapter\\_luid](#)  
*The adapter LUID to be set for the device.*

### 3.2.1 Detailed Description

A chained structure containing device creation parameters.

## 3.2.2 Field Documentation

### 3.2.2.1 adapter\_luid

```
LUID vkd3d_device_create_info::adapter_luid
```

The adapter LUID to be set for the device.

This is not used by vkd3d except for being returned by GetAdapterLuid.

### 3.2.2.2 device\_extensions

```
const char* const* vkd3d_device_create_info::device_extensions
```

A list of Vulkan device extensions to request.

They are intended as required, so device creation will fail if any of them is not available.

### 3.2.2.3 instance

```
struct vkd3d_instance* vkd3d_device_create_info::instance
```

The vkd3d instance to use to create a device.

Either this or instance\_create\_info must be set.

### 3.2.2.4 instance\_create\_info

```
const struct vkd3d_instance_create_info* vkd3d_device_create_info::instance_create_info
```

The parameters used to create an instance, which is then used to create a device.

Either this or instance must be set.

### 3.2.2.5 minimum\_feature\_level

```
D3D_FEATURE_LEVEL vkd3d_device_create_info::minimum_feature_level
```

The minimum feature level to request.

Device creation will fail with E\_INVALIDARG if the Vulkan device doesn't have the features needed to fulfill the request.

### 3.2.2.6 parent

```
Unknown* vkd3d_device_create_info::parent
```

An object to be set as the device parent.

This is not used by vkd3d except for being returned by vkd3d\_get\_device\_parent.

### 3.2.2.7 vk\_physical\_device

```
VkPhysicalDevice vkd3d_device_create_info::vk_physical_device
```

The Vulkan physical device to use.

If it is NULL, the first physical device found is used, prioritizing discrete GPUs over integrated GPUs and integrated GPUs over all the others.

This parameter can be overridden by setting environment variable VKD3D\_VULKAN\_DEVICE.

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

- [/builds/nsivov/vkd3d/include/vkd3d.h](#)

## 3.3 vkd3d\_host\_time\_domain\_info Struct Reference

A chained structure to specify the host time domain.

```
#include <vkd3d.h>
```

### Data Fields

- enum [vkd3d\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_STRUCTURE\_TYPE\_HOST\_TIME\_DOMAIN\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- uint64\_t [ticks\\_per\\_second](#)  
*The number of clock ticks per second, used for GetClockCalibration().*

### 3.3.1 Detailed Description

A chained structure to specify the host time domain.

This structure extends [vkd3d\\_instance\\_create\\_info](#).

Since

1.3

### 3.3.2 Field Documentation

#### 3.3.2.1 ticks\_per\_second

```
uint64_t vkd3d_host_time_domain_info::ticks_per_second
```

The number of clock ticks per second, used for GetClockCalibration().

It should normally match the expected result of QueryPerformanceFrequency(). If this chained structure is not used then 10 millions is used, which means that each tick is a tenth of microsecond, or equivalently 100 nanoseconds.

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

- [/builds/nsivov/vkd3d/include/vkd3d.h](#)

## 3.4 vkd3d\_image\_resource\_create\_info Struct Reference

A chained structure containing the parameters to create a D3D12 resource backed by a Vulkan image.

```
#include <vkd3d.h>
```

### Data Fields

- enum [vkd3d\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_STRUCTURE\_TYPE\_IMAGE\_RESOURCE\_CREATE\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- VkImage **vk\_image**  
*The Vulkan image that backs the resource.*
- D3D12\_RESOURCE\_DESC **desc**  
*The resource description.*
- unsigned int [flags](#)  
*A combination of zero or more flags.*
- D3D12\_RESOURCE\_STATES [present\\_state](#)  
*This field specifies how to handle resource state D3D12\_RESOURCE\_STATE\_PRESENT for the resource.*

#### 3.4.1 Detailed Description

A chained structure containing the parameters to create a D3D12 resource backed by a Vulkan image.

#### 3.4.2 Field Documentation

### 3.4.2.1 flags

```
unsigned int vkd3d_image_resource_create_info::flags
```

A combination of zero or more flags.

The valid flags are VKD3D\_RESOURCE\_INITIAL\_STATE\_TRANSITION and VKD3D\_RESOURCE\_PRESENT\_STATE\_TRANSITION.

### 3.4.2.2 present\_state

```
D3D12_RESOURCE_STATES vkd3d_image_resource_create_info::present_state
```

This field specifies how to handle resource state D3D12\_RESOURCE\_STATE\_PRESENT for the resource.

Notice that on D3D12 there is no difference between D3D12\_RESOURCE\_STATE\_COMMON and D3D12\_RESOURCE\_STATE\_PRESENT (they have the same value), while on Vulkan two different layouts are used (VK\_IMAGE\_LAYOUT\_GENERAL and VK\_IMAGE\_LAYOUT\_PRESENT\_SRC\_KHR).

- When flag VKD3D\_RESOURCE\_PRESENT\_STATE\_TRANSITION is not specified, field present\_state is ignored and resource state D3D12\_RESOURCE\_STATE\_COMMON/\_PRESENT is mapped to VK\_IMAGE\_LAYOUT\_GENERAL; this is useful for non-swapchain resources.
- Otherwise, when present\_state is D3D12\_RESOURCE\_STATE\_PRESENT/\_COMMON, resource state D3D12\_RESOURCE\_STATE\_COMMON/\_PRESENT is mapped to VK\_IMAGE\_LAYOUT\_PRESENT\_SRC\_KHR; this is useful for swapchain resources that are directly backed by a Vulkan swapchain image.
- Otherwise, resource state D3D12\_RESOURCE\_STATE\_COMMON/\_PRESENT is treated as resource state present\_state; this is useful for swapchain resources that backed by a Vulkan non-swapchain image, which the client will likely consume with a copy or drawing operation at presentation time.

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

- [/builds/nsivov/vkd3d/include/vkd3d.h](#)

## 3.5 vkd3d\_instance\_create\_info Struct Reference

A chained structure containing instance creation parameters.

```
#include <vkd3d.h>
```

## Data Fields

- enum [vkd3d\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_STRUCTURE\_TYPE\_INSTANCE\_CREATE\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- PFN\_vkd3d\_signal\_event **pfn\_signal\_event**  
*An pointer to a function to signal events.*
- PFN\_vkd3d\_create\_thread [pfn\\_create\\_thread](#)  
*An optional pointer to a function to create threads.*
- PFN\_vkd3d\_join\_thread [pfn\\_join\\_thread](#)  
*An optional pointer to a function to join threads.*
- size\_t [wchar\\_size](#)  
*The size of type WCHAR.*
- PFN\_vkGetInstanceProcAddr [pfn\\_vkGetInstanceProcAddr](#)  
*A pointer to the vkGetInstanceProcAddr Vulkan function, which will be used to load all the other Vulkan functions.*
- const char \*const \* [instance\\_extensions](#)  
*A list of Vulkan instance extensions to request.*
- uint32\_t **instance\_extension\_count**  
*The number of elements in the instance\_extensions array.*

### 3.5.1 Detailed Description

A chained structure containing instance creation parameters.

### 3.5.2 Field Documentation

#### 3.5.2.1 instance\_extensions

```
const char* const* vkd3d_instance_create_info::instance_extensions
```

A list of Vulkan instance extensions to request.

They are intended as required, so instance creation will fail if any of them is not available.

#### 3.5.2.2 pfn\_create\_thread

```
PFN_vkd3d_create_thread vkd3d_instance_create_info::pfn_create_thread
```

An optional pointer to a function to create threads.

If this is NULL vkd3d will use a function of its choice, depending on the platform. It must be NULL if and only if pfn\_join\_thread is NULL.

### 3.5.2.3 pfn\_join\_thread

```
PFN_vkJoinThread vkd3d_instance_create_info::pfn_join_thread
```

An optional pointer to a function to join threads.

If this is NULL vkd3d will use a function of its choice, depending on the platform. It must be NULL if and only if pfn\_create\_thread is NULL.

### 3.5.2.4 pfn\_vkGetInstanceProcAddr

```
PFN_vkGetInstanceProcAddr vkd3d_instance_create_info::pfn_vkGetInstanceProcAddr
```

A pointer to the vkGetInstanceProcAddr Vulkan function, which will be used to load all the other Vulkan functions.

If set to NULL, vkd3d will search and use the Vulkan loader.

### 3.5.2.5 wchar\_size

```
size_t vkd3d_instance_create_info::wchar_size
```

The size of type WCHAR.

It must be 2 or 4 and should normally be set to sizeof(WCHAR).

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

- [/builds/nsivov/vkd3d/include/vkd3d.h](#)

## 3.6 vkd3d\_optional\_device\_extensions\_info Struct Reference

A chained structure to specify optional device extensions.

```
#include <vkd3d.h>
```

### Data Fields

- enum [vkd3d\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_STRUCTURE\_TYPE\_OPTIONAL\_DEVICE\_EXTENSIONS\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- const char \*const \* [extensions](#)  
*A list of optional Vulkan device extensions to request.*
- uint32\_t **extension\_count**  
*The number of elements in the extensions array.*

### 3.6.1 Detailed Description

A chained structure to specify optional device extensions.

This structure extends [vkd3d\\_device\\_create\\_info](#).

Since

1.2

### 3.6.2 Field Documentation

#### 3.6.2.1 extensions

```
const char* const* vkd3d_optional_device_extensions_info::extensions
```

A list of optional Vulkan device extensions to request.

Device creation does not fail if they are not available.

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

- [/builds/nsivov/vkd3d/include/vkd3d.h](#)

## 3.7 vkd3d\_optional\_instance\_extensions\_info Struct Reference

A chained structure to specify optional instance extensions.

```
#include <vkd3d.h>
```

### Data Fields

- enum [vkd3d\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_STRUCTURE\_TYPE\_OPTIONAL\_INSTANCE\_EXTENSIONS\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- const char \*const \* [extensions](#)  
*A list of optional Vulkan instance extensions to request.*
- uint32\_t **extension\_count**  
*The number of elements in the extensions array.*

### 3.7.1 Detailed Description

A chained structure to specify optional instance extensions.

This structure extends [vkd3d\\_instance\\_create\\_info](#).

Since

1.1

### 3.7.2 Field Documentation

#### 3.7.2.1 extensions

```
const char* const* vkd3d_optional_instance_extensions_info::extensions
```

A list of optional Vulkan instance extensions to request.

Instance creation does not fail if they are not available.

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

- `/builds/nsivov/vkd3d/include/vkd3d.h`

## 3.8 vkd3d\_shader\_code Struct Reference

A generic structure containing a GPU shader, in text or byte-code format.

```
#include <vkd3d_shader.h>
```

### Data Fields

- `const void * code`  
*Pointer to the code.*
- `size_t size`  
*Size of code, in bytes.*

### 3.8.1 Detailed Description

A generic structure containing a GPU shader, in text or byte-code format.

### 3.8.2 Field Documentation

### 3.8.2.1 code

```
const void* vkd3d_shader_code::code
```

Pointer to the code.

Note that textual formats are not null-terminated. Therefore *size* should not include a null terminator, when this structure is passed as input to a vkD3d-shader function, and the allocated string will not include a null terminator when this structure is used as output.

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

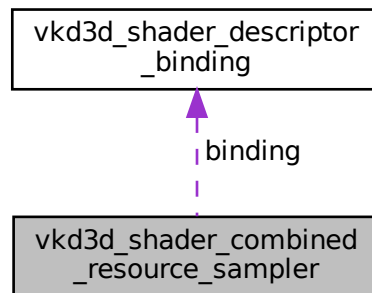
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.9 vkD3d\_shader\_combined\_resource\_sampler Struct Reference

Describes the mapping of a Direct3D resource-sampler pair to a combined sampler (i.e.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkD3d\_shader\_combined\_resource\_sampler:



### Data Fields

- unsigned int [resource\\_space](#)  
*Register space of the Direct3D resource.*
- unsigned int **resource\_index**  
*Register index of the Direct3D resource.*
- unsigned int [sampler\\_space](#)  
*Register space of the Direct3D sampler.*
- unsigned int **sampler\_index**  
*Register index of the Direct3D sampler.*
- enum [vkd3d\\_shader\\_visibility](#) **shader\_visibility**  
*Shader stage(s) to which the resource is visible.*
- unsigned int **flags**  
*A combination of zero or more elements of vkD3d\_shader\_binding\_flag.*
- struct [vkd3d\\_shader\\_descriptor\\_binding](#) **binding**  
*The binding in the target environment.*

### 3.9.1 Detailed Description

Describes the mapping of a Direct3D resource-sampler pair to a combined sampler (i.e. sampled image).

This structure is used in struct [vkd3d\\_shader\\_interface\\_info](#).

### 3.9.2 Field Documentation

#### 3.9.2.1 resource\_space

```
unsigned int vkd3d_shader_combined_resource_sampler::resource_space
```

Register space of the Direct3D resource.

If the source format does not support multiple register spaces, this parameter must be set to 0.

#### 3.9.2.2 sampler\_space

```
unsigned int vkd3d_shader_combined_resource_sampler::sampler_space
```

Register space of the Direct3D sampler.

If the source format does not support multiple register spaces, this parameter must be set to 0.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.10 vkd3d\_shader\_combined\_resource\_sampler\_info Struct Reference

This structure describes a single resource-sampler pair.

```
#include <vkd3d_shader.h>
```

### Data Fields

- unsigned int **resource\_space**
- unsigned int **resource\_index**
- unsigned int **sampler\_space**
- unsigned int **sampler\_index**

### 3.10.1 Detailed Description

This structure describes a single resource-sampler pair.

It is returned as part of struct [vkd3d\\_shader\\_scan\\_combined\\_resource\\_sampler\\_info](#).

Since

1.10

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

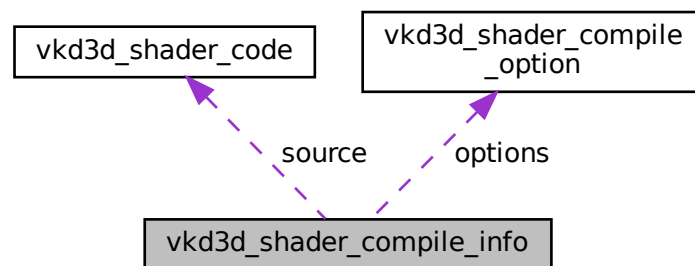
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.11 vkd3d\_shader\_compile\_info Struct Reference

A chained structure containing compilation parameters.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_compile\_info:



### Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_COMPILE\_INFO.*
- const void \* [next](#)  
*Optional pointer to a structure containing further parameters.*
- struct [vkd3d\\_shader\\_code](#) **source**  
*Input source code or byte code.*
- enum [vkd3d\\_shader\\_source\\_type](#) **source\_type**  
*Format of the input code passed in [source](#).*
- enum [vkd3d\\_shader\\_target\\_type](#) **target\_type**  
*Desired output format.*
- const struct [vkd3d\\_shader\\_compile\\_option](#) \* **options**  
*Pointer to an array of compilation options.*
- unsigned int **option\_count**  
*Size, in elements, of [options](#).*
- enum [vkd3d\\_shader\\_log\\_level](#) **log\_level**  
*Minimum severity of messages returned from the shader function.*
- const char \* [source\\_name](#)  
*Name of the initial source file, which may be used in error messages or debug information.*

### 3.11.1 Detailed Description

A chained structure containing compilation parameters.

### 3.11.2 Field Documentation

#### 3.11.2.1 next

```
const void* vkd3d_shader_compile_info::next
```

Optional pointer to a structure containing further parameters.

For a list of valid structures, refer to the respective function documentation. If no further parameters are needed, this field should be set to NULL.

#### 3.11.2.2 options

```
const struct vkd3d_shader_compile_option* vkd3d_shader_compile_info::options
```

Pointer to an array of compilation options.

This field is ignored if [option\\_count](#) is zero, but must be valid otherwise.

If the same option is specified multiple times, only the last value is used.

Options not relevant to or not supported by a particular shader compiler or scanner will be ignored.

#### 3.11.2.3 source\_name

```
const char* vkd3d_shader_compile_info::source_name
```

Name of the initial source file, which may be used in error messages or debug information.

This parameter is optional and may be NULL.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.12 vkd3d\_shader\_compile\_option Struct Reference

Various settings which may affect shader compilation or scanning, passed as part of struct [vkd3d\\_shader\\_compile\\_info](#).

```
#include <vkd3d_shader.h>
```

## Data Fields

- enum [vkd3d\\_shader\\_compile\\_option\\_name](#) **name**  
*Name of the option.*
- unsigned int [value](#)  
*A value associated with the option.*

### 3.12.1 Detailed Description

Various settings which may affect shader compilation or scanning, passed as part of struct [vkd3d\\_shader\\_compile\\_info](#).

For more details, see the documentation for individual options.

### 3.12.2 Field Documentation

#### 3.12.2.1 value

```
unsigned int vkd3d_shader_compile_option::value
```

A value associated with the option.

The type and interpretation of the value depends on the option in question.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.13 vkd3d\_shader\_descriptor\_binding Struct Reference

A common structure describing the bind point of a descriptor or descriptor array in the target environment.

```
#include <vkd3d_shader.h>
```

## Data Fields

- unsigned int [set](#)  
*The set of the descriptor.*
- unsigned int **binding**  
*The binding index of the descriptor.*
- unsigned int [count](#)  
*The size of this descriptor array.*

### 3.13.1 Detailed Description

A common structure describing the bind point of a descriptor or descriptor array in the target environment.

### 3.13.2 Field Documentation

#### 3.13.2.1 count

```
unsigned int vkd3d_shader_descriptor_binding::count
```

The size of this descriptor array.

If an offset is specified for this binding by the [vkd3d\\_shader\\_descriptor\\_offset\\_info](#) structure, counting starts at that offset.

#### 3.13.2.2 set

```
unsigned int vkd3d_shader_descriptor_binding::set
```

The set of the descriptor.

If the target environment does not support descriptor sets, this value must be set to 0.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.14 vkd3d\_shader\_descriptor\_info Struct Reference

Describes a single shader descriptor; returned as part of struct [vkd3d\\_shader\\_scan\\_descriptor\\_info](#).

```
#include <vkd3d_shader.h>
```

### Data Fields

- enum [vkd3d\\_shader\\_descriptor\\_type](#) **type**  
*Type of the descriptor (for example, SRV, CBV, UAV, or sampler).*
- unsigned int **register\_space**  
*Register space of the resource, or 0 if the shader does not support multiple register spaces.*
- unsigned int **register\_index**  
*Register index of the descriptor.*
- enum [vkd3d\\_shader\\_resource\\_type](#) **resource\_type**  
*Resource type, if applicable, including its dimension.*
- enum [vkd3d\\_shader\\_resource\\_data\\_type](#) **resource\_data\_type**  
*Data type contained in the resource (for example, float or integer).*
- unsigned int **flags**  
*Bitwise combination of zero or more members of [vkd3d\\_shader\\_descriptor\\_info\\_flag](#).*
- unsigned int [count](#)  
*Size of this descriptor array, or 1 if a single descriptor.*

### 3.14.1 Detailed Description

Describes a single shader descriptor; returned as part of struct [vkd3d\\_shader\\_scan\\_descriptor\\_info](#).

### 3.14.2 Field Documentation

#### 3.14.2.1 count

```
unsigned int vkd3d_shader_descriptor_info::count
```

Size of this descriptor array, or 1 if a single descriptor.

For an unbounded array this value is ~0u.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.15 vkd3d\_shader\_descriptor\_offset Struct Reference

### Data Fields

- unsigned int **static\_offset**
- unsigned int **dynamic\_offset\_index**

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

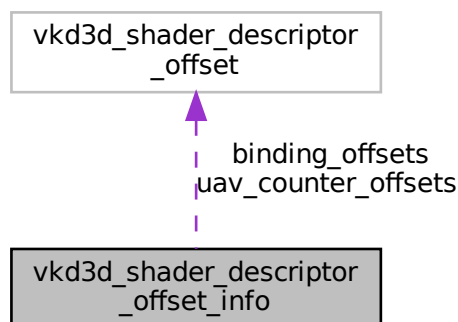
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.16 vkd3d\_shader\_descriptor\_offset\_info Struct Reference

A chained structure containing descriptor offsets.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_descriptor\_offset\_info:



## Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_DESCRIPTOR\_OFFSET\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- unsigned int [descriptor\\_table\\_offset](#)  
*Byte offset within the push constants of an array of 32-bit descriptor array offsets.*
- unsigned int **descriptor\_table\_count**  
*Size, in elements, of the descriptor table push constant array.*
- const struct [vkd3d\\_shader\\_descriptor\\_offset](#) \* [binding\\_offsets](#)  
*Pointer to an array of struct [vkd3d\\_shader\\_descriptor\\_offset](#) objects.*
- const struct [vkd3d\\_shader\\_descriptor\\_offset](#) \* [uav\\_counter\\_offsets](#)  
*Pointer to an array of offsets into the descriptor arrays referenced by the 'uav\_counters' array in struct [vkd3d\\_shader\\_interface\\_info](#).*

### 3.16.1 Detailed Description

A chained structure containing descriptor offsets.

This structure is optional.

This structure extends [vkd3d\\_shader\\_interface\\_info](#).

This structure contains only input parameters.

Since

1.3

### 3.16.2 Field Documentation

#### 3.16.2.1 [binding\\_offsets](#)

```
const struct vkd3d\_shader\_descriptor\_offset* vkd3d_shader_descriptor_offset_info::binding_↔
offsets
```

Pointer to an array of struct [vkd3d\\_shader\\_descriptor\\_offset](#) objects.

The 'static\_offset' field contains an offset into the descriptor arrays referenced by the 'bindings' array in struct [vkd3d\\_shader\\_interface\\_info](#). This allows mapping multiple shader resource arrays to a single binding point in the target environment.

'dynamic\_offset\_index' in struct [vkd3d\\_shader\\_descriptor\\_offset](#) allows offsets to be set at runtime. The 32-bit descriptor table push constant at this index will be added to 'static\_offset' to calculate the final binding offset.

If runtime offsets are not required, set all 'dynamic\_offset\_index' values to `~0u` and 'descriptor\_table\_count' to zero.

For example, to map Direct3D constant buffer registers 'cb0[0:3]' and 'cb1[6:7]' to descriptors 8-12 and 4-5 in the Vulkan descriptor array in descriptor set 3 and with binding 2, set the following values in the 'bindings' array in struct `vkd3d_shader_interface_info`:

```
type = VKD3D_SHADER_DESCRIPTOR_TYPE_CBV
register_space = 0
register_index = 0
binding.set = 3
binding.binding = 2
binding.count = 4
type = VKD3D_SHADER_DESCRIPTOR_TYPE_CBV
register_space = 0
register_index = 6
binding.set = 3
binding.binding = 2
binding.count = 2
```

and then pass { 8, 4 } as static binding offsets here.

This field may be NULL, in which case the corresponding offsets are specified to be 0.

### 3.16.2.2 descriptor\_table\_offset

```
unsigned int vkd3d_shader_descriptor_offset_info::descriptor_table_offset
```

Byte offset within the push constants of an array of 32-bit descriptor array offsets.

See the description of 'binding\_offsets' below.

### 3.16.2.3 uav\_counter\_offsets

```
const struct vkd3d_shader_descriptor_offset* vkd3d_shader_descriptor_offset_info::uav_counter↔
_offsets
```

Pointer to an array of offsets into the descriptor arrays referenced by the 'uav\_counters' array in struct `vkd3d_shader_interface_info`.

This works the same way as [binding\\_offsets](#) above.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.17 vkd3d\_shader\_descriptor\_range Struct Reference

### Data Fields

- enum [vkd3d\\_shader\\_descriptor\\_type](#) **range\_type**
- unsigned int **descriptor\_count**
- unsigned int **base\_shader\_register**
- unsigned int **register\_space**
- unsigned int **descriptor\_table\_offset**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.18 vkd3d\_shader\_descriptor\_range1 Struct Reference

### Data Fields

- enum [vkd3d\\_shader\\_descriptor\\_type](#) **range\_type**
- unsigned int **descriptor\_count**
- unsigned int **base\_shader\_register**
- unsigned int **register\_space**
- enum [vkd3d\\_shader\\_descriptor\\_range\\_flags](#) **flags**
- unsigned int **descriptor\_table\_offset**

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

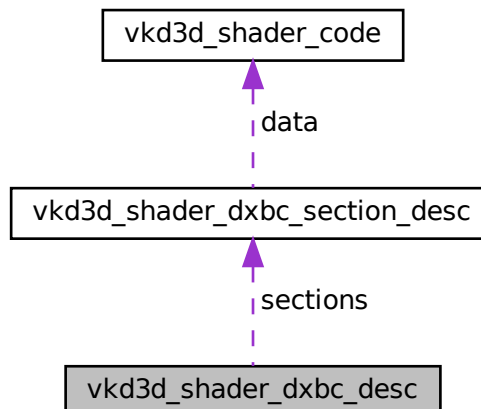
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.19 vkd3d\_shader\_dxbc\_desc Struct Reference

A description of a DXBC blob, as returned by [vkd3d\\_shader\\_parse\\_dxbc\(\)](#).

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_dxbc\_desc:



## Data Fields

- `uint32_t tag`  
*The DXBC tag.*
- `uint32_t checksum [4]`  
*A checksum of the DXBC contents.*
- `unsigned int version`  
*The DXBC version.*
- `size_t size`  
*The total size of the DXBC blob.*
- `size_t section_count`  
*The number of sections contained in the DXBC.*
- `struct vkd3d_shader_dxbc_section_desc * sections`  
*Descriptions of the sections contained in the DXBC.*

### 3.19.1 Detailed Description

A description of a DXBC blob, as returned by `vkd3d_shader_parse_dxbc()`.

Since

1.7

### 3.19.2 Field Documentation

#### 3.19.2.1 tag

```
uint32_t vkd3d_shader_dxbc_desc::tag
```

The DXBC tag.

This will always be "DXBC" in structures returned by this version of vkd3d-shader.

#### 3.19.2.2 version

```
unsigned int vkd3d_shader_dxbc_desc::version
```

The DXBC version.

This will always be 1 in structures returned by this version of vkd3d-shader.

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

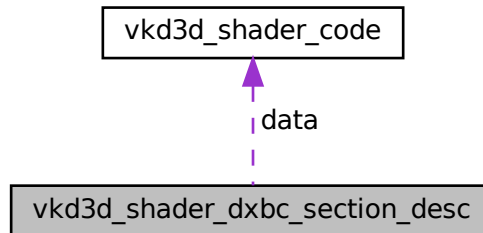
- `/builds/nsivov/vkd3d/include/vkd3d_shader.h`

## 3.20 vkd3d\_shader\_dxbc\_section\_desc Struct Reference

A description of a DXBC section.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_dxbc\_section\_desc:



### Data Fields

- `uint32_t tag`  
*The section tag.*
- struct `vkd3d_shader_code data`  
*The contents of the section.*

### 3.20.1 Detailed Description

A description of a DXBC section.

Since

1.7

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

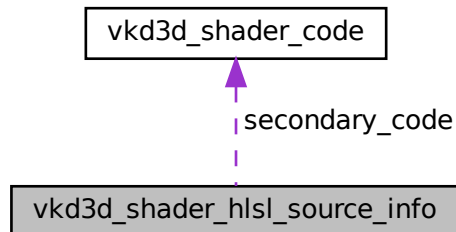
- `/builds/nsivov/vkd3d/include/vkd3d_shader.h`

## 3.21 vkd3d\_shader\_hlsl\_source\_info Struct Reference

A chained structure containing HLSL compilation parameters.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_hlsl\_source\_info:



### Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_HLSL\_SOURCE\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- const char \* [entry\\_point](#)  
*Optional pointer to a null-terminated string containing the shader entry point.*
- struct [vkd3d\\_shader\\_code](#) **secondary\_code**
- const char \* **profile**  
*Pointer to a null-terminated string containing the target shader profile.*

### 3.21.1 Detailed Description

A chained structure containing HLSL compilation parameters.

This structure is optional.

This structure extends [vkd3d\\_shader\\_compile\\_info](#).

This structure contains only input parameters.

Since

1.3

### 3.21.2 Field Documentation

## 3.21.2.1 entry\_point

```
const char* vkd3d_shader_hlsl_source_info::entry_point
```

Optional pointer to a null-terminated string containing the shader entry point.

If this parameter is NULL, vkd3d-shader uses the entry point "main".

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

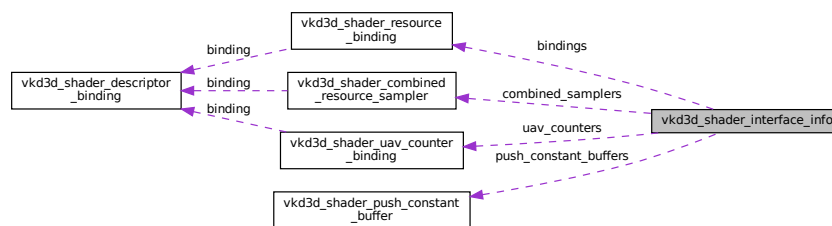
- /builds/nsivov/vkd3d/include/vkd3d\_shader.h

## 3.22 vkd3d\_shader\_interface\_info Struct Reference

A chained structure describing the interface between a compiled shader and the target environment.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_interface\_info:



## Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**  
Must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_INTERFACE\_INFO.
- const void \* **next**  
Optional pointer to a structure containing further parameters.
- const struct [vkd3d\\_shader\\_resource\\_binding](#) \* **bindings**  
Pointer to an array of bindings for shader resource descriptors.
- unsigned int **binding\_count**  
Size, in elements, of [bindings](#).
- const struct [vkd3d\\_shader\\_push\\_constant\\_buffer](#) \* **push\_constant\_buffers**  
Pointer to an array of bindings for push constant buffers.
- unsigned int **push\_constant\_buffer\_count**  
Size, in elements, of [push\\_constant\\_buffers](#).
- const struct [vkd3d\\_shader\\_combined\\_resource\\_sampler](#) \* **combined\_samplers**  
Pointer to an array of bindings for combined samplers.
- unsigned int **combined\_sampler\_count**  
Size, in elements, of [combined\\_samplers](#).
- const struct [vkd3d\\_shader\\_uav\\_counter\\_binding](#) \* **uav\_counters**  
Pointer to an array of bindings for UAV counters.
- unsigned int **uav\_counter\_count**  
Size, in elements, of [uav\\_counters](#).

### 3.22.1 Detailed Description

A chained structure describing the interface between a compiled shader and the target environment.

For example, when compiling Direct3D shader byte code to SPIR-V, this structure contains mappings from Direct3D descriptor registers to SPIR-V descriptor bindings.

This structure is optional. If omitted, [vkd3d\\_shader\\_compile\(\)](#) will use a default mapping, in which resources are mapped to sequential bindings in register set 0.

This structure extends [vkd3d\\_shader\\_compile\\_info](#).

This structure contains only input parameters.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.23 vkd3d\_shader\_macro Struct Reference

A single preprocessor macro, passed as part of struct [vkd3d\\_shader\\_preprocess\\_info](#).

```
#include <vkd3d_shader.h>
```

### Data Fields

- const char \* [name](#)  
*Pointer to a null-terminated string containing the name of a macro.*
- const char \* [value](#)  
*Optional pointer to a null-terminated string containing the expansion of the macro.*

### 3.23.1 Detailed Description

A single preprocessor macro, passed as part of struct [vkd3d\\_shader\\_preprocess\\_info](#).

### 3.23.2 Field Documentation

#### 3.23.2.1 name

```
const char* vkd3d_shader_macro::name
```

Pointer to a null-terminated string containing the name of a macro.

This macro must not be a parameterized (i.e. function-like) macro. If this field is not a valid macro identifier, this macro will be ignored.

## 3.23.2.2 value

```
const char* vkd3d_shader_macro::value
```

Optional pointer to a null-terminated string containing the expansion of the macro.

This field may be set to NULL, in which case the macro has an empty expansion.

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

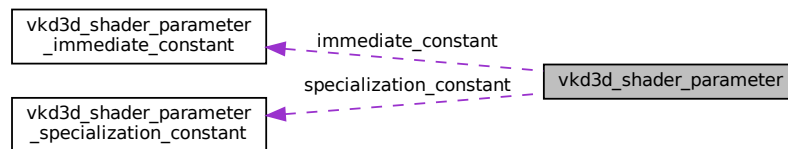
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.24 vkd3d\_shader\_parameter Struct Reference

An individual shader parameter.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_parameter:



## Data Fields

- enum [vkd3d\\_shader\\_parameter\\_name](#) **name**
  - enum [vkd3d\\_shader\\_parameter\\_type](#) **type**
  - enum [vkd3d\\_shader\\_parameter\\_data\\_type](#) **data\_type**
  -
- ```

union {
    struct vkd3d_shader_parameter_immediate_constant immediate_constant
    struct vkd3d_shader_parameter_specialization_constant specialization_constant
} u

```

## 3.24.1 Detailed Description

An individual shader parameter.

This structure is an earlier version of struct [vkd3d\\_shader\\_parameter1](#) which supports fewer parameter types; refer to that structure for usage information.

Only the following types may be used with this structure:

- VKD3D\_SHADER\_PARAMETER\_TYPE\_IMMEDIATE\_CONSTANT
- VKD3D\_SHADER\_PARAMETER\_TYPE\_SPECIALIZATION\_CONSTANT

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

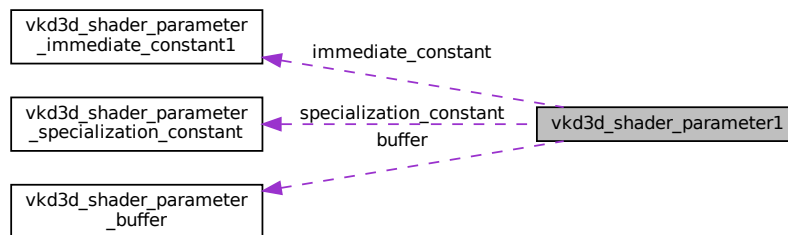
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.25 vkd3d\_shader\_parameter1 Struct Reference

An individual shader parameter.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_parameter1:



### Data Fields

- enum [vkd3d\\_shader\\_parameter\\_name](#) **name**  
*The builtin parameter to be mapped.*
  - enum [vkd3d\\_shader\\_parameter\\_type](#) **type**  
*How the parameter will be provided to the shader.*
  - enum [vkd3d\\_shader\\_parameter\\_data\\_type](#) **data\_type**  
*The data type of the supplied parameter, which determines how it is to be interpreted.*
  -
- ```

union {
    struct vkd3d\_shader\_parameter\_immediate\_constant1 immediate_constant
        Additional information if type is VKD3D_SHADER_PARAMETER_TYPE_IMMEDIATE_CONSTANT.
    struct vkd3d\_shader\_parameter\_specialization\_constant specialization_constant
        Additional information if type is VKD3D_SHADER_PARAMETER_TYPE_SPECIALIZATION_CONSTANT.
    struct vkd3d\_shader\_parameter\_buffer buffer
        Additional information if type is VKD3D_SHADER_PARAMETER_TYPE_BUFFER.
    void * _pointer_pad
    uint32_t _pad [4]
} u
  
```

### 3.25.1 Detailed Description

An individual shader parameter.

This structure is used in struct [vkd3d\\_shader\\_parameter\\_info](#); see there for explanation of shader parameters.

For example, to specify the rasterizer sample count to the shader via an unsigned integer specialization constant with ID 3, set the following members:

- *name* = VKD3D\_SHADER\_PARAMETER\_NAME\_RASTERIZER\_SAMPLE\_COUNT
- *type* = VKD3D\_SHADER\_PARAMETER\_TYPE\_SPECIALIZATION\_CONSTANT
- *data\_type* = VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_UINT32
- *u.specialization\_constant.id* = 3

This structure is an extended version of struct [vkd3d\\_shader\\_parameter](#).

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.26 vkd3d\_shader\_parameter\_buffer Struct Reference

The linkage of a parameter specified through a uniform buffer, used in struct [vkd3d\\_shader\\_parameter1](#).

```
#include <vkd3d_shader.h>
```

### Data Fields

- unsigned int **set**  
*The set of the uniform buffer descriptor.*
- unsigned int **binding**  
*The binding index of the uniform buffer descriptor.*
- uint32\_t **offset**  
*The byte offset of the parameter within the buffer.*

### 3.26.1 Detailed Description

The linkage of a parameter specified through a uniform buffer, used in struct [vkd3d\\_shader\\_parameter1](#).

### 3.26.2 Field Documentation

#### 3.26.2.1 set

```
unsigned int vkd3d_shader_parameter_buffer::set
```

The set of the uniform buffer descriptor.

If the target environment does not support descriptor sets, this value must be set to 0.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.27 vkd3d\_shader\_parameter\_immediate\_constant Struct Reference

The value of an immediate constant parameter, used in struct [vkd3d\\_shader\\_parameter](#).

```
#include <vkd3d_shader.h>
```

### Data Fields

- ```
union {
    uint32_t u32
        The value if the parameter's data type is VKD3D_SHADER_PARAMETER_DATA_TYPE_UINT32.
    float f32
        The value if the parameter's data type is VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32.
} u
```

### 3.27.1 Detailed Description

The value of an immediate constant parameter, used in struct [vkd3d\\_shader\\_parameter](#).

### 3.27.2 Field Documentation

#### 3.27.2.1 f32

```
float vkd3d_shader_parameter_immediate_constant::f32
```

The value if the parameter's data type is VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_FLOAT32.

Since

1.13

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

- `/builds/nsivov/vkd3d/include/vkd3d\_shader.h`

## 3.28 vkd3d\_shader\_parameter\_immediate\_constant1 Struct Reference

The value of an immediate constant parameter, used in struct [vkd3d\\_shader\\_parameter1](#).

```
#include <vkd3d_shader.h>
```

## Data Fields

- ```

union {
    uint32_t u32
        The value if the parameter's data type is VKD3D_SHADER_PARAMETER_DATA_TYPE_UINT32.
    float f32
        The value if the parameter's data type is VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32.
    float f32_vec4 [4]
        A pointer to the value if the parameter's data type is VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32_VEC4.
    void * _pointer_pad
    uint32_t _pad [4]
} u
```

### 3.28.1 Detailed Description

The value of an immediate constant parameter, used in struct [vkd3d\\_shader\\_parameter1](#).

Since

1.13

### 3.28.2 Field Documentation

#### 3.28.2.1 f32\_vec4

```
float vkd3d_shader_parameter_immediate_constant1::f32_vec4[4]
```

A pointer to the value if the parameter's data type is VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_FLOAT32\_VEC4.

Since

1.14

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.29 vkd3d\_shader\_parameter\_info Struct Reference

Interface information regarding a builtin shader parameter.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_parameter\_info:



### Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_PARAMETER\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- const struct [vkd3d\\_shader\\_parameter1](#) \* **parameters**  
*Pointer to an array of dynamic parameters for this shader instance.*
- unsigned int **parameter\_count**  
*Size, in elements, of [parameters](#).*

### 3.29.1 Detailed Description

Interface information regarding a builtin shader parameter.

Like compile options specified with struct [vkd3d\\_shader\\_compile\\_option](#), parameters are used to specify certain values which are not part of the source shader bytecode but which need to be specified in the shader bytecode in the target format. Unlike struct [vkd3d\\_shader\\_compile\\_option](#), however, this structure allows parameters to be specified in a variety of different ways, as described by enum [vkd3d\\_shader\\_parameter\\_type](#).

This structure is an extended version of struct [vkd3d\\_shader\\_parameter](#) as used in struct [vkd3d\\_shader\\_spirv\\_target\\_info](#), which allows more parameter types to be used, and also allows specifying parameters when compiling shaders to target types other than SPIR-V. If this structure is chained along with [vkd3d\\_shader\\_spirv\\_target\\_info](#), any parameters specified in the latter structure are ignored.

This structure is passed to [vkd3d\\_shader\\_compile\(\)](#) and extends [vkd3d\\_shader\\_compile\\_info](#).

This structure contains only input parameters.

Since

1.13

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.30 vkd3d\_shader\_parameter\_specialization\_constant Struct Reference

The linkage of a specialization constant parameter, used in struct [vkd3d\\_shader\\_parameter](#) and struct [vkd3d\\_shader\\_parameter1](#).

```
#include <vkd3d_shader.h>
```

### Data Fields

- [uint32\\_t id](#)

*The ID of the specialization constant.*

### 3.30.1 Detailed Description

The linkage of a specialization constant parameter, used in struct [vkd3d\\_shader\\_parameter](#) and struct [vkd3d\\_shader\\_parameter1](#).

### 3.30.2 Field Documentation

#### 3.30.2.1 id

```
uint32_t vkd3d_shader_parameter_specialization_constant::id
```

The ID of the specialization constant.

If the type comprises more than one constant, such as VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_FLOAT32↔\_VEC4, then a contiguous array of specialization constants should be used, one for each component, and this ID should point to the first component.

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

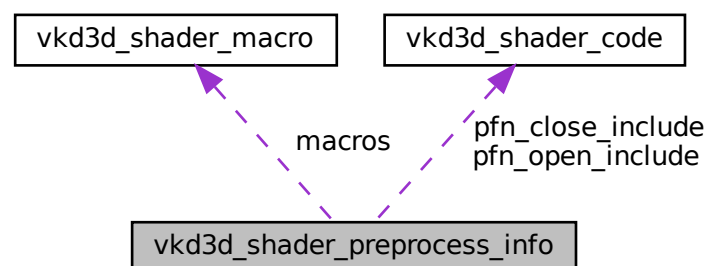
- `/builds/nsivov/vkd3d/include/vkd3d_shader.h`

## 3.31 vkd3d\_shader\_preprocess\_info Struct Reference

A chained structure containing preprocessing parameters.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_preprocess\_info:



## Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_PREPROCESS\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- const struct [vkd3d\\_shader\\_macro](#) \* **macros**  
*Pointer to an array of predefined macros.*
- unsigned int **macro\_count**  
*Size, in elements, of [macros](#).*
- PFN\_vkd3d\_shader\_open\_include **pfn\_open\_include**  
*Optional pointer to a callback function, which will be called in order to evaluate #include directives.*
- PFN\_vkd3d\_shader\_close\_include **pfn\_close\_include**  
*Optional pointer to a callback function, which will be called whenever an included file is closed.*
- void \* **include\_context**  
*User-defined pointer which will be passed unmodified to the [pfn\\_open\\_include](#) and [pfn\\_close\\_include](#) callbacks.*

### 3.31.1 Detailed Description

A chained structure containing preprocessing parameters.

This structure is optional.

This structure extends [vkd3d\\_shader\\_compile\\_info](#).

This structure contains only input parameters.

Since

1.3

### 3.31.2 Field Documentation

#### 3.31.2.1 macros

```
const struct vkd3d\_shader\_macro* vkd3d_shader_preprocess_info::macros
```

Pointer to an array of predefined macros.

Each macro in this array will be expanded as if a corresponding #define statement were prepended to the source code.

If the same macro is specified multiple times, only the last value is used.

### 3.31.2.2 pfn\_close\_include

`PFN_vk_d3d_shader_close_include vk_d3d_shader_preprocess_info::pfn_close_include`

Optional pointer to a callback function, which will be called whenever an included file is closed.

This function will be called exactly once for each successful call to `pfn_open_include`, and should be used to free any resources allocated thereby.

If this field is set to NULL, the `pfn_open_include` field must also be set to NULL.

### 3.31.2.3 pfn\_open\_include

`PFN_vk_d3d_shader_open_include vk_d3d_shader_preprocess_info::pfn_open_include`

Optional pointer to a callback function, which will be called in order to evaluate `#include` directives.

The function receives parameters corresponding to the directive's arguments, and should return the complete text of the included file.

If this field is set to NULL, or if this structure is omitted, vk\_d3d\_shader will attempt to open included files using POSIX file APIs.

If this field is set to NULL, the `pfn_close_include` field must also be set to NULL.

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

- `/builds/nsivov/vkd3d/include/vkd3d_shader.h`

## 3.32 vk\_d3d\_shader\_push\_constant\_buffer Struct Reference

Describes the mapping of a Direct3D constant buffer to a range of push constants in the target environment.

```
#include <vkd3d_shader.h>
```

### Data Fields

- unsigned int `register_space`  
*Register space of the Direct3D resource.*
- unsigned int `register_index`  
*Register index of the Direct3D resource.*
- enum `vk_d3d_shader_visibility` `shader_visibility`  
*Shader stage(s) to which the resource is visible.*
- unsigned int `offset`  
*Offset, in bytes, of the target push constants.*
- unsigned int `size`  
*Size, in bytes, of the target push constants.*

### 3.32.1 Detailed Description

Describes the mapping of a Direct3D constant buffer to a range of push constants in the target environment.

This structure is used in struct [vkd3d\\_shader\\_interface\\_info](#).

### 3.32.2 Field Documentation

#### 3.32.2.1 register\_space

```
unsigned int vkd3d_shader_push_constant_buffer::register_space
```

Register space of the Direct3D resource.

If the source format does not support multiple register spaces, this parameter must be set to 0.

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

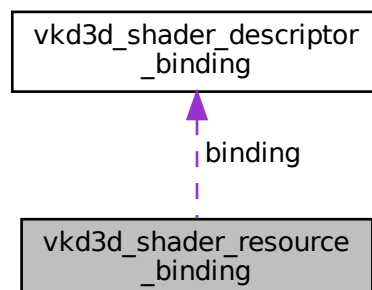
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.33 vkd3d\_shader\_resource\_binding Struct Reference

Describes the mapping of a single resource or resource array to its binding point in the target environment.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_resource\_binding:



## Data Fields

- enum [vkd3d\\_shader\\_descriptor\\_type](#) **type**  
*The type of this descriptor.*
- unsigned int [register\\_space](#)  
*Register space of the Direct3D resource.*
- unsigned int [register\\_index](#)  
*Register index of the Direct3D resource.*
- enum [vkd3d\\_shader\\_visibility](#) **shader\_visibility**  
*Shader stage(s) to which the resource is visible.*
- unsigned int **flags**  
*A combination of zero or more elements of [vkd3d\\_shader\\_binding\\_flag](#).*
- struct [vkd3d\\_shader\\_descriptor\\_binding](#) **binding**  
*The binding in the target environment.*

### 3.33.1 Detailed Description

Describes the mapping of a single resource or resource array to its binding point in the target environment.

For example, to map a Direct3D SRV with register space 2, register "t3" to a Vulkan descriptor in set 4 and with binding 5, set the following members:

- *type* = VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_SRV
- *register\_space* = 2
- *register\_index* = 3
- *binding.set* = 4
- *binding.binding* = 5
- *binding.count* = 1

This structure is used in struct [vkd3d\\_shader\\_interface\\_info](#).

### 3.33.2 Field Documentation

#### 3.33.2.1 register\_index

```
unsigned int vkd3d_shader_resource_binding::register_index
```

Register index of the Direct3D resource.

For legacy Direct3D shaders, vkd3d-shader maps each constant register set to a single constant buffer view. This parameter names the register set to map, and must be a member of enum [vkd3d\\_shader\\_d3dbc\\_constant\\_register](#).

### 3.33.2.2 register\_space

```
unsigned int vkd3d_shader_resource_binding::register_space
```

Register space of the Direct3D resource.

If the source format does not support multiple register spaces, this parameter must be set to 0.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.34 vkd3d\_shader\_root\_constants Struct Reference

### Data Fields

- unsigned int **shader\_register**
- unsigned int **register\_space**
- unsigned int **value\_count**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.35 vkd3d\_shader\_root\_descriptor Struct Reference

### Data Fields

- unsigned int **shader\_register**
- unsigned int **register\_space**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.36 vkd3d\_shader\_root\_descriptor1 Struct Reference

### Data Fields

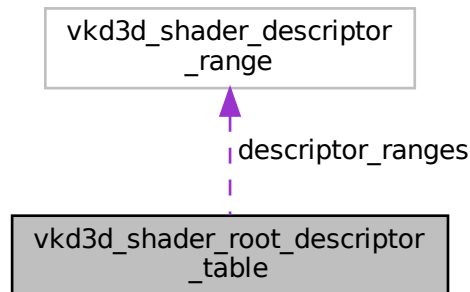
- unsigned int **shader\_register**
- unsigned int **register\_space**
- enum vkd3d\_shader\_root\_descriptor\_flags **flags**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

### 3.37 vkd3d\_shader\_root\_descriptor\_table Struct Reference

Collaboration diagram for vkd3d\_shader\_root\_descriptor\_table:



#### Data Fields

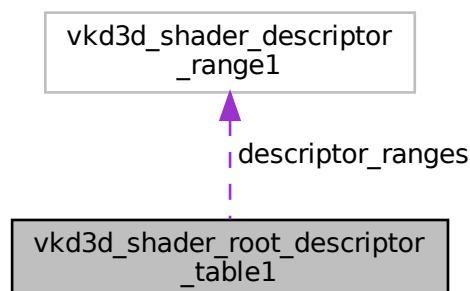
- unsigned int **descriptor\_range\_count**
- const struct [vkd3d\\_shader\\_descriptor\\_range](#) \* **descriptor\_ranges**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

### 3.38 vkd3d\_shader\_root\_descriptor\_table1 Struct Reference

Collaboration diagram for vkd3d\_shader\_root\_descriptor\_table1:



## Data Fields

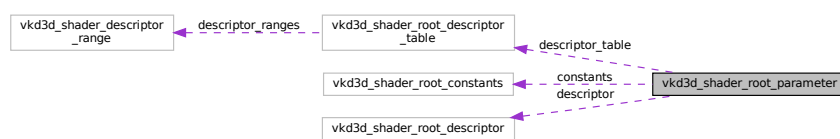
- unsigned int **descriptor\_range\_count**
- const struct [vk3d\\_shader\\_descriptor\\_range1](#) \* **descriptor\_ranges**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.39 vk3d\_shader\_root\_parameter Struct Reference

Collaboration diagram for vk3d\_shader\_root\_parameter:



## Data Fields

- enum vk3d\_shader\_root\_parameter\_type **parameter\_type**
- 

```

union {
    struct vk3d\_shader\_root\_descriptor\_table descriptor_table
    struct vk3d\_shader\_root\_constants constants
    struct vk3d\_shader\_root\_descriptor descriptor
} u

```

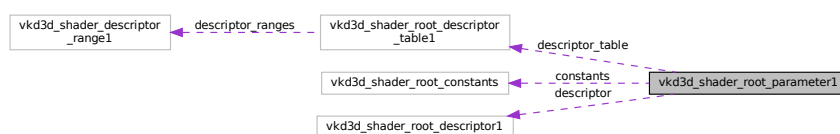
- enum [vk3d\\_shader\\_visibility](#) **shader\_visibility**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.40 vk3d\_shader\_root\_parameter1 Struct Reference

Collaboration diagram for vk3d\_shader\_root\_parameter1:



## Data Fields

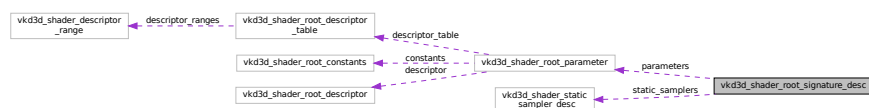
- enum vkd3d\_shader\_root\_parameter\_type **parameter\_type**
- union {
    - struct vkd3d\_shader\_root\_descriptor\_table1 **descriptor\_table**
    - struct vkd3d\_shader\_root\_constants **constants**
    - struct vkd3d\_shader\_root\_descriptor1 **descriptor**
- enum vkd3d\_shader\_visibility **shader\_visibility**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](/builds/nsivov/vkd3d/include/vkd3d_shader.h)

## 3.41 vkd3d\_shader\_root\_signature\_desc Struct Reference

Collaboration diagram for vkd3d\_shader\_root\_signature\_desc:



## Data Fields

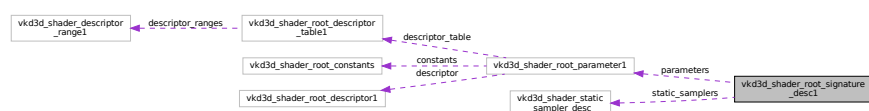
- unsigned int **parameter\_count**
- const struct vkd3d\_shader\_root\_parameter \* **parameters**
- unsigned int **static\_sampler\_count**
- const struct vkd3d\_shader\_static\_sampler\_desc \* **static\_samplers**
- enum vkd3d\_shader\_root\_signature\_flags **flags**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](/builds/nsivov/vkd3d/include/vkd3d_shader.h)

## 3.42 vkd3d\_shader\_root\_signature\_desc1 Struct Reference

Collaboration diagram for vkd3d\_shader\_root\_signature\_desc1:



## Data Fields

- unsigned int **parameter\_count**
- const struct [vk3d\\_shader\\_root\\_parameter1](#) \* **parameters**
- unsigned int **static\_sampler\_count**
- const struct [vk3d\\_shader\\_static\\_sampler\\_desc](#) \* **static\_samplers**
- enum vk3d\_shader\_root\_signature\_flags **flags**

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

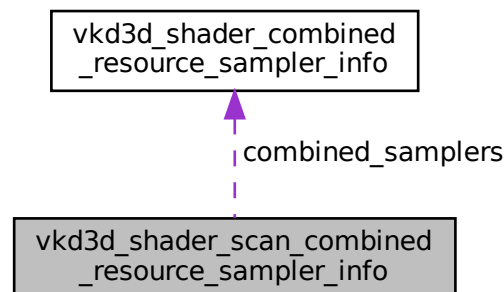
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.43 vk3d\_shader\_scan\_combined\_resource\_sampler\_info Struct Reference

A chained structure describing the resource-sampler pairs used by a shader.

```
#include <vk3d_shader.h>
```

Collaboration diagram for vk3d\_shader\_scan\_combined\_resource\_sampler\_info:



## Data Fields

- enum [vk3d\\_shader\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_COMBINED\_RESOURCE\_SAMPLER\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- struct [vk3d\\_shader\\_combined\\_resource\\_sampler\\_info](#) \* **combined\_samplers**  
*Pointer to an array of resource-sampler pairs.*
- unsigned int **combined\_sampler\_count**  
*The number of resource-sampler pairs in [combined\\_samplers](#).*

### 3.43.1 Detailed Description

A chained structure describing the resource-sampler pairs used by a shader.

This structure extends [vkd3d\\_shader\\_compile\\_info](#).

The information returned in this structure can be used to populate the [vkd3d\\_shader\\_interface\\_info::combined\\_samplers](#) field. This is particularly useful when targeting environments without separate binding points for samplers and resources, like OpenGL.

No resource-sampler pairs are returned for dynamic accesses to resource/sampler descriptor arrays, as can occur in Direct3D shader model 5.1 shaders.

Members of this structure are allocated by vkd3d-shader and should be freed with [vkd3d\\_shader\\_free\\_scan\\_combined\\_resource\\_sampler\\_info](#) when no longer needed.

Since

1.10

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

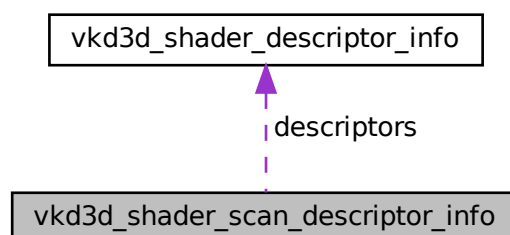
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.44 vkd3d\_shader\_scan\_descriptor\_info Struct Reference

A chained structure enumerating the descriptors declared by a shader.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_scan\_descriptor\_info:



## Data Fields

- enum [vkD3d\\_shader\\_structure\\_type](#) **type**  
*Input; must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_DESCRIPTOR\_INFO.*
- const void \* **next**  
*Input; optional pointer to a structure containing further parameters.*
- struct [vkD3d\\_shader\\_descriptor\\_info](#) \* **descriptors**  
*Output; returns a pointer to an array of descriptors.*
- unsigned int **descriptor\_count**  
*Output; size, in elements, of [descriptors](#).*

### 3.44.1 Detailed Description

A chained structure enumerating the descriptors declared by a shader.

This structure extends [vkD3d\\_shader\\_compile\\_info](#).

When scanning a legacy Direct3D shader, vkD3d-shader enumerates descriptors as follows:

- Each constant register set used by the shader is scanned as a single constant buffer descriptor. There may therefore be up to three such descriptors, one for each register set used by the shader: float, integer, and boolean. The fields are set as follows:
  - The [vkD3d\\_shader\\_descriptor\\_info::type](#) field is set to VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_CBV.
  - The [vkD3d\\_shader\\_descriptor\\_info::register\\_space](#) field is set to zero.
  - The [vkD3d\\_shader\\_descriptor\\_info::register\\_index](#) field is set to a member of enum [vkD3d\\_shader\\_d3dbc\\_constant\\_register](#) denoting which set is used.
  - The [vkD3d\\_shader\\_descriptor\\_info::count](#) field is set to one.
- Each sampler used by the shader is scanned as two separate descriptors, one representing the texture, and one representing the sampler state. If desired, these may be mapped back into a single combined sampler using struct [vkD3d\\_shader\\_combined\\_resource\\_sampler](#). The fields are set as follows:
  - The [vkD3d\\_shader\\_descriptor\\_info::type](#) field is set to VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_SRV and VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_SAMPLER respectively.
  - The [vkD3d\\_shader\\_descriptor\\_info::register\\_space](#) field is set to zero.
  - The [vkD3d\\_shader\\_descriptor\\_info::register\\_index](#) field is set to the binding index of the original sampler, for both descriptors.
  - The [vkD3d\\_shader\\_descriptor\\_info::count](#) field is set to one.

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

- `/builds/nsivov/vkd3d/include/vkd3d_shader.h`

## 3.45 vkD3d\_shader\_scan\_hull\_shader\_tessellation\_info Struct Reference

A chained structure describing the tessellation information in a hull shader.

```
#include <vkd3d_shader.h>
```

## Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_HULL\_SHADER\_TESSELLATION\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- enum vkd3d\_shader\_tessellator\_output\_primitive **output\_primitive**  
*The tessellation output primitive.*
- enum vkd3d\_shader\_tessellator\_partitioning **partitioning**  
*The tessellation partitioning mode.*

### 3.45.1 Detailed Description

A chained structure describing the tessellation information in a hull shader.

This structure extends [vkd3d\\_shader\\_compile\\_info](#).

Since

1.15

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

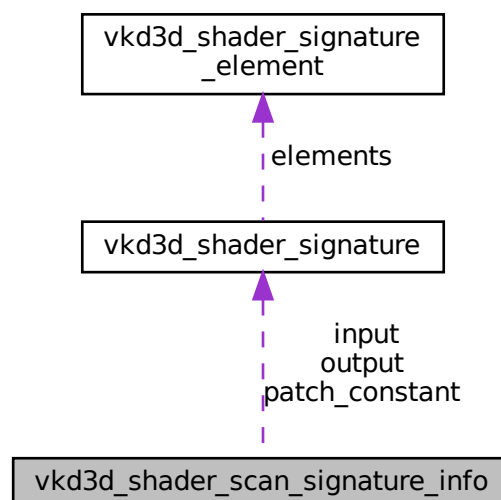
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.46 vkd3d\_shader\_scan\_signature\_info Struct Reference

A chained structure containing descriptions of shader inputs and outputs.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_scan\_signature\_info:



## Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**  
*Must be set to VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_SIGNATURE\_INFO.*
- const void \* **next**  
*Optional pointer to a structure containing further parameters.*
- struct [vkd3d\\_shader\\_signature](#) **input**  
*The shader input varyings.*
- struct [vkd3d\\_shader\\_signature](#) **output**  
*The shader output varyings.*
- struct [vkd3d\\_shader\\_signature](#) **patch\_constant**  
*The shader patch constant varyings.*

### 3.46.1 Detailed Description

A chained structure containing descriptions of shader inputs and outputs.

This structure is currently implemented only for DXBC and legacy D3D bytecode source types. For DXBC shaders, the returned information is parsed directly from the signatures embedded in the DXBC shader. For legacy D3D shaders, the returned information is synthesized based on registers declared or used by shader instructions. For all other shader types, the structure is zeroed.

All members (except for [type](#) and [next](#)) are output-only.

This structure is passed to [vkd3d\\_shader\\_scan\(\)](#) and extends [vkd3d\\_shader\\_compile\\_info](#).

Members of this structure are allocated by vkd3d-shader and should be freed with [vkd3d\\_shader\\_free\\_scan\\_signature\\_info\(\)](#) when no longer needed.

All signatures may contain pointers into the input shader, and should only be accessed while the input shader remains valid.

Signature elements are synthesized from legacy Direct3D bytecode as follows:

- The [vkd3d\\_shader\\_signature\\_element::semantic\\_name](#) field is set to an uppercase string corresponding to the HLSL name for the usage, e.g. "POSITION", "BLENDWEIGHT", "COLOR", "PSIZE", etc.
- The [vkd3d\\_shader\\_signature\\_element::semantic\\_index](#) field is set to the usage index.
- The [vkd3d\\_shader\\_signature\\_element::stream\\_index](#) is always 0.

Signature elements are synthesized for any input or output register declared or used in a legacy Direct3D bytecode shader, including the following:

- Shader model 1 and 2 colour and texture coordinate registers.
- The shader model 1 pixel shader output register.
- Shader model 1 and 2 vertex shader output registers (position, fog, and point size).
- Shader model 3 pixel shader system value input registers (pixel position and face).

Since

1.9

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

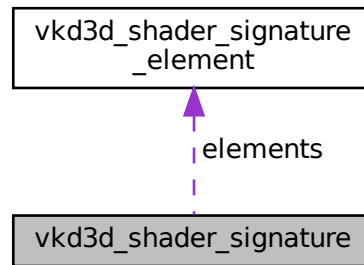
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.47 vkd3d\_shader\_signature Struct Reference

Description of a shader input or output signature.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_signature:



### Data Fields

- struct [vkd3d\\_shader\\_signature\\_element](#) \* **elements**  
*Pointer to an array of varying.*
- unsigned int **element\_count**  
*Size, in elements, of [elements](#).*

#### 3.47.1 Detailed Description

Description of a shader input or output signature.

This structure is populated by [vkd3d\\_shader\\_parse\\_input\\_signature\(\)](#).

The helper function [vkd3d\\_shader\\_find\\_signature\\_element\(\)](#) will look up a varying element by its semantic name, semantic index, and stream index.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.48 vkd3d\_shader\_signature\_element Struct Reference

A single shader varying, returned as part of struct [vkd3d\\_shader\\_signature](#).

```
#include <vkd3d_shader.h>
```

## Data Fields

- const char \* **semantic\_name**  
*Semantic name.*
- unsigned int **semantic\_index**  
*Semantic index, or 0 if the semantic is not indexed.*
- unsigned int **stream\_index**  
*Stream index of a geometry shader output semantic.*
- enum **vkd3d\_shader\_sysval\_semantic sysval\_semantic**  
*System value semantic.*
- enum **vkd3d\_shader\_component\_type component\_type**  
*Data type.*
- unsigned int **register\_index**  
*Register index.*
- unsigned int **mask**  
*Mask of the register components allocated to this varying.*
- unsigned int **used\_mask**  
*Subset of [mask](#) which the shader reads from or writes to.*
- enum **vkd3d\_shader\_minimum\_precision min\_precision**  
*Minimum interpolation precision.*

### 3.48.1 Detailed Description

A single shader varying, returned as part of struct [vkd3d\\_shader\\_signature](#).

### 3.48.2 Field Documentation

#### 3.48.2.1 stream\_index

```
unsigned int vkd3d_shader_signature_element::stream_index
```

Stream index of a geometry shader output semantic.

If the signature is not a geometry shader output signature, this field will be set to 0.

#### 3.48.2.2 sysval\_semantic

```
enum vkd3d_shader_sysval_semantic vkd3d_shader_signature_element::sysval_semantic
```

System value semantic.

If the varying is not a system value, this field will be set to VKD3D\_SHADER\_SV\_NONE.

### 3.48.2.3 used\_mask

```
unsigned int vkd3d_shader_signature_element::used_mask
```

Subset of [mask](#) which the shader reads from or writes to.

Unlike Direct3D shader bytecode, the mask for output and tessellation signatures is not inverted, i.e. bits set in this field denote components which are written to.

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.49 vkd3d\_shader\_spirv\_domain\_shader\_target\_info Struct Reference

### Data Fields

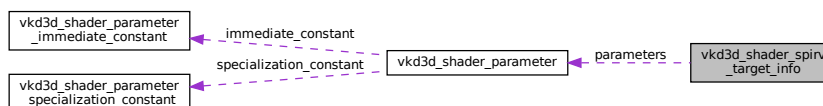
- enum [vkd3d\\_shader\\_structure\\_type](#) **type**
- const void \* **next**
- enum vkd3d\_shader\_tessellator\_output\_primitive **output\_primitive**
- enum vkd3d\_shader\_tessellator\_partitioning **partitioning**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

## 3.50 vkd3d\_shader\_spirv\_target\_info Struct Reference

Collaboration diagram for vkd3d\_shader\_spirv\_target\_info:



### Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**
- const void \* **next**
- const char \* **entry\_point**
- enum [vkd3d\\_shader\\_spirv\\_environment](#) **environment**
- enum [vkd3d\\_shader\\_spirv\\_extension](#) \* **extensions**
- unsigned int **extension\_count**
- const struct [vkd3d\\_shader\\_parameter](#) \* **parameters**
- unsigned int **parameter\_count**
- bool **dual\_source\_blending**
- const unsigned int \* **output\_swizzles**
- unsigned int **output\_swizzle\_count**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

### 3.51 vkd3d\_shader\_static\_sampler\_desc Struct Reference

#### Data Fields

- enum vkd3d\_shader\_filter **filter**
- enum vkd3d\_shader\_texture\_address\_mode **address\_u**
- enum vkd3d\_shader\_texture\_address\_mode **address\_v**
- enum vkd3d\_shader\_texture\_address\_mode **address\_w**
- float **mip\_lod\_bias**
- unsigned int **max\_anisotropy**
- enum vkd3d\_shader\_comparison\_func **comparison\_func**
- enum vkd3d\_shader\_static\_border\_colour **border\_colour**
- float **min\_lod**
- float **max\_lod**
- unsigned int **shader\_register**
- unsigned int **register\_space**
- enum [vkd3d\\_shader\\_visibility](#) **shader\_visibility**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

### 3.52 vkd3d\_shader\_transform\_feedback\_element Struct Reference

#### Data Fields

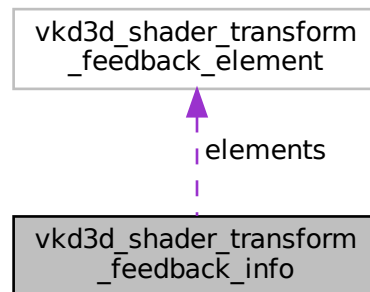
- unsigned int **stream\_index**
- const char \* **semantic\_name**
- unsigned int **semantic\_index**
- uint8\_t **component\_index**
- uint8\_t **component\_count**
- uint8\_t **output\_slot**

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

- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

### 3.53 vkd3d\_shader\_transform\_feedback\_info Struct Reference

Collaboration diagram for vkd3d\_shader\_transform\_feedback\_info:



#### Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) `type`
- const void \* `next`
- const struct [vkd3d\\_shader\\_transform\\_feedback\\_element](#) \* `elements`
- unsigned int `element_count`
- const unsigned int \* `buffer_strides`
- unsigned int `buffer_stride_count`

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

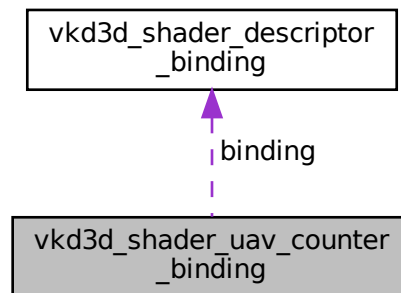
- [/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#)

### 3.54 vkd3d\_shader\_uav\_counter\_binding Struct Reference

Describes the mapping of a single Direct3D UAV counter.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_uav\_counter\_binding:



## Data Fields

- unsigned int [register\\_space](#)  
*Register space of the Direct3D UAV descriptor.*
- unsigned int **register\_index**  
*Register index of the Direct3D UAV descriptor.*
- enum [vkd3d\\_shader\\_visibility](#) **shader\_visibility**  
*Shader stage(s) to which the UAV counter is visible.*
- struct [vkd3d\\_shader\\_descriptor\\_binding](#) **binding**  
*The binding in the target environment.*
- unsigned int **offset**

### 3.54.1 Detailed Description

Describes the mapping of a single Direct3D UAV counter.

This structure is used in struct [vkd3d\\_shader\\_interface\\_info](#).

### 3.54.2 Field Documentation

#### 3.54.2.1 register\_space

```
unsigned int vkd3d_shader_uav_counter_binding::register_space
```

Register space of the Direct3D UAV descriptor.

If the source format does not support multiple register spaces, this parameter must be set to 0.

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

- `/builds/nsivov/vkd3d/include/vkd3d_shader.h`

## 3.55 vkd3d\_shader\_varying\_map Struct Reference

Describes the mapping of a output varying register in a shader stage, to an input varying register in the following shader stage.

```
#include <vkd3d_shader.h>
```

### Data Fields

- unsigned int [output\\_signature\\_index](#)  
*The signature index (in the output signature) of the output varying.*
- unsigned int **input\_register\_index**  
*The register index of the input varying to map this register to.*
- unsigned int **input\_mask**  
*The mask consumed by the destination register.*

### 3.55.1 Detailed Description

Describes the mapping of a output varying register in a shader stage, to an input varying register in the following shader stage.

This structure is used in struct [vkd3d\\_shader\\_varying\\_map\\_info](#).

### 3.55.2 Field Documentation

#### 3.55.2.1 output\_signature\_index

```
unsigned int vkd3d_shader_varying_map::output_signature_index
```

The signature index (in the output signature) of the output varying.

If greater than or equal to the number of elements in the output signature, signifies that the varying is consumed by the next stage but not written by this one.

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

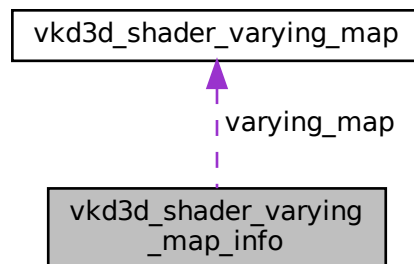
- `/builds/nsivov/vkd3d/include/vkd3d_shader.h`

### 3.56 vkd3d\_shader\_varying\_map\_info Struct Reference

A chained structure which describes how output varyings in this shader stage should be mapped to input varyings in the next stage.

```
#include <vkd3d_shader.h>
```

Collaboration diagram for vkd3d\_shader\_varying\_map\_info:



#### Data Fields

- enum [vkd3d\\_shader\\_structure\\_type](#) **type**  
Must be set to `VKD3D_SHADER_STRUCTURE_TYPE_VARYING_MAP_INFO`.
- const void \* **next**  
Optional pointer to a structure containing further parameters.
- const struct [vkd3d\\_shader\\_varying\\_map](#) \* **varying\_map**  
A mapping of output varyings in this shader stage to input varyings in the next shader stage.
- unsigned int **varying\_count**  
The number of registers provided in [varying\\_map](#).

#### 3.56.1 Detailed Description

A chained structure which describes how output varyings in this shader stage should be mapped to input varyings in the next stage.

This structure is optional. It should not be provided if there is no shader stage. However, depending on the input and output formats, this structure may be necessary in order to generate shaders which correctly match each other.

If this structure is absent, vkd3d-shader will map varyings from one stage to another based on their register index. For Direct3D shader model 3.0, such a default mapping will be incorrect unless the registers are allocated in the same order, and hence this field is necessary to correctly match inter-stage varyings. This mapping may also be necessary under other circumstances where the varying interface does not match exactly.

This structure is passed to [vkd3d\\_shader\\_compile\(\)](#) and extends [vkd3d\\_shader\\_compile\\_info](#).

This structure contains only input parameters.

Since

1.9





## Chapter 4

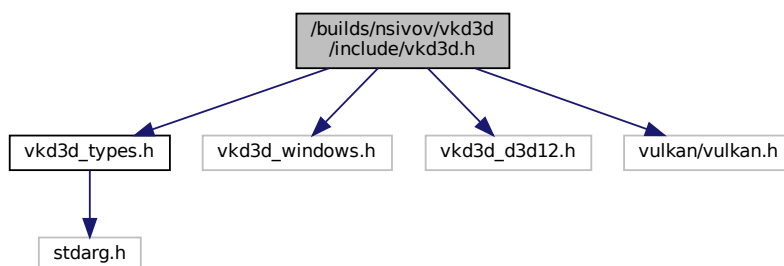
# File Documentation

### 4.1 /builds/nsivov/vkd3d/include/vkd3d.h File Reference

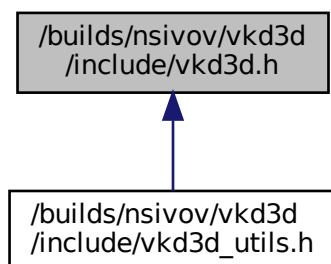
This file contains definitions for the vkd3d library.

```
#include <vkd3d_types.h>
#include <vkd3d_windows.h>
#include <vkd3d_d3d12.h>
#include <vulkan/vulkan.h>
```

Include dependency graph for vkd3d.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [vkd3d\\_instance\\_create\\_info](#)  
*A chained structure containing instance creation parameters.*
- struct [vkd3d\\_optional\\_instance\\_extensions\\_info](#)  
*A chained structure to specify optional instance extensions.*
- struct [vkd3d\\_application\\_info](#)  
*A chained structure to specify application information.*
- struct [vkd3d\\_host\\_time\\_domain\\_info](#)  
*A chained structure to specify the host time domain.*
- struct [vkd3d\\_device\\_create\\_info](#)  
*A chained structure containing device creation parameters.*
- struct [vkd3d\\_optional\\_device\\_extensions\\_info](#)  
*A chained structure to specify optional device extensions.*
- struct [vkd3d\\_image\\_resource\\_create\\_info](#)  
*A chained structure containing the parameters to create a D3D12 resource backed by a Vulkan image.*

## Macros

- #define [VKD3D\\_RESOURCE\\_INITIAL\\_STATE\\_TRANSITION](#) 0x00000001  
*When specified as a flag of [vkd3d\\_image\\_resource\\_create\\_info](#), it means that vkd3d will do the initial transition operation on the image from VK\_IMAGE\_LAYOUT\_UNDEFINED to its appropriate Vulkan layout (depending on its D3D12 resource state).*
- #define [VKD3D\\_RESOURCE\\_PRESENT\\_STATE\\_TRANSITION](#) 0x00000002  
*When specified as a flag of [vkd3d\\_image\\_resource\\_create\\_info](#), it means that field present\_state is honored.*
- #define [VKD3D\\_API](#) VKD3D\_IMPORT

## Typedefs

- typedef HRESULT(\* [PFN\\_vkd3d\\_signal\\_event](#)) (HANDLE event)
- typedef void(\* [PFN\\_vkd3d\\_thread](#)) (void \*data)
- typedef void(\* [PFN\\_vkd3d\\_create\\_thread](#)) (PFN\_vkd3d\_thread thread\_main, void \*data)
- typedef HRESULT(\* [PFN\\_vkd3d\\_join\\_thread](#)) (void \*thread)
- typedef HRESULT(\* [PFN\\_vkd3d\\_create\\_instance](#)) (const struct [vkd3d\\_instance\\_create\\_info](#) \*create\_info, struct vkd3d\_instance \*\*instance)
- typedef ULONG(\* [PFN\\_vkd3d\\_instance\\_decref](#)) (struct vkd3d\_instance \*instance)
- typedef VkInstance(\* [PFN\\_vkd3d\\_instance\\_get\\_vk\\_instance](#)) (struct vkd3d\_instance \*instance)
- typedef ULONG(\* [PFN\\_vkd3d\\_instance\\_incref](#)) (struct vkd3d\_instance \*instance)
- typedef HRESULT(\* [PFN\\_vkd3d\\_create\\_device](#)) (const struct [vkd3d\\_device\\_create\\_info](#) \*create\_info, REFIID iid, void \*\*device)
- typedef IUnknown(\* [PFN\\_vkd3d\\_get\\_device\\_parent](#)) (ID3D12Device \*device)
- typedef VkDevice(\* [PFN\\_vkd3d\\_get\\_vk\\_device](#)) (ID3D12Device \*device)
- typedef VkPhysicalDevice(\* [PFN\\_vkd3d\\_get\\_vk\\_physical\\_device](#)) (ID3D12Device \*device)
- typedef struct vkd3d\_instance(\* [PFN\\_vkd3d\\_instance\\_from\\_device](#)) (ID3D12Device \*device)
- typedef uint32\_t(\* [PFN\\_vkd3d\\_get\\_vk\\_queue\\_family\\_index](#)) (ID3D12CommandQueue \*queue)
- typedef VkQueue(\* [PFN\\_vkd3d\\_acquire\\_vk\\_queue](#)) (ID3D12CommandQueue \*queue)
- typedef void(\* [PFN\\_vkd3d\\_release\\_vk\\_queue](#)) (ID3D12CommandQueue \*queue)
- typedef HRESULT(\* [PFN\\_vkd3d\\_create\\_image\\_resource](#)) (ID3D12Device \*device, const struct [vkd3d\\_image\\_resource\\_create\\_info](#) \*create\_info, ID3D12Resource \*\*resource)
- typedef ULONG(\* [PFN\\_vkd3d\\_resource\\_decref](#)) (ID3D12Resource \*resource)
- typedef ULONG(\* [PFN\\_vkd3d\\_resource\\_incref](#)) (ID3D12Resource \*resource)

- typedef HRESULT(\* **PFN\_vk\_d3d\_serialize\_root\_signature**) (const D3D12\_ROOT\_SIGNATURE\_DESC \*desc, D3D\_ROOT\_SIGNATURE\_VERSION version, ID3DBlob \*\*blob, ID3DBlob \*\*error\_blob)
- typedef HRESULT(\* **PFN\_vk\_d3d\_create\_root\_signature\_deserializer**) (const void \*data, SIZE\_T data\_size, REFIID iid, void \*\*deserializer)
- typedef VkFormat(\* **PFN\_vk\_d3d\_get\_vk\_format**) (DXGI\_FORMAT format)
- typedef DXGI\_FORMAT(\* **PFN\_vk\_d3d\_get\_dxgi\_format**) (VkFormat format)
- typedef HRESULT(\* **PFN\_vk\_d3d\_serialize\_versioned\_root\_signature**) (const D3D12\_VERSIONED\_ROOT\_SIGNATURE\_DESC \*desc, ID3DBlob \*\*blob, ID3DBlob \*\*error\_blob)
- typedef HRESULT(\* **PFN\_vk\_d3d\_create\_versioned\_root\_signature\_deserializer**) (const void \*data, SIZE\_T data\_size, REFIID iid, void \*\*deserializer)
- typedef void(\* **PFN\_vk\_d3d\_set\_log\_callback**) (PFN\_vk\_d3d\_log callback)  
Type of *vk\_d3d\_set\_log\_callback()*.
- typedef HRESULT(\* **PFN\_vk\_d3d\_queue\_signal\_on\_cpu**) (ID3D12CommandQueue \*queue, ID3D12Fence \*fence, uint64\_t value)  
Type of *vk\_d3d\_queue\_signal\_on\_cpu()*.

## Enumerations

- enum **vk\_d3d\_structure\_type** {  
VKD3D\_STRUCTURE\_TYPE\_INSTANCE\_CREATE\_INFO, VKD3D\_STRUCTURE\_TYPE\_DEVICE\_CREATE\_INFO,  
VKD3D\_STRUCTURE\_TYPE\_IMAGE\_RESOURCE\_CREATE\_INFO, VKD3D\_STRUCTURE\_TYPE\_OPTIONAL\_INSTANCE\_CREATE\_INFO,  
VKD3D\_STRUCTURE\_TYPE\_OPTIONAL\_DEVICE\_EXTENSIONS\_INFO, VKD3D\_STRUCTURE\_TYPE\_APPLICATION\_INFO,  
VKD3D\_STRUCTURE\_TYPE\_HOST\_TIME\_DOMAIN\_INFO, VKD3D\_FORCE\_32\_BIT\_ENUM=(VKD3D\_STRUCTURE\_TYPE)}  
The type of a chained structure.
- enum **vk\_d3d\_api\_version** {  
VKD3D\_API\_VERSION\_1\_0, VKD3D\_API\_VERSION\_1\_1, VKD3D\_API\_VERSION\_1\_2, VKD3D\_API\_VERSION\_1\_3,  
VKD3D\_API\_VERSION\_1\_4, VKD3D\_API\_VERSION\_1\_5, VKD3D\_API\_VERSION\_1\_6, VKD3D\_API\_VERSION\_1\_7,  
VKD3D\_API\_VERSION\_1\_8, VKD3D\_API\_VERSION\_1\_9, VKD3D\_API\_VERSION\_1\_10, VKD3D\_API\_VERSION\_1\_11,  
VKD3D\_API\_VERSION\_1\_12, VKD3D\_API\_VERSION\_1\_13, VKD3D\_API\_VERSION\_1\_14, VKD3D\_API\_VERSION\_1\_15,  
VKD3D\_FORCE\_32\_BIT\_ENUM=(VKD3D\_STRUCTURE\_TYPE)}

## Functions

- VKD3D\_API HRESULT **vk\_d3d\_create\_instance** (const struct **vk\_d3d\_instance\_create\_info** \*create\_info, struct vk\_d3d\_instance \*\*instance)
- VKD3D\_API ULONG **vk\_d3d\_instance\_decref** (struct vk\_d3d\_instance \*instance)
- VKD3D\_API VkInstance **vk\_d3d\_instance\_get\_vk\_instance** (struct vk\_d3d\_instance \*instance)
- VKD3D\_API ULONG **vk\_d3d\_instance\_incref** (struct vk\_d3d\_instance \*instance)
- VKD3D\_API HRESULT **vk\_d3d\_create\_device** (const struct **vk\_d3d\_device\_create\_info** \*create\_info, REFIID iid, void \*\*device)
- VKD3D\_API IUnknown \* **vk\_d3d\_get\_device\_parent** (ID3D12Device \*device)
- VKD3D\_API VkDevice **vk\_d3d\_get\_vk\_device** (ID3D12Device \*device)
- VKD3D\_API VkPhysicalDevice **vk\_d3d\_get\_vk\_physical\_device** (ID3D12Device \*device)
- VKD3D\_API struct vk\_d3d\_instance \* **vk\_d3d\_instance\_from\_device** (ID3D12Device \*device)
- VKD3D\_API uint32\_t **vk\_d3d\_get\_vk\_queue\_family\_index** (ID3D12CommandQueue \*queue)
- VKD3D\_API VkQueue **vk\_d3d\_acquire\_vk\_queue** (ID3D12CommandQueue \*queue)  
Acquire the Vulkan queue backing a command queue.

- VKD3D\_API void [vkd3d\\_release\\_vk\\_queue](#) (ID3D12CommandQueue \*queue)  
*Release the Vulkan queue backing a command queue.*
- VKD3D\_API HRESULT [vkd3d\\_create\\_image\\_resource](#) (ID3D12Device \*device, const struct [vkd3d\\_image\\_resource\\_create\\_info](#), ID3D12Resource \*\*resource)
- VKD3D\_API ULONG [vkd3d\\_resource\\_decref](#) (ID3D12Resource \*resource)
- VKD3D\_API ULONG [vkd3d\\_resource\\_incref](#) (ID3D12Resource \*resource)
- VKD3D\_API HRESULT [vkd3d\\_serialize\\_root\\_signature](#) (const D3D12\_ROOT\_SIGNATURE\_DESC \*desc, D3D\_ROOT\_SIGNATURE\_VERSION version, ID3DBlob \*\*blob, ID3DBlob \*\*error\_blob)
- VKD3D\_API HRESULT [vkd3d\\_create\\_root\\_signature\\_deserializer](#) (const void \*data, SIZE\_T data\_size, REFIID iid, void \*\*deserializer)
- VKD3D\_API VkFormat [vkd3d\\_get\\_vk\\_format](#) (DXGI\_FORMAT format)
- VKD3D\_API DXGI\_FORMAT [vkd3d\\_get\\_dxgi\\_format](#) (VkFormat format)
- VKD3D\_API HRESULT [vkd3d\\_serialize\\_versioned\\_root\\_signature](#) (const D3D12\_VERSIONED\_ROOT\_SIGNATURE\_DESC \*desc, ID3DBlob \*\*blob, ID3DBlob \*\*error\_blob)
- VKD3D\_API HRESULT [vkd3d\\_create\\_versioned\\_root\\_signature\\_deserializer](#) (const void \*data, SIZE\_T data\_size, REFIID iid, void \*\*deserializer)
- VKD3D\_API void [vkd3d\\_set\\_log\\_callback](#) (PFN\_vkd3d\_log callback)  
*Set a callback to be called when vkd3d outputs debug logging.*
- VKD3D\_API HRESULT [vkd3d\\_queue\\_signal\\_on\\_cpu](#) (ID3D12CommandQueue \*queue, ID3D12Fence \*fence, uint64\_t value)  
*Signal a fence on the CPU once all the currently outstanding queue work is submitted to Vulkan.*

### 4.1.1 Detailed Description

This file contains definitions for the vkd3d library.

The vkd3d library is a 3D graphics library built on top of Vulkan. It has an API very similar, but not identical, to Direct3D 12.

Since

1.0

### 4.1.2 Macro Definition Documentation

#### 4.1.2.1 VKD3D\_RESOURCE\_INITIAL\_STATE\_TRANSITION

```
#define VKD3D_RESOURCE_INITIAL_STATE_TRANSITION 0x00000001
```

When specified as a flag of [vkd3d\\_image\\_resource\\_create\\_info](#), it means that vkd3d will do the initial transition operation on the image from VK\_IMAGE\_LAYOUT\_UNDEFINED to its appropriate Vulkan layout (depending on its D3D12 resource state).

If this flag is not specified the caller is responsible for transitioning the Vulkan image to the appropriate layout.

### 4.1.3 Typedef Documentation

#### 4.1.3.1 PFN\_vkd3d\_queue\_signal\_on\_cpu

```
typedef HRESULT(* PFN_vkd3d_queue_signal_on_cpu) (ID3D12CommandQueue *queue, ID3D12Fence *fence,
uint64_t value)
```

Type of [vkd3d\\_queue\\_signal\\_on\\_cpu\(\)](#).

Since

1.15

#### 4.1.3.2 PFN\_vkd3d\_set\_log\_callback

```
typedef void(* PFN_vkd3d_set_log_callback) (PFN_vkd3d_log callback)
```

Type of [vkd3d\\_set\\_log\\_callback\(\)](#).

Since

1.4

### 4.1.4 Enumeration Type Documentation

#### 4.1.4.1 vkd3d\_structure\_type

```
enum vkd3d_structure_type
```

The type of a chained structure.

Enumerator

VKD3D_STRUCTURE_TYPE_INSTANCE_CREATE_INFO ↔	The structure is a <a href="#">vkd3d_instance_create_info</a> structure.
VKD3D_STRUCTURE_TYPE_DEVICE_CREATE_INFO ↔	The structure is a <a href="#">vkd3d_device_create_info</a> structure.
VKD3D_STRUCTURE_TYPE_IMAGE_RESOURCE_CREATE_INFO ↔	The structure is a <a href="#">vkd3d_image_resource_create_info</a> structure.
VKD3D_STRUCTURE_TYPE_OPTIONAL_INSTANCE_EXTENSIONS_INFO ↔	The structure is a <a href="#">vkd3d_optional_instance_extensions_info</a> structure.  Since 1.1
VKD3D_STRUCTURE_TYPE_OPTIONAL_DEVICE_EXTENSIONS_INFO ↔	The structure is a <a href="#">vkd3d_optional_device_extensions_info</a> structure.  Since 1.2
Generated by Doxygen	

## Enumerator

VKD3D_STRUCTURE_TYPE_APPLICATION_INFO	The structure is a <a href="#">vkd3d_application_info</a> structure.  Since 1.2
VKD3D_STRUCTURE_TYPE_HOST_TIME_↔ DOMAIN_INFO	The structure is a <a href="#">vkd3d_host_time_domain_info</a> structure.  Since 1.3

## 4.1.5 Function Documentation

### 4.1.5.1 vkd3d\_acquire\_vk\_queue()

```
VKD3D_API VkQueue vkd3d_acquire_vk_queue (
    ID3D12CommandQueue * queue )
```

Acquire the Vulkan queue backing a command queue.

While a queue is acquired by the client, it is locked so that neither the vkd3d library nor other threads can submit work to it. For that reason it should be released as soon as possible with [vkd3d\\_release\\_vk\\_queue\(\)](#). The lock is not reentrant, so the same queue must not be acquired more than once by the same thread.

Work submitted through the Direct3D 12 API exposed by vkd3d is not always immediately submitted to the Vulkan queue; sometimes it is kept in another internal queue, which might not necessarily be empty at the time [vkd3d\\_acquire\\_vk\\_queue\(\)](#) is called. For this reason, work submitted directly to the Vulkan queue might appear to the Vulkan driver as being submitted before other work submitted through the Direct3D 12 API. If this is not desired, it is recommended to synchronize work submission using an ID3D12Fence object:

1. submit work through the Direct3D 12 API;
2. call [vkd3d\\_queue\\_signal\\_on\\_cpu\(\)](#);
3. wait for the fence to be signalled;
4. call [vkd3d\\_acquire\\_vk\\_queue\(\)](#); it is guaranteed that all work submitted at point 1 has already been submitted to Vulkan (though not necessarily executed).

Since

1.0

#### 4.1.5.2 vkd3d\_queue\_signal\_on\_cpu()

```
VKD3D_API HRESULT vkd3d_queue_signal_on_cpu (
    ID3D12CommandQueue * queue,
    ID3D12Fence * fence,
    uint64_t value )
```

Signal a fence on the CPU once all the currently outstanding queue work is submitted to Vulkan.

The fence will be signalled on the CPU (as if `ID3D12Fence_Signal()` was called) once all the work submitted through the Direct3D 12 API before `vkd3d_queue_signal_on_cpu()` is called has left the internal queue and has been submitted to the underlying Vulkan queue. Read the documentation for `vkd3d_acquire_vk_queue()` for more details.

Since

1.15

#### 4.1.5.3 vkd3d\_release\_vk\_queue()

```
VKD3D_API void vkd3d_release_vk_queue (
    ID3D12CommandQueue * queue )
```

Release the Vulkan queue backing a command queue.

This must be paired to an earlier corresponding `vkd3d_acquire_vk_queue()`. After this function is called, the Vulkan queue returned by `vkd3d_acquire_vk_queue()` must not be used any more.

Since

1.0

#### 4.1.5.4 vkd3d\_set\_log\_callback()

```
VKD3D_API void vkd3d_set_log_callback (
    PFN_vkd3d_log callback )
```

Set a callback to be called when vkd3d outputs debug logging.

If NULL, or if this function has not been called, libvkd3d will print all enabled log output to stderr.

Calling this function will also set the log callback for libvkd3d-shader.

Parameters

<i>callback</i>	Callback function to set.
-----------------	---------------------------

Since

1.4

## 4.2 vkd3d.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright 2016 Józef Kucia for CodeWeavers
3  *
4  * This library is free software; you can redistribute it and/or
5  * modify it under the terms of the GNU Lesser General Public
6  * License as published by the Free Software Foundation; either
7  * version 2.1 of the License, or (at your option) any later version.
8  *
9  * This library is distributed in the hope that it will be useful,
10 * but WITHOUT ANY WARRANTY; without even the implied warranty of
11 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
12 * Lesser General Public License for more details.
13 *
14 * You should have received a copy of the GNU Lesser General Public
15 * License along with this library; if not, write to the Free Software
16 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
17 */
18
19 #ifndef __VKD3D_H
20 #define __VKD3D_H
21
22 #include <vkd3d_types.h>
23
24 #ifndef VKD3D_NO_WIN32_TYPES
25 # include <vkd3d_windows.h>
26 # include <vkd3d_d3d12.h>
27 #endif /* VKD3D_NO_WIN32_TYPES */
28
29 #ifndef VKD3D_NO_VULKAN_H
30 # include <vulkan/vulkan.h>
31 #endif /* VKD3D_NO_VULKAN_H */
32
33 #ifdef __cplusplus
34 extern "C" {
35 #endif /* __cplusplus */
36
37 enum vkd3d_structure_type
38 {
39     VKD3D_STRUCTURE_TYPE_INSTANCE_CREATE_INFO,
40     VKD3D_STRUCTURE_TYPE_DEVICE_CREATE_INFO,
41     VKD3D_STRUCTURE_TYPE_IMAGE_RESOURCE_CREATE_INFO,
42     VKD3D_STRUCTURE_TYPE_OPTIONAL_INSTANCE_EXTENSIONS_INFO,
43     VKD3D_STRUCTURE_TYPE_OPTIONAL_DEVICE_EXTENSIONS_INFO,
44     VKD3D_STRUCTURE_TYPE_APPLICATION_INFO,
45     VKD3D_STRUCTURE_TYPE_HOST_TIME_DOMAIN_INFO,
46     VKD3D_FORCE_32_BIT_ENUM(VKD3D_STRUCTURE_TYPE),
47 };
48
49 enum vkd3d_api_version
50 {
51     VKD3D_API_VERSION_1_0,
52     VKD3D_API_VERSION_1_1,
53     VKD3D_API_VERSION_1_2,
54     VKD3D_API_VERSION_1_3,
55     VKD3D_API_VERSION_1_4,
56     VKD3D_API_VERSION_1_5,
57     VKD3D_API_VERSION_1_6,
58     VKD3D_API_VERSION_1_7,
59     VKD3D_API_VERSION_1_8,
60     VKD3D_API_VERSION_1_9,
61     VKD3D_API_VERSION_1_10,
62     VKD3D_API_VERSION_1_11,
63     VKD3D_API_VERSION_1_12,
64     VKD3D_API_VERSION_1_13,
65     VKD3D_API_VERSION_1_14,
66     VKD3D_API_VERSION_1_15,
67     VKD3D_FORCE_32_BIT_ENUM(VKD3D_API_VERSION),
68 };
69

```

```

107 typedef HRESULT (*PFN_vkd3d_signal_event)(HANDLE event);
108
109 typedef void * (*PFN_vkd3d_thread)(void *data);
110
111 typedef void * (*PFN_vkd3d_create_thread)(PFN_vkd3d_thread thread_main, void *data);
112 typedef HRESULT (*PFN_vkd3d_join_thread)(void *thread);
113
114 struct vkd3d_instance;
115
116 struct vkd3d_instance_create_info
117 {
118     enum vkd3d_structure_type type;
119     const void *next;
120
121     PFN_vkd3d_signal_event pfn_signal_event;
122     PFN_vkd3d_create_thread pfn_create_thread;
123     PFN_vkd3d_join_thread pfn_join_thread;
124     size_t wchar_size;
125
126     PFN_vkGetInstanceProcAddr pfn_vkGetInstanceProcAddr;
127
128     const char * const *instance_extensions;
129     uint32_t instance_extension_count;
130 };
131
132 struct vkd3d_optional_instance_extensions_info
133 {
134     enum vkd3d_structure_type type;
135     const void *next;
136
137     const char * const *extensions;
138     uint32_t extension_count;
139 };
140
141 struct vkd3d_application_info
142 {
143     enum vkd3d_structure_type type;
144     const void *next;
145
146     const char *application_name;
147     uint32_t application_version;
148
149     const char *engine_name;
150     uint32_t engine_version;
151
152     enum vkd3d_api_version api_version;
153 };
154
155 struct vkd3d_host_time_domain_info
156 {
157     enum vkd3d_structure_type type;
158     const void *next;
159
160     uint64_t ticks_per_second;
161 };
162
163 struct vkd3d_device_create_info
164 {
165     enum vkd3d_structure_type type;
166     const void *next;
167
168     D3D_FEATURE_LEVEL minimum_feature_level;
169
170     struct vkd3d_instance *instance;
171     const struct vkd3d_instance_create_info *instance_create_info;
172
173     VkPhysicalDevice vk_physical_device;
174
175     const char * const *device_extensions;
176     uint32_t device_extension_count;
177
178     IUnknown *parent;
179     LUID adapter_luid;
180 };
181
182 struct vkd3d_optional_device_extensions_info
183 {
184     enum vkd3d_structure_type type;
185     const void *next;
186
187     const char * const *extensions;
188     uint32_t extension_count;
189 };
190
191 #define VKD3D_RESOURCE_INITIAL_STATE_TRANSITION 0x00000001
192 #define VKD3D_RESOURCE_PRESENT_STATE_TRANSITION 0x00000002
193

```

```

337 struct vkd3d_image_resource_create_info
338 {
339     enum vkd3d_structure_type type;
340     const void *next;
341
342     VkImage vk_image;
343     D3D12_RESOURCE_DESC desc;
344     unsigned int flags;
345     D3D12_RESOURCE_STATES present_state;
346 };
347
348 #ifdef LIBVKD3D_SOURCE
349 # define VKD3D_API VKD3D_EXPORT
350 #else
351 # define VKD3D_API VKD3D_IMPORT
352 #endif
353
354 #ifndef VKD3D_NO_PROTOTYPES
355
356 VKD3D_API HRESULT vkd3d_create_instance(const struct vkd3d_instance_create_info *create_info,
357     struct vkd3d_instance **instance);
358 VKD3D_API ULONG vkd3d_instance_decref(struct vkd3d_instance *instance);
359 VKD3D_API VkInstance vkd3d_instance_get_vk_instance(struct vkd3d_instance *instance);
360 VKD3D_API ULONG vkd3d_instance_incref(struct vkd3d_instance *instance);
361
362 VKD3D_API HRESULT vkd3d_create_device(const struct vkd3d_device_create_info *create_info,
363     REFIID iid, void **device);
364 VKD3D_API IUnknown *vkd3d_get_device_parent(ID3D12Device *device);
365 VKD3D_API VkDevice vkd3d_get_vk_device(ID3D12Device *device);
366 VKD3D_API VkPhysicalDevice vkd3d_get_vk_physical_device(ID3D12Device *device);
367 VKD3D_API struct vkd3d_instance *vkd3d_instance_from_device(ID3D12Device *device);
368
369 VKD3D_API uint32_t vkd3d_get_vk_queue_family_index(ID3D12CommandQueue *queue);
370
371 VKD3D_API VkQueue vkd3d_acquire_vk_queue(ID3D12CommandQueue *queue);
372
373 VKD3D_API void vkd3d_release_vk_queue(ID3D12CommandQueue *queue);
374
375 VKD3D_API HRESULT vkd3d_create_image_resource(ID3D12Device *device,
376     const struct vkd3d_image_resource_create_info *create_info, ID3D12Resource **resource);
377 VKD3D_API ULONG vkd3d_resource_decref(ID3D12Resource *resource);
378 VKD3D_API ULONG vkd3d_resource_incref(ID3D12Resource *resource);
379
380 VKD3D_API HRESULT vkd3d_serialize_root_signature(const D3D12_ROOT_SIGNATURE_DESC *desc,
381     D3D_ROOT_SIGNATURE_VERSION version, ID3DBlob **blob, ID3DBlob **error_blob);
382 VKD3D_API HRESULT vkd3d_create_root_signature_deserializer(const void *data, SIZE_T data_size,
383     REFIID iid, void **deserializer);
384
385 VKD3D_API VkFormat vkd3d_get_vk_format(DXGI_FORMAT format);
386
387 /* 1.1 */
388 VKD3D_API DXGI_FORMAT vkd3d_get_dxgi_format(VkFormat format);
389
390 /* 1.2 */
391 VKD3D_API HRESULT vkd3d_serialize_versioned_root_signature(const D3D12_VERSIONED_ROOT_SIGNATURE_DESC
392     *desc,
393     ID3DBlob **blob, ID3DBlob **error_blob);
394 VKD3D_API HRESULT vkd3d_create_versioned_root_signature_deserializer(const void *data, SIZE_T data_size,
395     REFIID iid, void **deserializer);
396
397 VKD3D_API void vkd3d_set_log_callback(PFN_vkd3d_log callback);
398
399 VKD3D_API HRESULT vkd3d_queue_signal_on_cpu(ID3D12CommandQueue *queue,
400     ID3D12Fence *fence, uint64_t value);
401
402 #endif /* VKD3D_NO_PROTOTYPES */
403
404 /*
405  * Function pointer typedefs for vkd3d functions.
406  */
407
408 typedef HRESULT (*PFN_vkd3d_create_instance)(const struct vkd3d_instance_create_info *create_info,
409     struct vkd3d_instance **instance);
410 typedef ULONG (*PFN_vkd3d_instance_decref)(struct vkd3d_instance *instance);
411 typedef VkInstance (*PFN_vkd3d_instance_get_vk_instance)(struct vkd3d_instance *instance);
412 typedef ULONG (*PFN_vkd3d_instance_incref)(struct vkd3d_instance *instance);
413
414 typedef HRESULT (*PFN_vkd3d_create_device)(const struct vkd3d_device_create_info *create_info,
415     REFIID iid, void **device);
416 typedef IUnknown * (*PFN_vkd3d_get_device_parent)(ID3D12Device *device);
417 typedef VkDevice (*PFN_vkd3d_get_vk_device)(ID3D12Device *device);
418 typedef VkPhysicalDevice (*PFN_vkd3d_get_vk_physical_device)(ID3D12Device *device);
419 typedef struct vkd3d_instance * (*PFN_vkd3d_instance_from_device)(ID3D12Device *device);
420
421 typedef uint32_t (*PFN_vkd3d_get_vk_queue_family_index)(ID3D12CommandQueue *queue);
422 typedef VkQueue (*PFN_vkd3d_acquire_vk_queue)(ID3D12CommandQueue *queue);
423 typedef void (*PFN_vkd3d_release_vk_queue)(ID3D12CommandQueue *queue);
424

```

```

511 typedef HRESULT (*PFN_vk3d_create_image_resource)(ID3D12Device *device,
512     const struct vk3d_image_resource_create_info *create_info, ID3D12Resource **resource);
513 typedef ULONG (*PFN_vk3d_resource_decref)(ID3D12Resource *resource);
514 typedef ULONG (*PFN_vk3d_resource_incref)(ID3D12Resource *resource);
515
516 typedef HRESULT (*PFN_vk3d_serialize_root_signature)(const D3D12_ROOT_SIGNATURE_DESC *desc,
517     D3D_ROOT_SIGNATURE_VERSION version, ID3DBlob **blob, ID3DBlob **error_blob);
518 typedef HRESULT (*PFN_vk3d_create_root_signature_deserializer)(const void *data, SIZE_T data_size,
519     REFIID iid, void **deserializer);
520
521 typedef VkFormat (*PFN_vk3d_get_vk_format)(DXGI_FORMAT format);
522
523 /* 1.1 */
524 typedef DXGI_FORMAT (*PFN_vk3d_get_dxgi_format)(VkFormat format);
525
526 /* 1.2 */
527 typedef HRESULT (*PFN_vk3d_serialize_versioned_root_signature)(const
528     D3D12_VERSIONED_ROOT_SIGNATURE_DESC *desc,
529     ID3DBlob **blob, ID3DBlob **error_blob);
530 typedef HRESULT (*PFN_vk3d_create_versioned_root_signature_deserializer)(const void *data, SIZE_T
531     data_size,
532     REFIID iid, void **deserializer);
533
534 typedef void (*PFN_vk3d_set_log_callback)(PFN_vk3d_log callback);
535
536 typedef HRESULT (*PFN_vk3d_queue_signal_on_cpu)(ID3D12CommandQueue *queue,
537     ID3D12Fence *fence, uint64_t value);
538
539 #ifdef __cplusplus
540 }
541 #endif /* __cplusplus */
542
543 #endif /* __VKD3D_H */

```

## 4.3 /builds/nsivov/vkd3d/include/vkd3d\_shader.h File Reference

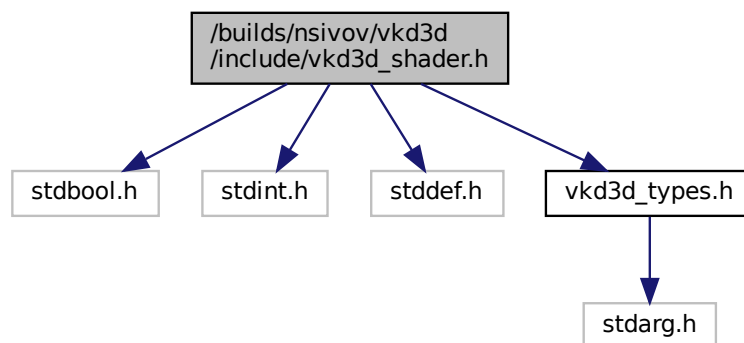
This file contains definitions for the vk3d-shader library.

```

#include <stdbool.h>
#include <stdint.h>
#include <stddef.h>
#include <vk3d_types.h>

```

Include dependency graph for vk3d\_shader.h:



## Data Structures

- struct [vk3d\\_shader\\_compile\\_option](#)

- Various settings which may affect shader compilation or scanning, passed as part of struct [vk3d\\_shader\\_compile\\_info](#).

  - struct [vk3d\\_shader\\_code](#)

A generic structure containing a GPU shader, in text or byte-code format.
  - struct [vk3d\\_shader\\_descriptor\\_binding](#)

A common structure describing the bind point of a descriptor or descriptor array in the target environment.
  - struct [vk3d\\_shader\\_parameter\\_immediate\\_constant](#)

The value of an immediate constant parameter, used in struct [vk3d\\_shader\\_parameter](#).
  - struct [vk3d\\_shader\\_parameter\\_immediate\\_constant1](#)

The value of an immediate constant parameter, used in struct [vk3d\\_shader\\_parameter1](#).
  - struct [vk3d\\_shader\\_parameter\\_specialization\\_constant](#)

The linkage of a specialization constant parameter, used in struct [vk3d\\_shader\\_parameter](#) and struct [vk3d\\_shader\\_parameter1](#).
  - struct [vk3d\\_shader\\_parameter\\_buffer](#)

The linkage of a parameter specified through a uniform buffer, used in struct [vk3d\\_shader\\_parameter1](#).
  - struct [vk3d\\_shader\\_parameter](#)

An individual shader parameter.
  - struct [vk3d\\_shader\\_parameter1](#)

An individual shader parameter.
  - struct [vk3d\\_shader\\_resource\\_binding](#)

Describes the mapping of a single resource or resource array to its binding point in the target environment.
  - struct [vk3d\\_shader\\_combined\\_resource\\_sampler](#)

Describes the mapping of a Direct3D resource-sampler pair to a combined sampler (i.e.
  - struct [vk3d\\_shader\\_uav\\_counter\\_binding](#)

Describes the mapping of a single Direct3D UAV counter.
  - struct [vk3d\\_shader\\_push\\_constant\\_buffer](#)

Describes the mapping of a Direct3D constant buffer to a range of push constants in the target environment.
  - struct [vk3d\\_shader\\_interface\\_info](#)

A chained structure describing the interface between a compiled shader and the target environment.
  - struct [vk3d\\_shader\\_transform\\_feedback\\_element](#)
  - struct [vk3d\\_shader\\_transform\\_feedback\\_info](#)
  - struct [vk3d\\_shader\\_descriptor\\_offset](#)
  - struct [vk3d\\_shader\\_descriptor\\_offset\\_info](#)

A chained structure containing descriptor offsets.
  - struct [vk3d\\_shader\\_compile\\_info](#)

A chained structure containing compilation parameters.
  - struct [vk3d\\_shader\\_spirv\\_target\\_info](#)
  - struct [vk3d\\_shader\\_spirv\\_domain\\_shader\\_target\\_info](#)
  - struct [vk3d\\_shader\\_macro](#)

A single preprocessor macro, passed as part of struct [vk3d\\_shader\\_preprocess\\_info](#).
  - struct [vk3d\\_shader\\_preprocess\\_info](#)

A chained structure containing preprocessing parameters.
  - struct [vk3d\\_shader\\_hlsl\\_source\\_info](#)

A chained structure containing HLSL compilation parameters.
  - struct [vk3d\\_shader\\_static\\_sampler\\_desc](#)
  - struct [vk3d\\_shader\\_descriptor\\_range](#)
  - struct [vk3d\\_shader\\_root\\_descriptor\\_table](#)
  - struct [vk3d\\_shader\\_root\\_constants](#)
  - struct [vk3d\\_shader\\_root\\_descriptor](#)
  - struct [vk3d\\_shader\\_root\\_parameter](#)
  - struct [vk3d\\_shader\\_root\\_signature\\_desc](#)
  - struct [vk3d\\_shader\\_descriptor\\_range1](#)
  - struct [vk3d\\_shader\\_root\\_descriptor\\_table1](#)

- struct [vkd3d\\_shader\\_root\\_descriptor1](#)
- struct [vkd3d\\_shader\\_root\\_parameter1](#)
- struct [vkd3d\\_shader\\_root\\_signature\\_desc1](#)
- struct [vkd3d\\_shader\\_versioned\\_root\\_signature\\_desc](#)
- struct [vkd3d\\_shader\\_descriptor\\_info](#)  
*Describes a single shader descriptor; returned as part of struct [vkd3d\\_shader\\_scan\\_descriptor\\_info](#).*
- struct [vkd3d\\_shader\\_scan\\_descriptor\\_info](#)  
*A chained structure enumerating the descriptors declared by a shader.*
- struct [vkd3d\\_shader\\_combined\\_resource\\_sampler\\_info](#)  
*This structure describes a single resource-sampler pair.*
- struct [vkd3d\\_shader\\_scan\\_combined\\_resource\\_sampler\\_info](#)  
*A chained structure describing the resource-sampler pairs used by a shader.*
- struct [vkd3d\\_shader\\_scan\\_hull\\_shader\\_tessellation\\_info](#)  
*A chained structure describing the tessellation information in a hull shader.*
- struct [vkd3d\\_shader\\_signature\\_element](#)  
*A single shader varying, returned as part of struct [vkd3d\\_shader\\_signature](#).*
- struct [vkd3d\\_shader\\_signature](#)  
*Description of a shader input or output signature.*
- struct [vkd3d\\_shader\\_dxbc\\_section\\_desc](#)  
*A description of a DXBC section.*
- struct [vkd3d\\_shader\\_dxbc\\_desc](#)  
*A description of a DXBC blob, as returned by [vkd3d\\_shader\\_parse\\_dxbc\(\)](#).*
- struct [vkd3d\\_shader\\_scan\\_signature\\_info](#)  
*A chained structure containing descriptions of shader inputs and outputs.*
- struct [vkd3d\\_shader\\_varying\\_map](#)  
*Describes the mapping of a output varying register in a shader stage, to an input varying register in the following shader stage.*
- struct [vkd3d\\_shader\\_varying\\_map\\_info](#)  
*A chained structure which describes how output varyings in this shader stage should be mapped to input varyings in the next stage.*
- struct [vkd3d\\_shader\\_parameter\\_info](#)  
*Interface information regarding a builtin shader parameter.*

## Macros

- #define **VKD3D\_SHADER\_DUMMY\_SAMPLER\_INDEX** ~0u
- #define [VKD3D\\_SHADER\\_SWIZZLE\\_MASK](#) (0xffu)  
*A mask selecting one component from a vkd3d-shader swizzle.*
- #define **VKD3D\_SHADER\_SWIZZLE\_SHIFT**(idx) (8u \* (idx))  
*The offset, in bits, of the nth parameter of a vkd3d-shader swizzle.*
- #define [VKD3D\\_SHADER\\_SWIZZLE](#)(x, y, z, w)  
*A helper macro which returns a vkd3d-shader swizzle with the given components.*
- #define **VKD3D\_SHADER\_NO\_SWIZZLE** [VKD3D\\_SHADER\\_SWIZZLE](#)(X, Y, Z, W)  
*The identity swizzle ".xyzw".*
- #define **VKD3D\_SHADER\_API** VKD3D\_IMPORT

## Typedefs

- typedef int(\* [PFN\\_vk3d\\_shader\\_open\\_include](#)) (const char \*filename, bool local, const char \*parent\_data, void \*context, struct [vk3d\\_shader\\_code](#) \*out)  
*Type of a callback function which will be used to open preprocessor includes.*
- typedef void(\* [PFN\\_vk3d\\_shader\\_close\\_include](#)) (const struct [vk3d\\_shader\\_code](#) \*code, void \*context)  
*Type of a callback function which will be used to close preprocessor includes.*
- typedef const char \*(\* [PFN\\_vk3d\\_shader\\_get\\_version](#)) (unsigned int \*major, unsigned int \*minor)  
*Type of [vk3d\\_shader\\_get\\_version\(\)](#).*
- typedef enum [vk3d\\_shader\\_source\\_type](#) \*(\* [PFN\\_vk3d\\_shader\\_get\\_supported\\_source\\_types](#)) (unsigned int \*count)  
*Type of [vk3d\\_shader\\_get\\_supported\\_source\\_types\(\)](#).*
- typedef enum [vk3d\\_shader\\_target\\_type](#) \*(\* [PFN\\_vk3d\\_shader\\_get\\_supported\\_target\\_types](#)) (enum [vk3d\\_shader\\_source\\_type](#) source\_type, unsigned int \*count)  
*Type of [vk3d\\_shader\\_get\\_supported\\_target\\_types\(\)](#).*
- typedef int(\* [PFN\\_vk3d\\_shader\\_compile](#)) (const struct [vk3d\\_shader\\_compile\\_info](#) \*compile\_info, struct [vk3d\\_shader\\_code](#) \*out, char \*\*messages)  
*Type of [vk3d\\_shader\\_compile\(\)](#).*
- typedef void(\* [PFN\\_vk3d\\_shader\\_free\\_messages](#)) (char \*messages)  
*Type of [vk3d\\_shader\\_free\\_messages\(\)](#).*
- typedef void(\* [PFN\\_vk3d\\_shader\\_free\\_shader\\_code](#)) (struct [vk3d\\_shader\\_code](#) \*code)  
*Type of [vk3d\\_shader\\_free\\_shader\\_code\(\)](#).*
- typedef int(\* [PFN\\_vk3d\\_shader\\_parse\\_root\\_signature](#)) (const struct [vk3d\\_shader\\_code](#) \*dxbc, struct [vk3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*root\_signature, char \*\*messages)  
*Type of [vk3d\\_shader\\_parse\\_root\\_signature\(\)](#).*
- typedef void(\* [PFN\\_vk3d\\_shader\\_free\\_root\\_signature](#)) (struct [vk3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*root\_signature)  
*Type of [vk3d\\_shader\\_free\\_root\\_signature\(\)](#).*
- typedef int(\* [PFN\\_vk3d\\_shader\\_serialize\\_root\\_signature](#)) (const struct [vk3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*root\_signature, struct [vk3d\\_shader\\_code](#) \*dxbc, char \*\*messages)  
*Type of [vk3d\\_shader\\_serialize\\_root\\_signature\(\)](#).*
- typedef int(\* [PFN\\_vk3d\\_shader\\_convert\\_root\\_signature](#)) (struct [vk3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*dst, enum [vk3d\\_shader\\_root\\_signature\\_version](#) version, const struct [vk3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*src)  
*Type of [vk3d\\_shader\\_convert\\_root\\_signature\(\)](#).*
- typedef int(\* [PFN\\_vk3d\\_shader\\_scan](#)) (const struct [vk3d\\_shader\\_compile\\_info](#) \*compile\_info, char \*\*messages)  
*Type of [vk3d\\_shader\\_scan\(\)](#).*
- typedef void(\* [PFN\\_vk3d\\_shader\\_free\\_scan\\_descriptor\\_info](#)) (struct [vk3d\\_shader\\_scan\\_descriptor\\_info](#) \*scan\_descriptor\_info)  
*Type of [vk3d\\_shader\\_free\\_scan\\_descriptor\\_info\(\)](#).*
- typedef int(\* [PFN\\_vk3d\\_shader\\_parse\\_input\\_signature](#)) (const struct [vk3d\\_shader\\_code](#) \*dxbc, struct [vk3d\\_shader\\_signature](#) \*signature, char \*\*messages)  
*Type of [vk3d\\_shader\\_parse\\_input\\_signature\(\)](#).*
- typedef struct [vk3d\\_shader\\_signature\\_element](#) \*(\* [PFN\\_vk3d\\_shader\\_find\\_signature\\_element](#)) (const struct [vk3d\\_shader\\_signature](#) \*signature, const char \*semantic\_name, unsigned int semantic\_index, unsigned int stream\_index)  
*Type of [vk3d\\_shader\\_find\\_signature\\_element\(\)](#).*
- typedef void(\* [PFN\\_vk3d\\_shader\\_free\\_shader\\_signature](#)) (struct [vk3d\\_shader\\_signature](#) \*signature)  
*Type of [vk3d\\_shader\\_free\\_shader\\_signature\(\)](#).*
- typedef void(\* [PFN\\_vk3d\\_shader\\_preprocess](#)) (struct [vk3d\\_shader\\_compile\\_info](#) \*compile\_info, struct [vk3d\\_shader\\_code](#) \*out, char \*\*messages)  
*Type of [vk3d\\_shader\\_preprocess\(\)](#).*

- typedef void(\* [PFN\\_vk3d\\_shader\\_set\\_log\\_callback](#)) (PFN\_vk3d\_log callback)  
Type of [vkd3d\\_shader\\_set\\_log\\_callback\(\)](#).
- typedef void(\* [PFN\\_vk3d\\_shader\\_free\\_dxbc](#)) (struct [vkd3d\\_shader\\_dxbc\\_desc](#) \*dxbc)  
Type of [vkd3d\\_shader\\_free\\_dxbc\(\)](#).
- typedef int(\* [PFN\\_vk3d\\_shader\\_parse\\_dxbc](#)) (const struct [vkd3d\\_shader\\_code](#) \*dxbc, uint32\_t flags, struct [vkd3d\\_shader\\_dxbc\\_desc](#) \*desc, char \*\*messages)  
Type of [vkd3d\\_shader\\_parse\\_dxbc\(\)](#).
- typedef int(\* [PFN\\_vk3d\\_shader\\_serialize\\_dxbc](#)) (size\_t section\_count, const struct [vkd3d\\_shader\\_dxbc\\_section\\_desc](#) \*sections, struct [vkd3d\\_shader\\_code](#) \*dxbc, char \*\*messages)  
Type of [vkd3d\\_shader\\_serialize\\_dxbc\(\)](#).
- typedef void(\* [PFN\\_vk3d\\_shader\\_build\\_varying\\_map](#)) (const struct [vkd3d\\_shader\\_signature](#) \*output ↵ signature, const struct [vkd3d\\_shader\\_signature](#) \*input\_signature, unsigned int \*count, struct [vkd3d\\_shader\\_varying\\_map](#) \*varyings)  
Type of [vkd3d\\_shader\\_build\\_varying\\_map\(\)](#).
- typedef void(\* [PFN\\_vk3d\\_shader\\_free\\_scan\\_signature\\_info](#)) (struct [vkd3d\\_shader\\_scan\\_signature\\_info](#) \*info)  
Type of [vkd3d\\_shader\\_free\\_scan\\_signature\\_info\(\)](#).
- typedef void(\* [PFN\\_vk3d\\_shader\\_free\\_scan\\_combined\\_resource\\_sampler\\_info](#)) (struct [vkd3d\\_shader\\_scan\\_combined\\_resource\\_sampler\\_info](#) \*info)  
Type of [vkd3d\\_shader\\_free\\_scan\\_combined\\_resource\\_sampler\\_info\(\)](#).

## Enumerations

- enum [vkd3d\\_shader\\_api\\_version](#) {  
VKD3D\_SHADER\_API\_VERSION\_1\_0 , VKD3D\_SHADER\_API\_VERSION\_1\_1 , VKD3D\_SHADER\_API\_VERSION\_1\_2 , VKD3D\_SHADER\_API\_VERSION\_1\_3 ,  
VKD3D\_SHADER\_API\_VERSION\_1\_4 , VKD3D\_SHADER\_API\_VERSION\_1\_5 , VKD3D\_SHADER\_API\_VERSION\_1\_6 , VKD3D\_SHADER\_API\_VERSION\_1\_7 ,  
VKD3D\_SHADER\_API\_VERSION\_1\_8 , VKD3D\_SHADER\_API\_VERSION\_1\_9 , VKD3D\_SHADER\_API\_VERSION\_1\_10 , VKD3D\_SHADER\_API\_VERSION\_1\_11 ,  
VKD3D\_SHADER\_API\_VERSION\_1\_12 , VKD3D\_SHADER\_API\_VERSION\_1\_13 , VKD3D\_SHADER\_API\_VERSION\_1\_14 , VKD3D\_SHADER\_API\_VERSION\_1\_15 ,  
VKD3D\_FORCE\_32\_BIT\_ENUM =(VKD3D\_SHADER\_API\_VERSION) }  
The type of a chained structure.
- enum [vkd3d\\_shader\\_structure\\_type](#) {  
VKD3D\_SHADER\_STRUCTURE\_TYPE\_COMPILE\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_INTERFACE\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_DESCRIPTOR\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_SPIRV\_DESCRIPTOR\_INFO ,  
VKD3D\_SHADER\_STRUCTURE\_TYPE\_SPIRV\_TARGET\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_TRANSFORM\_FEEDBACK\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_HLSL\_SOURCE\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_PREPROCESSOR\_INFO ,  
VKD3D\_SHADER\_STRUCTURE\_TYPE\_DESCRIPTOR\_OFFSET\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_SIGNATURE\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_VARYING\_MAP\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_COMBINED\_RESOURCE\_SAMPLER\_INFO ,  
VKD3D\_SHADER\_STRUCTURE\_TYPE\_PARAMETER\_INFO , VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_HULL\_SHADER\_INFO , VKD3D\_FORCE\_32\_BIT\_ENUM =(VKD3D\_SHADER\_API\_VERSION) }  
The type of a chained structure.
- enum [vkd3d\\_shader\\_compile\\_option\\_buffer\\_uav](#) { VKD3D\_SHADER\_COMPILE\_OPTION\_BUFFER\_UAV\_STORAGE\_TEXEL = 0x00000000 , VKD3D\_SHADER\_COMPILE\_OPTION\_BUFFER\_UAV\_STORAGE\_BUFFER = 0x00000001 , VKD3D\_FORCE\_32\_BIT\_ENUM =(VKD3D\_SHADER\_API\_VERSION) }  
Determines how buffer UAVs are stored.
- enum [vkd3d\\_shader\\_compile\\_option\\_typed\\_uav](#) { VKD3D\_SHADER\_COMPILE\_OPTION\_TYPED\_UAV\_READ\_FORMAT\_R32\_UINT = 0x00000000 , VKD3D\_SHADER\_COMPILE\_OPTION\_TYPED\_UAV\_READ\_FORMAT\_UNKNOWN = 0x00000001 , VKD3D\_FORCE\_32\_BIT\_ENUM =(VKD3D\_SHADER\_API\_VERSION) }  
Determines how typed UAVs are declared.

- enum `vkd3d_shader_compile_option_formatting_flags` {  
`VKD3D_SHADER_COMPILE_OPTION_FORMATTING_NONE` = 0x00000000 , `VKD3D_SHADER_COMPILE_OPTION_FORMATTING_COLOUR` = 0x00000001 , `VKD3D_SHADER_COMPILE_OPTION_FORMATTING_INDENT` = 0x00000002 , `VKD3D_SHADER_COMPILE_OPTION_FORMATTING_OFFSETS` = 0x00000004 ,  
`VKD3D_SHADER_COMPILE_OPTION_FORMATTING_HEADER` = 0x00000008 , `VKD3D_SHADER_COMPILE_OPTION_FORMATTING_RAW_IDS` = 0x00000010 , `VKD3D_SHADER_COMPILE_OPTION_FORMATTING_IO_SIZE` = 0x00000020 , `VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }
- enum `vkd3d_shader_compile_option_pack_matrix_order` { `VKD3D_SHADER_COMPILE_OPTION_PACK_MATRIX_ROW_MAJOR` = 0x00000001 , `VKD3D_SHADER_COMPILE_OPTION_PACK_MATRIX_COLUMN_MAJOR` = 0x00000002 , `VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }

*Determines how matrices are stored.*

- enum `vkd3d_shader_compile_option_backward_compatibility` { `VKD3D_SHADER_COMPILE_OPTION_BACKCOMPAT_MAP` = 0x00000001 , `VKD3D_SHADER_COMPILE_OPTION_DOUBLE_AS_FLOAT_ALIAS` = 0x00000002 , `VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }

*Individual options to enable various backward compatibility features.*

- enum `vkd3d_shader_compile_option_fragment_coordinate_origin` { `VKD3D_SHADER_COMPILE_OPTION_FRAGMENT_COORDINATE_ORIGIN_LOWER_LEFT` = 0x00000000 , `VKD3D_SHADER_COMPILE_OPTION_FRAGMENT_COORDINATE_ORIGIN_UPPER_LEFT` = 0x00000001 , `VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }

*Determines the origin of fragment coordinates.*

- enum `vkd3d_shader_compile_option_feature_flags` { `VKD3D_SHADER_COMPILE_OPTION_FEATURE_INT64` = 0x00000001 , `VKD3D_SHADER_COMPILE_OPTION_FEATURE_FLOAT64` = 0x00000002 , `VKD3D_SHADER_COMPILE_OPTION_FEATURE_ZERO_INITIALIZE_WORKGROUP_MEMORY` = 0x00000004 , `VKD3D_SHADER_COMPILE_OPTION_FEATURE_SHADER_EXTENSIONS` = 0x00000008 , `VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }

*Advertises feature availability.*

- enum `vkd3d_shader_parse_dxbc_flags` { `VKD3D_SHADER_PARSE_DXBC_IGNORE_CHECKSUM` = 0x00000001 , `VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }

*Flags for `vkd3d_shader_parse_dxbc()`.*

- enum `vkd3d_shader_compile_option_name` {  
`VKD3D_SHADER_COMPILE_OPTION_STRIP_DEBUG` = 0x00000001 , `VKD3D_SHADER_COMPILE_OPTION_BUFFER_USAGE` = 0x00000002 , `VKD3D_SHADER_COMPILE_OPTION_FORMATTING` = 0x00000003 , `VKD3D_SHADER_COMPILE_OPTION_TYPED_UAV` = 0x00000004 ,  
`VKD3D_SHADER_COMPILE_OPTION_TYPED_UAV` = 0x00000005 , `VKD3D_SHADER_COMPILE_OPTION_WRITE_TESS` = 0x00000006 , `VKD3D_SHADER_COMPILE_OPTION_PACK_MATRIX_ORDER` = 0x00000007 ,  
`VKD3D_SHADER_COMPILE_OPTION_BACKWARD_COMPATIBILITY` = 0x00000008 ,  
`VKD3D_SHADER_COMPILE_OPTION_FRAGMENT_COORDINATE_ORIGIN` = 0x00000009 , `VKD3D_SHADER_COMPILE_OPTION_CHILD_EFFECT` = 0x0000000a , `VKD3D_SHADER_COMPILE_OPTION_INCLUDE_EMPTY_BUFFERS_IN_EFFECTS` = 0x0000000b ,  
`VKD3D_SHADER_COMPILE_OPTION_INCLUDE_EMPTY_BUFFERS_IN_EFFECTS` = 0x0000000c ,  
`VKD3D_SHADER_COMPILE_OPTION_INCLUDE_EMPTY_BUFFERS_IN_EFFECTS` = 0x0000000d ,  
`VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }

- enum `vkd3d_shader_visibility` {  
`VKD3D_SHADER_VISIBILITY_ALL` = 0 , `VKD3D_SHADER_VISIBILITY_VERTEX` = 1 , `VKD3D_SHADER_VISIBILITY_HULL` = 2 , `VKD3D_SHADER_VISIBILITY_DOMAIN` = 3 ,  
`VKD3D_SHADER_VISIBILITY_GEOMETRY` = 4 , `VKD3D_SHADER_VISIBILITY_PIXEL` = 5 , `VKD3D_SHADER_VISIBILITY_COUNT` = 1000000000 , `VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }

*Describes which shader stages a resource is visible to.*

- enum `vkd3d_shader_descriptor_type` { `VKD3D_SHADER_DESCRIPTOR_TYPE_SRV` = 0x0 , `VKD3D_SHADER_DESCRIPTOR_TYPE_CBV` = 0x1 , `VKD3D_SHADER_DESCRIPTOR_TYPE_CBV` = 0x2 , `VKD3D_SHADER_DESCRIPTOR_TYPE_SAMPLER` = 0x3 , `VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }

*The type of a shader resource descriptor.*

- enum `vkd3d_shader_binding_flag` { `VKD3D_SHADER_BINDING_FLAG_BUFFER` = 0x00000001 , `VKD3D_SHADER_BINDING_FLAG_IMAGE` = 0x00000002 , `VKD3D_FORCE_32_BIT_ENUM`=(VKD3D\_SHADER\_API\_VERSION) }
- enum `vkd3d_shader_fog_fragment_mode` { `VKD3D_SHADER_FOG_FRAGMENT_NONE` = 0x0 , `VKD3D_SHADER_FOG_FRAGMENT_EXP` = 0x1 , `VKD3D_SHADER_FOG_FRAGMENT_EXP2` = 0x2 }

```
, VKD3D_SHADER_FOG_FRAGMENT_LINEAR = 0x3 , VKD3D_FORCE_32_BIT_ENUM =(VKD3D_SHADER_API_VERSION) }
```

*The factor used to interpolate the fragment output colour with fog.*

- enum `vkd3d_shader_fog_source` { `VKD3D_SHADER_FOG_SOURCE_FOG` = 0x0 , `VKD3D_SHADER_FOG_SOURCE_FOG` = 0x1 , `VKD3D_SHADER_FOG_SOURCE_Z` = 0x2 , `VKD3D_SHADER_FOG_SOURCE_W` = 0x3 , `VKD3D_FORCE_32_BIT_ENUM` =(VKD3D\_SHADER\_API\_VERSION) }

*The source of the fog varying output by a pre-rasterization shader.*

- enum `vkd3d_shader_parameter_type` { `VKD3D_SHADER_PARAMETER_TYPE_UNKNOWN` , `VKD3D_SHADER_PARAMETER_TYPE_SPECIALIZATION_CONSTANT` , `VKD3D_SHADER_PARAMETER_TYPE_BUFFER` , `VKD3D_FORCE_32_BIT_ENUM` =(VKD3D\_SHADER\_API\_VERSION) }

*The manner in which a parameter value is provided to the shader, used in struct `vkd3d_shader_parameter` and struct `vkd3d_shader_parameter1`.*

- enum `vkd3d_shader_parameter_data_type` { `VKD3D_SHADER_PARAMETER_DATA_TYPE_UNKNOWN` , `VKD3D_SHADER_PARAMETER_DATA_TYPE_UINT32` , `VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32` , `VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32_VEC4` , `VKD3D_FORCE_32_BIT_ENUM` =(VKD3D\_SHADER\_API\_VERSION) }

*The format of data provided to the shader, used in struct `vkd3d_shader_parameter` and struct `vkd3d_shader_parameter1`.*

- enum `vkd3d_shader_parameter_name` { `VKD3D_SHADER_PARAMETER_NAME_UNKNOWN` , `VKD3D_SHADER_PARAMETER_NAME_RASTERIZER_SAMPLE_C` , `VKD3D_SHADER_PARAMETER_NAME_ALPHA_TEST_FUNC` , `VKD3D_SHADER_PARAMETER_NAME_ALPHA_TEST_F` , `VKD3D_SHADER_PARAMETER_NAME_FLAT_INTERPOLATION` , `VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_0` , `VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_1` , `VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_2` , `VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_3` , `VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_4` , `VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_5` , `VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_6` , `VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_7` , `VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE` , `VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE_MAX` , `VKD3D_SHADER_PARAMETER_NAME_POINT_SPRITE` , `VKD3D_SHADER_PARAMETER_NAME_FOG_FRAGMENT_MODE` , `VKD3D_SHADER_PARAMETER_NAME_FOG_COLOR` , `VKD3D_SHADER_PARAMETER_NAME_FOG_END` , `VKD3D_SHADER_PARAMETER_NAME_FOG_SCALE` , `VKD3D_SHADER_PARAMETER_NAME_FOG_SOURCE` , `VKD3D_FORCE_32_BIT_ENUM` =(VKD3D\_SHADER\_API\_VERSION) }

*Names a specific shader parameter, used in struct `vkd3d_shader_parameter` and struct `vkd3d_shader_parameter1`.*

- enum `vkd3d_shader_d3dbc_constant_register` { `VKD3D_SHADER_D3DBC_FLOAT_CONSTANT_REGISTER` = 0x0 , `VKD3D_SHADER_D3DBC_INT_CONSTANT_REGISTER` = 0x1 , `VKD3D_SHADER_D3DBC_BOOL_CONSTANT_REGISTER` = 0x2 , `VKD3D_FORCE_32_BIT_ENUM` =(VKD3D\_SHADER\_API\_VERSION) }

*Symbolic register indices for mapping uniform constant register sets in legacy Direct3D bytecode to constant buffer views in the target environment.*

- enum `vkd3d_shader_source_type` { `VKD3D_SHADER_SOURCE_NONE` , `VKD3D_SHADER_SOURCE_DXBC_TPF` , `VKD3D_SHADER_SOURCE_HLSL` , `VKD3D_SHADER_SOURCE_D3D_BYTECODE` , `VKD3D_SHADER_SOURCE_DXBC_DXIL` , `VKD3D_SHADER_SOURCE_FX` , `VKD3D_FORCE_32_BIT_ENUM` =(VKD3D\_SHADER\_API\_VERSION) }

*The format of a shader to be compiled or scanned.*

- enum `vkd3d_shader_target_type` { `VKD3D_SHADER_TARGET_NONE` , `VKD3D_SHADER_TARGET_SPIRV_BINARY` , `VKD3D_SHADER_TARGET_SPIRV_TEXT` , `VKD3D_SHADER_TARGET_D3D_ASM` , `VKD3D_SHADER_TARGET_D3D_BYTECODE` , `VKD3D_SHADER_TARGET_DXBC_TPF` , `VKD3D_SHADER_TARGET_GLSL` , `VKD3D_SHADER_TARGET_FX` , `VKD3D_SHADER_TARGET_MSL` , `VKD3D_FORCE_32_BIT_ENUM` =(VKD3D\_SHADER\_API\_VERSION) }

*The output format of a compiled shader.*

- enum `vkd3d_shader_log_level` { `VKD3D_SHADER_LOG_NONE` , `VKD3D_SHADER_LOG_ERROR` , `VKD3D_SHADER_LOG_WARNING` , `VKD3D_SHADER_LOG_INFO` , `VKD3D_FORCE_32_BIT_ENUM` `=(VKD3D_SHADER_API_VERSION)` }

Describes the minimum severity of compilation messages returned by `vkd3d_shader_compile()` and similar functions.

- enum `vkd3d_shader_spirv_environment` { `VKD3D_SHADER_SPIRV_ENVIRONMENT_NONE` , `VKD3D_SHADER_SPIRV_ENVIRONMENT_OPENGL_4_5` , `VKD3D_SHADER_SPIRV_ENVIRONMENT_VULKAN_1_0` , `VKD3D_SHADER_SPIRV_ENVIRONMENT_VULKAN_1_1` , `VKD3D_FORCE_32_BIT_ENUM` `=(VKD3D_SHADER_API_VERSION)` }

- enum `vkd3d_shader_spirv_extension` { `VKD3D_SHADER_SPIRV_EXTENSION_NONE` , `VKD3D_SHADER_SPIRV_EXTENSION_EXT_DEMOTE_TO_HELPER_INVOCATION` , `VKD3D_SHADER_SPIRV_EXTENSION_EXT_DESCRIPTOR_INDEXING` , `VKD3D_SHADER_SPIRV_EXTENSION_EXT_STENCIL_EXPORT` , `VKD3D_SHADER_SPIRV_EXTENSION_EXT_VIEWPORT_INDEX_LAYER` , `VKD3D_SHADER_SPIRV_EXTENSION_EXT_VIEWPORT_INDEX_LAYER` , `VKD3D_FORCE_32_BIT_ENUM` `=(VKD3D_SHADER_API_VERSION)` }

- enum `vkd3d_shader_tessellator_output_primitive` { `VKD3D_SHADER_TESSELLATOR_OUTPUT_POINT` = 0x1 , `VKD3D_SHADER_TESSELLATOR_OUTPUT_LINE` = 0x2 , `VKD3D_SHADER_TESSELLATOR_OUTPUT_TRIANGLE_CW` = 0x3 , `VKD3D_SHADER_TESSELLATOR_OUTPUT_TRIANGLE_CCW` = 0x4 , `VKD3D_FORCE_32_BIT_ENUM` `=(VKD3D_SHADER_API_VERSION)` }

- enum `vkd3d_shader_tessellator_partitioning` { `VKD3D_SHADER_TESSELLATOR_PARTITIONING_INTEGER` = 0x1 , `VKD3D_SHADER_TESSELLATOR_PARTITIONING_POW2` = 0x2 , `VKD3D_SHADER_TESSELLATOR_PARTITIONING_FRACTIONAL_ODD` = 0x3 , `VKD3D_SHADER_TESSELLATOR_PARTITIONING_FRACTIONAL_EVEN` = 0x4 , `VKD3D_FORCE_32_BIT_ENUM` `=(VKD3D_SHADER_API_VERSION)` }

- enum `vkd3d_shader_filter` { `VKD3D_SHADER_FILTER_MIN_MAG_MIP_POINT` = 0x000 , `VKD3D_SHADER_FILTER_MIN_MAG_POINT_MIP_LINEAR` = 0x001 , `VKD3D_SHADER_FILTER_MIN_POINT_MAG_LINEAR_MIP_POINT` = 0x004 , `VKD3D_SHADER_FILTER_MIN_POINT_MAG_MIP_LINEAR` = 0x005 , `VKD3D_SHADER_FILTER_MIN_LINEAR_MAG_MIP_POINT` = 0x010 , `VKD3D_SHADER_FILTER_MIN_LINEAR_MAG_POINT_MIP_LINEAR` = 0x011 , `VKD3D_SHADER_FILTER_MIN_MAG_LINEAR_MIP_POINT` = 0x014 , `VKD3D_SHADER_FILTER_MIN_MAG_MIP_LINEAR` = 0x015 , `VKD3D_SHADER_FILTER_ANISOTROPIC` = 0x055 , `VKD3D_SHADER_FILTER_COMPARISON_MIN_MAG_MIP_POINT` = 0x080 , `VKD3D_SHADER_FILTER_COMPARISON_MIN_MAG_POINT_MIP_LINEAR` = 0x081 , `VKD3D_SHADER_FILTER_COMPARISON_MIN_POINT_MAG_LINEAR_MIP_POINT` = 0x084 , `VKD3D_SHADER_FILTER_COMPARISON_MIN_POINT_MAG_MIP_LINEAR` = 0x085 , `VKD3D_SHADER_FILTER_COMPARISON_MIN_LINEAR_MAG_MIP_POINT` = 0x090 , `VKD3D_SHADER_FILTER_COMPARISON_MIN_LINEAR_MAG_POINT_MIP_LINEAR` = 0x091 , `VKD3D_SHADER_FILTER_COMPARISON_MIN_MAG_LINEAR_MIP_POINT` = 0x094 , `VKD3D_SHADER_FILTER_COMPARISON_MIN_MAG_MIP_LINEAR` = 0x095 , `VKD3D_SHADER_FILTER_COMPARISON_ANISOTROPIC` = 0x0d5 , `VKD3D_SHADER_FILTER_MINIMUM_MIN_MAG_MIP_POINT` = 0x100 , `VKD3D_SHADER_FILTER_MINIMUM_MIN_MAG_POINT_MIP_LINEAR` = 0x101 , `VKD3D_SHADER_FILTER_MINIMUM_MIN_POINT_MAG_LINEAR_MIP_POINT` = 0x104 , `VKD3D_SHADER_FILTER_MINIMUM_MIN_POINT_MAG_MIP_LINEAR` = 0x105 , `VKD3D_SHADER_FILTER_MINIMUM_MIN_LINEAR_MAG_MIP_POINT` = 0x110 , `VKD3D_SHADER_FILTER_MINIMUM_MIN_LINEAR_MAG_POINT_MIP_LINEAR` = 0x111 , `VKD3D_SHADER_FILTER_MINIMUM_MIN_MAG_LINEAR_MIP_POINT` = 0x114 , `VKD3D_SHADER_FILTER_MINIMUM_MIN_MAG_MIP_LINEAR` = 0x115 , `VKD3D_SHADER_FILTER_MINIMUM_ANISOTROPIC` = 0x155 , `VKD3D_SHADER_FILTER_MAXIMUM_MIN_MAG_MIP_POINT` = 0x180 , `VKD3D_SHADER_FILTER_MAXIMUM_MIN_MAG_POINT_MIP_LINEAR` = 0x181 , `VKD3D_SHADER_FILTER_MAXIMUM_MIN_POINT_MAG_LINEAR_MIP_POINT` = 0x184 , `VKD3D_SHADER_FILTER_MAXIMUM_MIN_POINT_MAG_MIP_LINEAR` = 0x185 , `VKD3D_SHADER_FILTER_MAXIMUM_MIN_LINEAR_MAG_MIP_POINT` = 0x190 , `VKD3D_SHADER_FILTER_MAXIMUM_MIN_LINEAR_MAG_POINT_MIP_LINEAR` = 0x191 , `VKD3D_SHADER_FILTER_MAXIMUM_MIN_MAG_LINEAR_MIP_POINT` = 0x194 , `VKD3D_SHADER_FILTER_MAXIMUM_MIN_MAG_MIP_LINEAR` = 0x195 , `VKD3D_SHADER_FILTER_MAXIMUM_ANISOTROPIC` = 0x1d5 , `VKD3D_FORCE_32_BIT_ENUM` `=(VKD3D_SHADER_API_VERSION)` }
- enum `vkd3d_shader_texture_address_mode` { `VKD3D_SHADER_TEXTURE_ADDRESS_MODE_WRAP` = 0x1 , `VKD3D_SHADER_TEXTURE_ADDRESS_MODE_CLAMP` = 0x2 , `VKD3D_SHADER_TEXTURE_ADDRESS_MODE_MIRROR_CLAMP` = 0x3 , `VKD3D_SHADER_TEXTURE_ADDRESS_MODE_MIRROR_CLAMP_TO_EDGE` = 0x4 , `VKD3D_SHADER_TEXTURE_ADDRESS_MODE_MIRROR_CLAMP_TO_BORDER` = 0x5 , `VKD3D_SHADER_TEXTURE_ADDRESS_MODE_BORDER` = 0x6 , `VKD3D_SHADER_TEXTURE_ADDRESS_MODE_NONE` = 0x7 , `VKD3D_FORCE_32_BIT_ENUM` `=(VKD3D_SHADER_API_VERSION)` }

```

ADDRESS_MODE_MIRROR = 0x2 , VKD3D_SHADER_TEXTURE_ADDRESS_MODE_CLAMP = 0x3
, VKD3D_SHADER_TEXTURE_ADDRESS_MODE_BORDER = 0x4 ,
VKD3D_SHADER_TEXTURE_ADDRESS_MODE_MIRROR_ONCE = 0x5 , VKD3D_FORCE_32_BIT_↵
ENUM =(VKD3D_SHADER_API_VERSION) }
• enum vkd3d_shader_comparison_func {
VKD3D_SHADER_COMPARISON_FUNC_NEVER = 0x1 , VKD3D_SHADER_COMPARISON_FUNC_↵
_LESS = 0x2 , VKD3D_SHADER_COMPARISON_FUNC_EQUAL = 0x3 , VKD3D_SHADER_↵
COMPARISON_FUNC_LESS_EQUAL = 0x4 ,
VKD3D_SHADER_COMPARISON_FUNC_GREATER = 0x5 , VKD3D_SHADER_COMPARISON_FUNC_↵
_NOT_EQUAL = 0x6 , VKD3D_SHADER_COMPARISON_FUNC_GREATER_EQUAL = 0x7 , VKD3D_↵
SHADER_COMPARISON_FUNC_ALWAYS = 0x8 ,
VKD3D_FORCE_32_BIT_ENUM =(VKD3D_SHADER_API_VERSION) }
• enum vkd3d_shader_static_border_colour { VKD3D_SHADER_STATIC_BORDER_COLOUR_↵
TRANSPARENT_BLACK = 0x0 , VKD3D_SHADER_STATIC_BORDER_COLOUR_OPAQUE_BLACK
= 0x1 , VKD3D_SHADER_STATIC_BORDER_COLOUR_OPAQUE_WHITE = 0x2 , VKD3D_FORCE_32_↵
_BIT_ENUM =(VKD3D_SHADER_API_VERSION) }
• enum vkd3d_shader_root_parameter_type {
VKD3D_SHADER_ROOT_PARAMETER_TYPE_DESCRIPTOR_TABLE = 0x0 , VKD3D_SHADER_↵
ROOT_PARAMETER_TYPE_32BIT_CONSTANTS = 0x1 , VKD3D_SHADER_ROOT_PARAMETER_↵
TYPE_CBV = 0x2 , VKD3D_SHADER_ROOT_PARAMETER_TYPE_SRV = 0x3 ,
VKD3D_SHADER_ROOT_PARAMETER_TYPE_UAV = 0x4 , VKD3D_FORCE_32_BIT_ENUM =(VKD3_↵
D_SHADER_API_VERSION) }
• enum vkd3d_shader_root_signature_flags {
VKD3D_SHADER_ROOT_SIGNATURE_FLAG_NONE = 0x00 , VKD3D_SHADER_ROOT_SIGNATURE_↵
_FLAG_ALLOW_INPUT_ASSEMBLER_INPUT_LAYOUT = 0x01 , VKD3D_SHADER_ROOT_SIGNATURE_↵
_FLAG_DENY_VERTEX_SHADER_ROOT_ACCESS = 0x02 , VKD3D_SHADER_ROOT_SIGNATURE_↵
_FLAG_DENY_HULL_SHADER_ROOT_ACCESS = 0x04 ,
VKD3D_SHADER_ROOT_SIGNATURE_FLAG_DENY_DOMAIN_SHADER_ROOT_ACCESS = 0x08 ,
VKD3D_SHADER_ROOT_SIGNATURE_FLAG_DENY_GEOMETRY_SHADER_ROOT_ACCESS = 0x10
, VKD3D_SHADER_ROOT_SIGNATURE_FLAG_DENY_PIXEL_SHADER_ROOT_ACCESS = 0x20 ,
VKD3D_SHADER_ROOT_SIGNATURE_FLAG_ALLOW_STREAM_OUTPUT = 0x40 ,
VKD3D_FORCE_32_BIT_ENUM =(VKD3D_SHADER_API_VERSION) }
• enum vkd3d_shader_root_descriptor_flags { VKD3D_SHADER_ROOT_DESCRIPTOR_FLAG_NONE
= 0x0 , VKD3D_SHADER_ROOT_DESCRIPTOR_FLAG_DATA_VOLATILE = 0x2 , VKD3D_SHADER_↵
ROOT_DESCRIPTOR_FLAG_DATA_STATIC_WHILE_SET_AT_EXECUTE = 0x4 , VKD3D_SHADER_↵
_ROOT_DESCRIPTOR_FLAG_DATA_STATIC = 0x8 , VKD3D_FORCE_32_BIT_ENUM =(VKD3D_↵
SHADER_API_VERSION) }
• enum vkd3d_shader_descriptor_range_flags {
VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_NONE = 0x0 , VKD3D_SHADER_DESCRIPTOR_↵
RANGE_FLAG_DESCRIPTOR_VOLATILE = 0x1 , VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_↵
_DATA_VOLATILE = 0x2 , VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_DATA_STATIC_WHILE_↵
SET_AT_EXECUTE = 0x4 ,
VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_DATA_STATIC = 0x8 , VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_↵
= 0x10000 , VKD3D_FORCE_32_BIT_ENUM =(VKD3D_SHADER_API_VERSION) }
• enum vkd3d_shader_root_signature_version { VKD3D_SHADER_ROOT_SIGNATURE_VERSION_1_↵
0 = 0x1 , VKD3D_SHADER_ROOT_SIGNATURE_VERSION_1_1 = 0x2 , VKD3D_FORCE_32_BIT_ENUM
=(VKD3D_SHADER_API_VERSION) }
• enum vkd3d_shader_resource_type {
VKD3D_SHADER_RESOURCE_NONE = 0x0 , VKD3D_SHADER_RESOURCE_BUFFER = 0x1 ,
VKD3D_SHADER_RESOURCE_TEXTURE_1D = 0x2 , VKD3D_SHADER_RESOURCE_TEXTURE_2D
= 0x3 ,
VKD3D_SHADER_RESOURCE_TEXTURE_2DMS = 0x4 , VKD3D_SHADER_RESOURCE_TEXTURE_3D
= 0x5 , VKD3D_SHADER_RESOURCE_TEXTURE_CUBE = 0x6 , VKD3D_SHADER_RESOURCE_TEXTURE_1DARRAY
= 0x7 ,
VKD3D_SHADER_RESOURCE_TEXTURE_2DARRAY = 0x8 , VKD3D_SHADER_RESOURCE_TEXTURE_2DMSARRAY
= 0x9 , VKD3D_SHADER_RESOURCE_TEXTURE_CUBEARRAY = 0xa , VKD3D_FORCE_32_BIT_ENUM
=(VKD3D_SHADER_API_VERSION) }

```

*The type of a shader resource, returned as part of struct [vkd3d\\_shader\\_descriptor\\_info](#).*

- enum `vkd3d_shader_resource_data_type` {  
`VKD3D_SHADER_RESOURCE_DATA_UNORM` = 0x1 , `VKD3D_SHADER_RESOURCE_DATA_SNORM` = 0x2 , `VKD3D_SHADER_RESOURCE_DATA_INT` = 0x3 , `VKD3D_SHADER_RESOURCE_DATA_UINT` = 0x4 ,  
`VKD3D_SHADER_RESOURCE_DATA_FLOAT` = 0x5 , `VKD3D_SHADER_RESOURCE_DATA_MIXED` = 0x6 , `VKD3D_SHADER_RESOURCE_DATA_DOUBLE` = 0x7 , `VKD3D_SHADER_RESOURCE_DATA_CONTINUED` = 0x8 ,  
`VKD3D_FORCE_32_BIT_ENUM`=(`VKD3D_SHADER_API_VERSION`) }

The type of the data contained in a shader resource, returned as part of struct `vkd3d_shader_descriptor_info`.

- enum `vkd3d_shader_descriptor_info_flag` {  
`VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_UAV_COUNTER` = 0x00000001 , `VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_SAMPLER_COMPARISON_MODE` = 0x00000002 , `VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_UAV_ATOMICS` = 0x00000008 , `VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_RAW_BUFFER` = 0x00000010 , `VKD3D_FORCE_32_BIT_ENUM`=(`VKD3D_SHADER_API_VERSION`) }

Additional flags describing a shader descriptor, returned as part of struct `vkd3d_shader_descriptor_info`.

- enum `vkd3d_shader_component_type` {  
`VKD3D_SHADER_COMPONENT_VOID` = 0x0 , `VKD3D_SHADER_COMPONENT_UINT` = 0x1 , `VKD3D_SHADER_COMPONENT_INT` = 0x2 , `VKD3D_SHADER_COMPONENT_FLOAT` = 0x3 , `VKD3D_SHADER_COMPONENT_BOOL` = 0x4 , `VKD3D_SHADER_COMPONENT_DOUBLE` = 0x5 , `VKD3D_SHADER_COMPONENT_UINT64` = 0x6 , `VKD3D_SHADER_COMPONENT_INT64` = 0x7 , `VKD3D_SHADER_COMPONENT_FLOAT16` = 0x8 , `VKD3D_SHADER_COMPONENT_UINT16` = 0x9 , `VKD3D_SHADER_COMPONENT_INT16` = 0xa , `VKD3D_FORCE_32_BIT_ENUM`=(`VKD3D_SHADER_API_VERSION`) }

Data type of a shader varying, returned as part of struct `vkd3d_shader_signature_element`.

- enum `vkd3d_shader_sysval_semantic` {  
`VKD3D_SHADER_SV_NONE` = 0x00 , `VKD3D_SHADER_SV_POSITION` = 0x01 , `VKD3D_SHADER_SV_CLIP_DISTANCE` = 0x02 , `VKD3D_SHADER_SV_CULL_DISTANCE` = 0x03 , `VKD3D_SHADER_SV_RENDER_TARGET_ARRAY_INDEX` = 0x04 , `VKD3D_SHADER_SV_VIEWPORT_ARRAY_INDEX` = 0x05 , `VKD3D_SHADER_SV_VERTEX_ID` = 0x06 , `VKD3D_SHADER_SV_PRIMITIVE_ID` = 0x07 , `VKD3D_SHADER_SV_INSTANCE_ID` = 0x08 , `VKD3D_SHADER_SV_IS_FRONT_FACE` = 0x09 , `VKD3D_SHADER_SV_SAMPLE_INDEX` = 0x0a , `VKD3D_SHADER_SV_TESS_FACTOR_QUADEGE` = 0x0b , `VKD3D_SHADER_SV_TESS_FACTOR_QUADINT` = 0x0c , `VKD3D_SHADER_SV_TESS_FACTOR_TRIEDGE` = 0x0d , `VKD3D_SHADER_SV_TESS_FACTOR_TRIINT` = 0x0e , `VKD3D_SHADER_SV_TESS_FACTOR_LINEDET` = 0x0f , `VKD3D_SHADER_SV_TESS_FACTOR_LINEDEN` = 0x10 , `VKD3D_SHADER_SV_TARGET` = 0x40 , `VKD3D_SHADER_SV_DEPTH` = 0x41 , `VKD3D_SHADER_SV_COVERAGE` = 0x42 , `VKD3D_SHADER_SV_DEPTH_GREATER_EQUAL` = 0x43 , `VKD3D_SHADER_SV_DEPTH_LESS_EQUAL` = 0x44 , `VKD3D_SHADER_SV_STENCIL_REF` = 0x45 , `VKD3D_FORCE_32_BIT_ENUM`=(`VKD3D_SHADER_API_VERSION`) }

System value semantic, returned as part of struct `vkd3d_shader_signature`.

- enum `vkd3d_shader_minimum_precision` {  
`VKD3D_SHADER_MINIMUM_PRECISION_NONE` = 0 , `VKD3D_SHADER_MINIMUM_PRECISION_FLOAT_16` = 1 , `VKD3D_SHADER_MINIMUM_PRECISION_FIXED_8_2` = 2 , `VKD3D_SHADER_MINIMUM_PRECISION_INT_16` = 4 , `VKD3D_SHADER_MINIMUM_PRECISION_UINT_16` = 5 , `VKD3D_FORCE_32_BIT_ENUM`=(`VKD3D_SHADER_API_VERSION`) }

Minimum interpolation precision of a shader varying, returned as part of struct `vkd3d_shader_signature_element`.

- enum `vkd3d_shader_swizzle_component` { `VKD3D_SHADER_SWIZZLE_X` = 0x0 , `VKD3D_SHADER_SWIZZLE_Y` = 0x1 , `VKD3D_SHADER_SWIZZLE_Z` = 0x2 , `VKD3D_SHADER_SWIZZLE_W` = 0x3 , `VKD3D_FORCE_32_BIT_ENUM`=(`VKD3D_SHADER_API_VERSION`) }

Possible values for a single component of a vkd3d-shader swizzle.

## Functions

- static uint32\_t **vkd3d\_shader\_create\_swizzle** (enum [vkd3d\\_shader\\_swizzle\\_component](#) x, enum [vkd3d\\_shader\\_swizzle\\_component](#) y, enum [vkd3d\\_shader\\_swizzle\\_component](#) z, enum [vkd3d\\_shader\\_swizzle\\_component](#) w)  
*Build a vkd3d-shader swizzle with the given components.*
- VKD3D\_SHADER\_API const char \* **vkd3d\_shader\_get\_version** (unsigned int \*major, unsigned int \*minor)  
*Returns the current version of this library.*
- VKD3D\_SHADER\_API enum [vkd3d\\_shader\\_source\\_type](#) \* **vkd3d\_shader\_get\_supported\_source\_types** (unsigned int \*count)  
*Returns the source types supported, with any target type, by [vkd3d\\_shader\\_compile\(\)](#).*
- VKD3D\_SHADER\_API enum [vkd3d\\_shader\\_target\\_type](#) \* **vkd3d\_shader\_get\_supported\_target\_types** (enum [vkd3d\\_shader\\_source\\_type](#) source\_type, unsigned int \*count)  
*Returns the target types supported, with the given source type, by [vkd3d\\_shader\\_compile\(\)](#).*
- VKD3D\_SHADER\_API int **vkd3d\_shader\_compile** (const struct [vkd3d\\_shader\\_compile\\_info](#) \*compile\_info, struct [vkd3d\\_shader\\_code](#) \*out, char \*\*messages)  
*Transform a form of GPU shader source code or byte code into another form of source code or byte code.*
- VKD3D\_SHADER\_API void **vkd3d\_shader\_free\_messages** (char \*messages)  
*Free shader messages allocated by another vkd3d-shader function, such as [vkd3d\\_shader\\_compile\(\)](#).*
- VKD3D\_SHADER\_API void **vkd3d\_shader\_free\_shader\_code** (struct [vkd3d\\_shader\\_code](#) \*code)  
*Free shader code allocated by another vkd3d-shader function, such as [vkd3d\\_shader\\_compile\(\)](#).*
- VKD3D\_SHADER\_API int **vkd3d\_shader\_parse\_root\_signature** (const struct [vkd3d\\_shader\\_code](#) \*dxbc, struct [vkd3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*root\_signature, char \*\*messages)  
*Convert a byte code description of a shader root signature to a structural description which can be easily parsed by C code.*
- VKD3D\_SHADER\_API void **vkd3d\_shader\_free\_root\_signature** (struct [vkd3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*root\_signature)  
*Free a structural representation of a shader root signature allocated by [vkd3d\\_shader\\_convert\\_root\\_signature\(\)](#) or [vkd3d\\_shader\\_parse\\_root\\_signature\(\)](#).*
- VKD3D\_SHADER\_API int **vkd3d\_shader\_serialize\_root\_signature** (const struct [vkd3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*root\_signature, struct [vkd3d\\_shader\\_code](#) \*dxbc, char \*\*messages)  
*Convert a structural description of a shader root signature to a byte code format capable of being read by ID3D12↔Device::CreateRootSignature.*
- VKD3D\_SHADER\_API int **vkd3d\_shader\_convert\_root\_signature** (struct [vkd3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*dst, enum [vkd3d\\_shader\\_root\\_signature\\_version](#) version, const struct [vkd3d\\_shader\\_versioned\\_root\\_signature\\_desc](#) \*src)  
*Convert a structural representation of a root signature to a different version of structural representation.*
- VKD3D\_SHADER\_API int **vkd3d\_shader\_scan** (const struct [vkd3d\\_shader\\_compile\\_info](#) \*compile\_info, char \*\*messages)  
*Parse shader source code or byte code, returning various types of requested information.*
- VKD3D\_SHADER\_API void **vkd3d\_shader\_free\_scan\_descriptor\_info** (struct [vkd3d\\_shader\\_scan\\_descriptor\\_info](#) \*scan\_descriptor\_info)  
*Free members of struct [vkd3d\\_shader\\_scan\\_descriptor\\_info\(\)](#) allocated by [vkd3d\\_shader\\_scan\(\)](#).*
- VKD3D\_SHADER\_API int **vkd3d\_shader\_parse\_input\_signature** (const struct [vkd3d\\_shader\\_code](#) \*dxbc, struct [vkd3d\\_shader\\_signature](#) \*signature, char \*\*messages)  
*Read the input signature of a compiled DXBC shader, returning a structural description which can be easily parsed by C code.*
- VKD3D\_SHADER\_API struct [vkd3d\\_shader\\_signature\\_element](#) \* **vkd3d\_shader\_find\_signature\_element** (const struct [vkd3d\\_shader\\_signature](#) \*signature, const char \*semantic\_name, unsigned int semantic\_index, unsigned int stream\_index)  
*Find a single element of a parsed input signature.*
- VKD3D\_SHADER\_API void **vkd3d\_shader\_free\_shader\_signature** (struct [vkd3d\\_shader\\_signature](#) \*signature)  
*Free a structural representation of a shader input signature allocated by [vkd3d\\_shader\\_parse\\_input\\_signature\(\)](#).*

- VKD3D\_SHADER\_API int [vkd3d\\_shader\\_preprocess](#) (const struct [vkd3d\\_shader\\_compile\\_info](#) \*compile\_info, struct [vkd3d\\_shader\\_code](#) \*out, char \*\*messages)  
*Preprocess the given source code.*
- VKD3D\_SHADER\_API void [vkd3d\\_shader\\_set\\_log\\_callback](#) (PFN\_vkd3d\_log callback)  
*Set a callback to be called when vkd3d-shader outputs debug logging.*
- VKD3D\_SHADER\_API void [vkd3d\\_shader\\_free\\_dxbc](#) (struct [vkd3d\\_shader\\_dxbc\\_desc](#) \*dxbc)  
*Free the contents of a [vkd3d\\_shader\\_dxbc\\_desc](#) structure allocated by another vkd3d-shader function, such as [vkd3d\\_shader\\_parse\\_dxbc](#)().*
- VKD3D\_SHADER\_API int [vkd3d\\_shader\\_parse\\_dxbc](#) (const struct [vkd3d\\_shader\\_code](#) \*dxbc, uint32\_t flags, struct [vkd3d\\_shader\\_dxbc\\_desc](#) \*desc, char \*\*messages)  
*Parse a DXBC blob contained in a [vkd3d\\_shader\\_code](#) structure.*
- VKD3D\_SHADER\_API int [vkd3d\\_shader\\_serialize\\_dxbc](#) (size\_t section\_count, const struct [vkd3d\\_shader\\_dxbc\\_section\\_desc](#) \*sections, struct [vkd3d\\_shader\\_code](#) \*dxbc, char \*\*messages)  
*Serialize a DXBC description into a blob stored in a [vkd3d\\_shader\\_code](#) structure.*
- VKD3D\_SHADER\_API void [vkd3d\\_shader\\_free\\_scan\\_signature\\_info](#) (struct [vkd3d\\_shader\\_scan\\_signature\\_info](#) \*info)  
*Free members of struct [vkd3d\\_shader\\_scan\\_signature\\_info](#) allocated by [vkd3d\\_shader\\_scan](#)().*
- VKD3D\_SHADER\_API void [vkd3d\\_shader\\_build\\_varying\\_map](#) (const struct [vkd3d\\_shader\\_signature](#) \*output\_signature, const struct [vkd3d\\_shader\\_signature](#) \*input\_signature, unsigned int \*count, struct [vkd3d\\_shader\\_varying\\_map](#) \*varyings)  
*Build a mapping of output varyings in a shader stage to input varyings in the following shader stage.*
- VKD3D\_SHADER\_API void [vkd3d\\_shader\\_free\\_scan\\_combined\\_resource\\_sampler\\_info](#) (struct [vkd3d\\_shader\\_scan\\_combined\\_resource\\_sampler\\_info](#) \*info)  
*Free members of struct [vkd3d\\_shader\\_scan\\_combined\\_resource\\_sampler\\_info](#) allocated by [vkd3d\\_shader\\_scan](#)().*

### 4.3.1 Detailed Description

This file contains definitions for the vkd3d-shader library.

The vkd3d-shader library provides multiple utilities related to the compilation, transformation, and reflection of GPU shaders.

Since

1.2

### 4.3.2 Macro Definition Documentation

#### 4.3.2.1 VKD3D\_SHADER\_SWIZZLE

```
#define VKD3D_SHADER_SWIZZLE(  
    x,  
    y,  
    z,  
    w )
```

Value:

```
(VKD3D_SHADER_SWIZZLE_ ## x << VKD3D_SHADER_SWIZZLE_SHIFT(0) \  
 | VKD3D_SHADER_SWIZZLE_ ## y << VKD3D_SHADER_SWIZZLE_SHIFT(1) \  
 | VKD3D_SHADER_SWIZZLE_ ## z << VKD3D_SHADER_SWIZZLE_SHIFT(2) \  
 | VKD3D_SHADER_SWIZZLE_ ## w << VKD3D_SHADER_SWIZZLE_SHIFT(3) )
```

A helper macro which returns a vkd3d-shader swizzle with the given components.

The components are specified as the suffixes to members of [vkd3d\\_shader\\_swizzle\\_component](#). For example, the swizzle ".xwyy" can be represented as:

```
VKD3D_SHADER_SWIZZLE(X, W, Y, Y)
```

#### 4.3.2.2 VKD3D\_SHADER\_SWIZZLE\_MASK

```
#define VKD3D_SHADER_SWIZZLE_MASK (0xffu)
```

A mask selecting one component from a vkd3d-shader swizzle.

The component has type [vkd3d\\_shader\\_swizzle\\_component](#).

### 4.3.3 Typedef Documentation

#### 4.3.3.1 PFN\_vkd3d\_shader\_build\_varying\_map

```
typedef void(* PFN_vkd3d_shader_build_varying_map) (const struct vkd3d\_shader\_signature *output_↵  
_signature, const struct vkd3d\_shader\_signature *input_signature, unsigned int *count, struct  
vkd3d\_shader\_varying\_map *varyings)
```

Type of [vkd3d\\_shader\\_build\\_varying\\_map\(\)](#).

Since

1.9

#### 4.3.3.2 PFN\_vkd3d\_shader\_close\_include

```
typedef void(* PFN_vkd3d_shader_close_include) (const struct vkd3d\_shader\_code *code, void  
*context)
```

Type of a callback function which will be used to close preprocessor includes.

This callback function is passed as part of struct [vkd3d\\_shader\\_preprocess\\_info](#).

Parameters

<i>code</i>	Contents of the included file, which were allocated by the <a href="#">vkd3d_shader_preprocess_info.pfn_open_include</a> callback. The user must free them.
<i>context</i>	The user-defined pointer passed to struct <a href="#">vkd3d_shader_preprocess_info</a> .

#### 4.3.3.3 PFN\_vkd3d\_shader\_free\_dxbc

```
typedef void(* PFN_vkd3d_shader_free_dxbc) (struct vkd3d\_shader\_dxbc\_desc *dxbc)
```

Type of [vkd3d\\_shader\\_free\\_dxbc\(\)](#).

Since

1.7

#### 4.3.3.4 PFN\_vk3d\_shader\_free\_scan\_combined\_resource\_sampler\_info

```
typedef void(* PFN_vk3d_shader_free_scan_combined_resource_sampler_info) (struct vk3d_shader_scan_combined_resource_sampler_info *info)
```

Type of [vk3d\\_shader\\_free\\_scan\\_combined\\_resource\\_sampler\\_info\(\)](#).

Since

1.10

#### 4.3.3.5 PFN\_vk3d\_shader\_free\_scan\_signature\_info

```
typedef void(* PFN_vk3d_shader_free_scan_signature_info) (struct vk3d_shader_scan_signature_info *info)
```

Type of [vk3d\\_shader\\_free\\_scan\\_signature\\_info\(\)](#).

Since

1.9

#### 4.3.3.6 PFN\_vk3d\_shader\_open\_include

```
typedef int(* PFN_vk3d_shader_open_include) (const char *filename, bool local, const char *parent_data, void *context, struct vk3d_shader_code *out)
```

Type of a callback function which will be used to open preprocessor includes.

This callback function is passed as part of struct [vk3d\\_shader\\_preprocess\\_info](#).

If this function fails, vk3d-shader will emit a compilation error, and the *pfn\_close\_include* callback will not be called.

Parameters

<i>filename</i>	Unquoted string used as an argument to the #include directive.
<i>local</i>	Whether the #include directive is requesting a local (i.e. double-quoted) or system (i.e. angle-bracketed) include.
<i>parent_data</i>	Unprocessed source code of the file in which this #include directive is evaluated. This parameter may be NULL.
<i>context</i>	The user-defined pointer passed to struct <a href="#">vk3d_shader_preprocess_info</a> .
<i>out</i>	Output location for the full contents of the included file. The code need not be allocated using standard vk3d functions, but must remain valid until the corresponding call to <i>pfn_close_include</i> . If this function fails, the contents of this parameter are ignored.

#### Returns

A member of [vkd3d\\_result](#).

#### 4.3.3.7 PFN\_vkd3d\_shader\_parse\_dxbc

```
typedef int(* PFN_vkd3d_shader_parse_dxbc) (const struct vkd3d\_shader\_code *dxbc, uint32_t flags, struct vkd3d\_shader\_dxbc\_desc *desc, char **messages)
```

Type of [vkd3d\\_shader\\_parse\\_dxbc\(\)](#).

#### Since

1.7

#### 4.3.3.8 PFN\_vkd3d\_shader\_preprocess

```
typedef void(* PFN_vkd3d_shader_preprocess) (struct vkd3d\_shader\_compile\_info *compile_info, struct vkd3d\_shader\_code *out, char **messages)
```

Type of [vkd3d\\_shader\\_preprocess\(\)](#).

#### Since

1.3

#### 4.3.3.9 PFN\_vkd3d\_shader\_serialize\_dxbc

```
typedef int(* PFN_vkd3d_shader_serialize_dxbc) (size_t section_count, const struct vkd3d\_shader\_dxbc\_section\_desc *sections, struct vkd3d\_shader\_code *dxbc, char **messages)
```

Type of [vkd3d\\_shader\\_serialize\\_dxbc\(\)](#).

#### Since

1.7

#### 4.3.3.10 PFN\_vkD3d\_shader\_set\_log\_callback

```
typedef void(* PFN_vkD3d_shader_set_log_callback) (PFN_vkD3d_log callback)
```

Type of [vkD3d\\_shader\\_set\\_log\\_callback\(\)](#).

Since

1.4

### 4.3.4 Enumeration Type Documentation

#### 4.3.4.1 vkD3d\_shader\_api\_version

```
enum vkD3d_shader_api_version
```

Since

1.3

#### 4.3.4.2 vkD3d\_shader\_compile\_option\_backward\_compatibility

```
enum vkD3d_shader_compile_option_backward_compatibility
```

Individual options to enable various backward compatibility features.

Since

1.10

Enumerator

VKD3D_SHADER_COMPILE_OPTION_↔ BACKCOMPAT_MAP_SEMANTIC_NAMES	Causes compiler to convert SM1-3 semantics to corresponding System Value semantics, when compiling HLSL sources for SM4+ targets. This option does the following conversions: <ul style="list-style-type: none"> <li>• POSITION to SV_Position for vertex shader outputs, pixel shader inputs, and geometry shader inputs and outputs;</li> <li>• COLORN to SV_TargetN for pixel shader outputs;</li> <li>• DEPTH to SV_Depth for pixel shader outputs.</li> </ul>
VKD3D_SHADER_COMPILE_OPTION_DOUBLE↔ _AS_FLOAT_ALIAS	Causes 'double' to behave as an alias for 'float'. This option only applies to HLSL sources with shader model 1-3 target profiles. Without this option using the 'double' type produces compilation errors in these target profiles. This option is disabled by default.

#### 4.3.4.3 vkd3d\_shader\_compile\_option\_buffer\_uav

enum `vkd3d_shader_compile_option_buffer_uav`

Determines how buffer UAVs are stored.

This also affects UAV counters in Vulkan environments. In OpenGL environments, atomic counter buffers are always used for UAV counters.

##### Enumerator

VKD3D_SHADER_COMPILE_OPTION_BUFFER_UAV_STORAGE_TEXEL_BUFFER↔	Use buffer textures for buffer UAVs. This is the default value.
VKD3D_SHADER_COMPILE_OPTION_BUFFER_UAV_STORAGE_BUFFER↔	Use storage buffers for buffer UAVs.

#### 4.3.4.4 vkd3d\_shader\_compile\_option\_feature\_flags

enum `vkd3d_shader_compile_option_feature_flags`

Advertises feature availability.

##### Since

1.11

##### Enumerator

VKD3D_SHADER_COMPILE_OPTION_FEATURE_64_BIT_INTEGER↔	The SPIR-V target environment supports 64-bit integer types. This corresponds to the "shaderInt64" feature in the Vulkan API, and the "GL_ARB_gpu_shader_int64" extension in the OpenGL API.
VKD3D_SHADER_COMPILE_OPTION_FEATURE_64_BIT_FLOAT↔	The SPIR-V target environment supports 64-bit floating-point types. This corresponds to the "shaderFloat64" feature in the Vulkan API, and the "GL_ARB_gpu_shader_fp64" extension in the OpenGL API.

## Enumerator

VKD3D_SHADER_COMPILE_OPTION_FEATURE↔ _WAVE_OPS	<p>The SPIR-V target environment supports wave operations. This flag is valid only in VKD3D_↔SHADER_SPIRV_ENVIRONMENT_VULKAN_1_1 or greater, and corresponds to the following minimum requirements in <code>VkPhysicalDeviceSubgroupProperties</code>:</p> <ul style="list-style-type: none"> <li>• <code>subgroupSize</code> <math>\geq 4</math>.</li> <li>• <code>supportedOperations</code> has BASIC, VOTE, ARITHMETIC, BALLOT, SHUFFLE and QUAD bits set.</li> <li>• <code>supportedStages</code> include COMPUTE and FRAGMENT.</li> </ul> <p>Since 1.12</p>
VKD3D_SHADER_COMPILE_OPTION_FEATURE↔ _ZERO_INITIALIZE_WORKGROUP_MEMORY	<p>The SPIR-V target environment supports zero-initializing workgroup memory. This corresponds to the "shaderZeroInitializeWorkgroupMemory" Vulkan feature.</p> <p>Since 1.16</p>

## 4.3.4.5 vkd3d\_shader\_compile\_option\_formatting\_flags

```
enum vkd3d_shader_compile_option_formatting_flags
```

## Enumerator

VKD3D_SHADER_COMPILE_OPTION_↔ FORMATTING_IO_SIGNATURES	<p>Emit the signatures when disassembling a shader.</p> <p>Since 1.12</p>
---	---

## 4.3.4.6 vkd3d\_shader\_compile\_option\_fragment\_coordinate\_origin

```
enum vkd3d_shader_compile_option_fragment_coordinate_origin
```

Determines the origin of fragment coordinates.

## Since

1.10

## Enumerator

VKD3D_SHADER_COMPILE_OPTION_↔ FRAGMENT_COORDINATE_ORIGIN_UPPER_LEFT	Fragment coordinates originate from the upper-left. This is the default; it's also the only value supported by Vulkan environments.
VKD3D_SHADER_COMPILE_OPTION_↔ FRAGMENT_COORDINATE_ORIGIN_LOWER_↔ LEFT	Fragment coordinates originate from the lower-left. This matches the traditional behaviour of OpenGL environments.

## 4.3.4.7 vkd3d\_shader\_compile\_option\_name

```
enum vkd3d_shader_compile_option_name
```

## Enumerator

VKD3D_SHADER_COMPILE_OPTION_STRIP_↔ DEBUG	If <i>value</i> is nonzero, do not include debug information in the compiled shader. The default value is zero. This option is supported by <a href="#">vkd3d_shader_compile()</a> . However, not all compilers support generating debug information.
VKD3D_SHADER_COMPILE_OPTION_BUFFER_↔ UAV	<i>value</i> is a member of enum <a href="#">vkd3d_shader_compile_option_buffer_uav</a> .
VKD3D_SHADER_COMPILE_OPTION_↔ FORMATTING	<i>value</i> is a member of enum <a href="#">vkd3d_shader_compile_option_formatting_flags</a> .
VKD3D_SHADER_COMPILE_OPTION_API_↔ VERSION	<i>value</i> is a member of enum <a href="#">vkd3d_shader_api_version</a> .  Since  1.3
VKD3D_SHADER_COMPILE_OPTION_TYPED_↔ UAV	<i>value</i> is a member of enum <a href="#">vkd3d_shader_compile_option_typed_uav</a> .  Since  1.5
VKD3D_SHADER_COMPILE_OPTION_WRITE_↔ TESS_GEOM_POINT_SIZE	If <i>value</i> is nonzero, write the point size for Vulkan tessellation and geometry shaders. This option should be enabled if and only if the <code>shaderTessellationAndGeometryPointSize</code> feature is enabled. The default value is nonzero, i.e. write the point size. This option is supported by <a href="#">vkd3d_shader_compile()</a> for the SPIR-V target type and Vulkan targets; it should not be enabled otherwise.  Since  1.7

## Enumerator

VKD3D_SHADER_COMPILE_OPTION_PACK_↔ MATRIX_ORDER	<p>This option specifies default matrix packing order for HLSL sources. Explicit variable modifiers or pragmas will take precedence.</p> <p><i>value</i> is a member of enum <code>vkd3d_shader_compile_option_pack_matrix_order</code>.</p> <p>Since</p> <p>1.9</p>
VKD3D_SHADER_COMPILE_OPTION_↔ BACKWARD_COMPATIBILITY	<p>This option is used to enable various backward compatibility features. <i>value</i> is a mask of values from enum <code>vkd3d_shader_compile_option_backward_↔compatibility</code>.</p> <p>Since</p> <p>1.10</p>
VKD3D_SHADER_COMPILE_OPTION_↔ FRAGMENT_COORDINATE_ORIGIN	<p>This option specifies the origin of fragment coordinates for SPIR-V targets. <i>value</i> is a member of enum <code>vkd3d_shader_compile_option_fragment_↔coordinate_origin</code>.</p> <p>Since</p> <p>1.10</p>
VKD3D_SHADER_COMPILE_OPTION_FEATURE	<p>This option specifies the shader features available in the target environment. These are not extensions, i.e. they are always supported by the driver, but may not be supported by the available hardware.</p> <p><i>value</i> is a member of enum <code>vkd3d_shader_compile_option_feature_flags</code>.</p> <p>Since</p> <p>1.11</p>
VKD3D_SHADER_COMPILE_OPTION_CHILD_↔ EFFECT	<p>If <i>value</i> is non-zero compilation will produce a child effect using shared object descriptions, as instructed by the "shared" modifier. Child effects are supported with <code>fx_4_0</code>, and <code>fx_4_1</code> profiles. This option and "shared" modifiers are ignored for the <code>fx_5_0</code> profile and non-fx profiles. The <code>fx_2_0</code> profile does not have a separate concept of child effects, variables marked with "shared" modifier will be marked as such in a binary.</p> <p>Since</p> <p>1.12</p>

## Enumerator

VKD3D_SHADER_COMPILE_OPTION_WARN_IMPLICIT_TRUNCATION↔	<p>If <i>value</i> is nonzero, emit a compile warning warn when vectors or matrices are truncated in an implicit conversion. If warnings are disabled, this option has no effect. This option has no effects for targets other than HLSL.</p> <p>The default value is nonzero, i.e. enable implicit truncation warnings.</p> <p>Since</p> <p>1.12</p>
VKD3D_SHADER_COMPILE_OPTION_INCLUDE_EMPTY_BUFFERS_IN_EFFECTS↔	<p>If <i>value</i> is nonzero, empty constant buffers descriptions are written out in the output effect binary. This option applies only to fx_4_0 and fx_4_1 profiles and is otherwise ignored.</p> <p>Since</p> <p>1.12</p>

## 4.3.4.8 vkd3d\_shader\_compile\_option\_pack\_matrix\_order

enum vkd3d\_shader\_compile\_option\_pack\_matrix\_order

Determines how matrices are stored.

Since

1.9

## 4.3.4.9 vkd3d\_shader\_compile\_option\_typed\_uav

enum vkd3d\_shader\_compile\_option\_typed\_uav

Determines how typed UAVs are declared.

Since

1.5

## Enumerator

VKD3D_SHADER_COMPILE_OPTION_Typed_UAV_READ_FORMAT_R32↔	Use R32(u)/R32f format for UAVs which are read from. This is the default value.
VKD3D_SHADER_COMPILE_OPTION_Typed_UAV_READ_FORMAT_UNKNOWN↔	Use Unknown format for UAVs which are read from. This should only be set if shaderStorageImageReadWithoutFormat is enabled in the target environment.
Generated by Doxygen	

#### 4.3.4.10 vkd3d\_shader\_component\_type

enum [vkd3d\\_shader\\_component\\_type](#)

Data type of a shader varying, returned as part of struct [vkd3d\\_shader\\_signature\\_element](#).

##### Enumerator

VKD3D_SHADER_COMPONENT_VOID	The varying has no type.
VKD3D_SHADER_COMPONENT_UINT	32-bit unsigned integer.
VKD3D_SHADER_COMPONENT_INT	32-bit signed integer.
VKD3D_SHADER_COMPONENT_FLOAT	32-bit IEEE floating-point.
VKD3D_SHADER_COMPONENT_BOOL	Boolean.
VKD3D_SHADER_COMPONENT_DOUBLE	64-bit IEEE floating-point.
VKD3D_SHADER_COMPONENT_UINT64	64-bit unsigned integer.  Since 1.11
VKD3D_SHADER_COMPONENT_INT64	64-bit signed integer.  Since 1.16
VKD3D_SHADER_COMPONENT_FLOAT16	16-bit IEEE floating-point.  Since 1.16
VKD3D_SHADER_COMPONENT_UINT16	16-bit unsigned integer.  Since 1.16
VKD3D_SHADER_COMPONENT_INT16	16-bit signed integer.  Since 1.16

#### 4.3.4.11 vkd3d\_shader\_d3dbc\_constant\_register

enum [vkd3d\\_shader\\_d3dbc\\_constant\\_register](#)

Symbolic register indices for mapping uniform constant register sets in legacy Direct3D bytecode to constant buffer views in the target environment.

Members of this enumeration are used in [vkd3d\\_shader\\_resource\\_binding::register\\_index](#).

Since

1.9

Enumerator

VKD3D_SHADER_D3DBC_FLOAT_CONSTANT_↔ REGISTER	The float constant register set, c# in Direct3D assembly.
VKD3D_SHADER_D3DBC_INT_CONSTANT_↔ REGISTER	The integer constant register set, i# in Direct3D assembly.
VKD3D_SHADER_D3DBC_BOOL_CONSTANT_↔ REGISTER	The boolean constant register set, b# in Direct3D assembly.

#### 4.3.4.12 vkd3d\_shader\_descriptor\_info\_flag

```
enum vkd3d_shader_descriptor_info_flag
```

Additional flags describing a shader descriptor, returned as part of struct [vkd3d\\_shader\\_descriptor\\_info](#).

Enumerator

VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_↔ UAV_COUNTER	The descriptor is a UAV resource, whose counter is read from or written to by the shader.
VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_↔ UAV_READ	The descriptor is a UAV resource, which is read from by the shader.
VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_↔ SAMPLER_COMPARISON_MODE	The descriptor is a comparison sampler.
VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_↔ UAV_ATOMICS	The descriptor is a UAV resource, on which the shader performs atomic ops.  Since 1.6
VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_↔ RAW_BUFFER	The descriptor is a raw (byte-addressed) buffer.  Since 1.9

#### 4.3.4.13 vkd3d\_shader\_descriptor\_range\_flags

```
enum vkd3d_shader_descriptor_range_flags
```

## Enumerator

VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_DESCRIPTOR_STATIC_KEEPING_↔ BUFFER_BOUNDS_CHECKS	Since 1.11
--	---------------

## 4.3.4.14 vkd3d\_shader\_descriptor\_type

enum [vkd3d\\_shader\\_descriptor\\_type](#)

The type of a shader resource descriptor.

## Enumerator

VKD3D_SHADER_DESCRIPTOR_TYPE_SRV	The descriptor is a shader resource view. In Direct3D assembly, this is bound to a t# register.
VKD3D_SHADER_DESCRIPTOR_TYPE_UAV	The descriptor is an unordered access view. In Direct3D assembly, this is bound to a u# register.
VKD3D_SHADER_DESCRIPTOR_TYPE_CBV	The descriptor is a constant buffer view. In Direct3D assembly, this is bound to a cb# register.
VKD3D_SHADER_DESCRIPTOR_TYPE_SAMPLER	The descriptor is a sampler. In Direct3D assembly, this is bound to an s# register.

## 4.3.4.15 vkd3d\_shader\_fog\_fragment\_mode

enum [vkd3d\\_shader\\_fog\\_fragment\\_mode](#)

The factor used to interpolate the fragment output colour with fog.

See VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_FRAGMENT\_MODE for specification of the interpolation factor as defined here.

The following variables may be used to determine the interpolation factor:

c = The fog coordinate value output from the vertex shader. This is an inter-stage varying with the semantic name "FOG" and semantic index 0. It may be modified by VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_SOURCE.  
 E = The value of VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_END. k = The value of VKD3D\_SHADER\_↔  
 PARAMETER\_NAME\_FOG\_SCALE.

## Since

1.15

## Enumerator

VKD3D_SHADER_FOG_FRAGMENT_NONE	No fog interpolation is applied; the output colour is passed through unmodified. Equivalently, the fog interpolation factor is 1.
VKD3D_SHADER_FOG_FRAGMENT_EXP	<p>The fog interpolation factor is <math>2^{-(k * c)}</math>. In order to implement traditional exponential fog, as present in Direct3D and OpenGL, i.e.</p> $e^{-(\text{density} * c)}$ <p>set</p> $k = \text{density} * \log(e)$
VKD3D_SHADER_FOG_FRAGMENT_EXP2	<p>The fog interpolation factor is <math>2^{-((k * c)^2)}</math>. In order to implement traditional square-exponential fog, as present in Direct3D and OpenGL, i.e.</p> $e^{-((\text{density} * c)^2)}$ <p>set</p> $k = \text{density} * \log(e)$
VKD3D_SHADER_FOG_FRAGMENT_LINEAR	<p>The fog interpolation factor is <math>(E - c) * k</math>. In order to implement traditional linear fog, as present in Direct3D and OpenGL, i.e.</p> $(\text{end} - c) / (\text{end} - \text{start})$ <p>set</p> $E = \text{end}$ $k = 1 / (\text{end} - \text{start})$

## 4.3.4.16 vkd3d\_shader\_fog\_source

```
enum vkd3d_shader_fog_source
```

The source of the fog varying output by a pre-rasterization shader.

The fog varying is defined as the output varying with the semantic name "FOG" and semantic index 0.

See VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_SOURCE for further documentation of this parameter.

Since

1.15

## Enumerator

VKD3D_SHADER_FOG_SOURCE_FOG	The source shader is not modified. That is, the fog varying in the target shader is the original fog varying if and only if present.
-----------------------------	--

## Enumerator

VKD3D_SHADER_FOG_SOURCE_FOG_OR_SPECULAR_W	If the source shader has a fog varying, it is not modified. Otherwise, if the source shader outputs a varying with semantic name "COLOR" and semantic index 1 whose index includes a W component, said W component is output as fog varying. Otherwise, no fog varying is output.
VKD3D_SHADER_FOG_SOURCE_Z	The fog source is the Z component of the position output by the vertex shader.
VKD3D_SHADER_FOG_SOURCE_W	The fog source is the W component of the position output by the vertex shader.

## 4.3.4.17 vkd3d\_shader\_log\_level

enum [vkd3d\\_shader\\_log\\_level](#)

Describes the minimum severity of compilation messages returned by [vkd3d\\_shader\\_compile\(\)](#) and similar functions.

## Enumerator

VKD3D_SHADER_LOG_NONE	No messages will be returned.
VKD3D_SHADER_LOG_ERROR	Only fatal errors which prevent successful compilation will be returned.
VKD3D_SHADER_LOG_WARNING	Non-fatal warnings and fatal errors will be returned.
VKD3D_SHADER_LOG_INFO	All messages, including general informational messages, will be returned.

## 4.3.4.18 vkd3d\_shader\_minimum\_precision

enum [vkd3d\\_shader\\_minimum\\_precision](#)

Minimum interpolation precision of a shader varying, returned as part of struct [vkd3d\\_shader\\_signature\\_element](#).

## Enumerator

VKD3D_SHADER_MINIMUM_PRECISION_FLOAT_16	16-bit floating-point.
VKD3D_SHADER_MINIMUM_PRECISION_FIXED_8_2	10-bit fixed point (2 integer and 8 fractional bits).
VKD3D_SHADER_MINIMUM_PRECISION_INT_16	16-bit signed integer.
VKD3D_SHADER_MINIMUM_PRECISION_UINT_16	16-bit unsigned integer.

#### 4.3.4.19 vkd3d\_shader\_parameter\_data\_type

enum [vkd3d\\_shader\\_parameter\\_data\\_type](#)

The format of data provided to the shader, used in struct [vkd3d\\_shader\\_parameter](#) and struct [vkd3d\\_shader\\_parameter1](#).

##### Enumerator

VKD3D_SHADER_PARAMETER_DATA_TYPE_↔ UINT32	The parameter is provided as a 32-bit unsigned integer.
VKD3D_SHADER_PARAMETER_DATA_TYPE_↔ FLOAT32	The parameter is provided as a 32-bit float.  Since  1.13
VKD3D_SHADER_PARAMETER_DATA_TYPE_↔ FLOAT32_VEC4	The parameter is provided as a 4-dimensional vector of 32-bit floats. This parameter must be used with struct <a href="#">vkd3d_shader_parameter1</a> ; it cannot be used with struct <a href="#">vkd3d_shader_parameter</a> .  Since  1.14

#### 4.3.4.20 vkd3d\_shader\_parameter\_name

enum [vkd3d\\_shader\\_parameter\\_name](#)

Names a specific shader parameter, used in struct [vkd3d\\_shader\\_parameter](#) and struct [vkd3d\\_shader\\_parameter1](#).

##### Enumerator

VKD3D_SHADER_PARAMETER_NAME_↔ RASTERIZER_SAMPLE_COUNT	The sample count of the framebuffer, as returned by the HLSL function <code>GetRenderTargetSampleCount()</code> or the GLSL builtin <code>gl_NumSamples</code> . This parameter should be specified when compiling to SPIR-V, which provides no builtin ability to query this information from the shader.  The default value is 1.  The data type for this parameter must be VKD3D_↔ SHADER_PARAMETER_DATA_TYPE_UINT32.
--	---

## Enumerator

VKD3D_SHADER_PARAMETER_NAME_ALPHA_↔ TEST_FUNC	<p>Alpha test comparison function. When this parameter is provided, if the alpha component of the pixel shader colour output at location 0 fails the test, as defined by this function and the reference value provided by VKD3D_SHADER_PARAMETER_NAME_ALPHA_↔ TEST_REF, the fragment will be discarded.</p> <p>This parameter, along with VKD3D_SHADER_↔ PARAMETER_NAME_ALPHA_TEST_REF, can be used to implement fixed function alpha test, as present in Direct3D versions up to 9, if the target environment does not support alpha test as part of its own fixed-function API (as Vulkan and core OpenGL). The default value is VKD3D_SHADER_COMPARISON_FUNC_ALWAYS.</p> <p>The data type for this parameter must be VKD3D_↔ SHADER_PARAMETER_DATA_TYPE_UINT32. The value specified must be a member of enum vkd3d_shader_comparison_func.</p> <p>Only VKD3D_SHADER_PARAMETER_TYPE_↔ IMMEDIATE_CONSTANT is supported in this version of vkd3d-shader.</p> <p>Since</p> <p>1.13</p>
VKD3D_SHADER_PARAMETER_NAME_ALPHA_↔ TEST_REF	<p>Alpha test reference value. See VKD3D_SHADER_↔ PARAMETER_NAME_ALPHA_TEST_FUNC for documentation of alpha test.</p> <p>The default value is zero.</p> <p>Since</p> <p>1.13</p>
VKD3D_SHADER_PARAMETER_NAME_FLAT_↔ INTERPOLATION	<p>Whether to use flat interpolation for fragment shader colour inputs. If the value is nonzero, inputs whose semantic usage is COLOR will use flat interpolation instead of linear. This parameter is ignored if the shader model is 4 or greater, since only shader model 3 and below do not specify the interpolation mode in the shader bytecode.</p> <p>This parameter can be used to implement fixed function shade mode, as present in Direct3D versions up to 9, if the target environment does not support shade mode as part of its own fixed-function API (as Vulkan and core OpenGL).</p> <p>The data type for this parameter must be VKD3D_↔ SHADER_PARAMETER_DATA_TYPE_UINT32. The default value is zero, i.e. use linear interpolation.</p> <p>Only VKD3D_SHADER_PARAMETER_TYPE_↔ IMMEDIATE_CONSTANT is supported in this version of vkd3d-shader.</p> <p>Since</p> <p>1.13</p>

## Enumerator

VKD3D_SHADER_PARAMETER_NAME_CLIP_↔ PLANE_MASK	<p>A mask of enabled clip planes. When this parameter is provided to a vertex shader, for each nonzero bit of this mask, a user clip distance will be generated from vertex position in clip space, and the clip plane defined by the indexed vector, taken from the VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_# parameter.</p> <p>Regardless of the specific clip planes which are enabled, the clip distances which are output are a contiguous array starting from clip distance 0. This affects the interface of OpenGL. For example, if only clip planes 1 and 3 are enabled (and so the value of the mask is 0xa), the user should enable only GL_CLIP_DISTANCE0 and GL_CLIP_DISTANCE1. The default value is zero, i.e. do not enable any clip planes.</p> <p>The data type for this parameter must be VKD3D_SHADER_PARAMETER_DATA_TYPE_UINT32. Only VKD3D_SHADER_PARAMETER_TYPE_IMMEDIATE_CONSTANT is supported in this version of vkd3d-shader.</p> <p>If the source shader writes clip distances and this parameter is nonzero, compilation fails.</p> <p>Since</p> <p>1.14</p>
VKD3D_SHADER_PARAMETER_NAME_CLIP_↔ PLANE_0	<p>Clip plane values. See VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_MASK for documentation of clip planes.</p> <p>These enum values are contiguous and arithmetic may safely be performed on them. That is, VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_[n] is VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_0 plus n.</p> <p>The data type for each parameter must be VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32_VEC4.</p> <p>The default value for each plane is a (0, 0, 0, 0) vector.</p> <p>Since</p> <p>1.14</p>

## Enumerator

VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE↔	<p>Point size. When this parameter is provided to a vertex, tessellation, or geometry shader, and the source shader does not write point size, it specifies a uniform value which will be written to point size. If the source shader writes point size, this parameter is ignored.</p> <p>This parameter can be used to implement fixed function point size, as present in Direct3D versions 8 and 9, if the target environment does not support point size as part of its own fixed-function API (as Vulkan and core OpenGL).</p> <p>The data type for this parameter must be VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32.</p> <p>Since 1.14</p>
VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE_MIN↔	<p>Minimum point size. When this parameter is provided to a vertex, tessellation, or geometry shader, and the source shader writes point size or uses the VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE↔ parameter, the point size will be clamped to the provided minimum value. If point size is not written in one of these ways, this parameter is ignored. If this parameter is not provided, the point size will not be clamped to a minimum size by vkd3d-shader.</p> <p>This parameter can be used to implement fixed function point size, as present in Direct3D versions 8 and 9, if the target environment does not support point size as part of its own fixed-function API (as Vulkan and core OpenGL).</p> <p>The data type for this parameter must be VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32.</p> <p>Since 1.14</p>
VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE_MAX↔	<p>Maximum point size. This parameter has identical behaviour to VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE_MIN↔, except that it provides the maximum size rather than the minimum.</p> <p>Since 1.14</p>

## Enumerator

VKD3D_SHADER_PARAMETER_NAME_POINT_↔ SPRITE	<p>Whether texture coordinate inputs should take their values from the point coordinate. When this parameter is provided to a pixel shader, and the value is nonzero, any fragment shader input with the semantic name "TEXCOORD" takes its value from the point coordinates instead of from the previous shader. The point coordinates here are defined as a four-component vector whose X and Y components are the X and Y coordinates of the fragment within a point being rasterized, and whose Z and W components are zero.</p> <p>In GLSL, the X and Y components are drawn from <code>gl_PointCoord</code>; in SPIR-V, they are drawn from a variable with the <code>BuiltinPointCoord</code> decoration. This includes <code>t#</code> fragment shader inputs in shader model 2 shaders, as well as texture sampling in shader model 1 shaders.</p> <p>This parameter can be used to implement fixed function point sprite, as present in Direct3D versions 8 and 9, if the target environment does not support point sprite as part of its own fixed-function API (as Vulkan and core OpenGL).</p> <p>The data type for this parameter must be VKD3D_↔ SHADER_PARAMETER_DATA_TYPE_UINT32.</p> <p>The default value is zero, i.e. use the original varyings. Only VKD3D_SHADER_PARAMETER_TYPE_↔ IMMEDIATE_CONSTANT is supported in this version of vkd3d-shader.</p> <p>Since</p> <p>1.14</p>
---	---

## Enumerator

<p>VKD3D_SHADER_PARAMETER_NAME_FOG_↔ FRAGMENT_MODE</p>	<p>Fog mode used in fragment shaders. The value specified by this parameter must be a member of enum vkd3d_shader_fog_fragment_mode. If not VKD3D_SHADER_FOG_FRAGMENT_NONE, the pixel shader colour output at location 0 is linearly interpolated with the fog colour defined by VKD3D_↔ SHADER_PARAMETER_NAME_FOG_COLOUR. The interpolation factor is defined according to the enumerant selected by this parameter. The interpolated value is then outputted instead of the original value at location 0.</p> <p>An interpolation factor of 0 specifies to use the fog colour; a factor of 1 specifies to use the original colour output. The interpolation factor is clamped to the [0, 1] range before interpolating.</p> <p>The default value is VKD3D_SHADER_FOG_FRAGMENT_NONE.</p> <p>The data type for this parameter must be VKD3D_↔ SHADER_PARAMETER_DATA_TYPE_UINT32.</p> <p>Only VKD3D_SHADER_PARAMETER_TYPE_↔ IMMEDIATE_CONSTANT is supported in this version of vkd3d-shader.</p> <p>Since 1.15</p>
<p>VKD3D_SHADER_PARAMETER_NAME_FOG_↔ COLOUR</p>	<p>Fog colour. See VKD3D_SHADER_PARAMETER_↔ NAME_FOG_FRAGMENT_MODE for documentation of fog.</p> <p>The data type for this parameter must be VKD3D_SHADER_PARAMETER_DATA_TYPE_↔ FLOAT32_VEC4.</p> <p>The default value is transparent black, i.e. the vector {0, 0, 0, 0}.</p> <p>Since 1.15</p>
<p>VKD3D_SHADER_PARAMETER_NAME_FOG_END</p>	<p>End coordinate for linear fog. See VKD3D_SHADER_PARAMETER_NAME_FOG_↔ FRAGMENT_MODE for documentation of fog.</p> <p>The data type for this parameter must be VKD3D_↔ SHADER_PARAMETER_DATA_TYPE_FLOAT32.</p> <p>The default value is 1.0.</p> <p>Since 1.15</p>

## Enumerator

VKD3D_SHADER_PARAMETER_NAME_FOG_↔ SCALE	<p>Scale value for fog. See VKD3D_SHADER_↔ PARAMETER_NAME_FOG_FRAGMENT_MODE for documentation of fog.</p> <p>The data type for this parameter must be VKD3D_↔ SHADER_PARAMETER_DATA_TYPE_FLOAT32.</p> <p>The default value is 1.0.</p> <p>Since</p> <p>1.15</p>
VKD3D_SHADER_PARAMETER_NAME_FOG_↔ SOURCE	<p>Fog source. The value specified by this parameter must be a member of enum <code>vkd3d_shader_fog_source</code>.</p> <p>This parameter replaces or suppletes the fog varying output by a pre-rasterization shader. The fog varying is defined as the output varying with the semantic name "FOG" and semantic index 0.</p> <p>Together with other fog parameters, this parameter can be used to implement fixed function fog, as present in Direct3D versions up to 9, if the target environment does not support fog as part of its own fixed-function API (as Vulkan and core OpenGL).</p> <p>The default value is <code>VKD3D_SHADER_FOG_SOURCE_FOG</code>.</p> <p>The data type for this parameter must be VKD3D_↔ SHADER_PARAMETER_DATA_TYPE_UINT32.</p> <p>Only VKD3D_SHADER_PARAMETER_TYPE_↔ IMMEDIATE_CONSTANT is supported in this version of vkd3d-shader.</p> <p>Since</p> <p>1.15</p>

## 4.3.4.21 vkd3d\_shader\_parameter\_type

```
enum vkd3d_shader_parameter_type
```

The manner in which a parameter value is provided to the shader, used in struct `vkd3d_shader_parameter` and struct `vkd3d_shader_parameter1`.

## Enumerator

VKD3D_SHADER_PARAMETER_TYPE_↔ IMMEDIATE_CONSTANT	The parameter value is embedded directly in the shader.
VKD3D_SHADER_PARAMETER_TYPE_↔ SPECIALIZATION_CONSTANT	The parameter value is provided to the shader via specialization constants. This value is only supported for the SPIR-V target type.
VKD3D_SHADER_PARAMETER_TYPE_BUFFER	<p>The parameter value is provided to the shader as part of a uniform buffer.</p> <p>Since</p> <p>1.13</p>

#### 4.3.4.22 vkd3d\_shader\_parse\_dxbc\_flags

enum [vkd3d\\_shader\\_parse\\_dxbc\\_flags](#)

Flags for [vkd3d\\_shader\\_parse\\_dxbc\(\)](#).

Since

1.12

Enumerator

VKD3D_SHADER_PARSE_DXBC_IGNORE_↔ CHECKSUM	Ignore the checksum and continue parsing even if it is incorrect.
--	---

#### 4.3.4.23 vkd3d\_shader\_resource\_data\_type

enum [vkd3d\\_shader\\_resource\\_data\\_type](#)

The type of the data contained in a shader resource, returned as part of struct [vkd3d\\_shader\\_descriptor\\_info](#).

All formats are 32-bit.

Enumerator

VKD3D_SHADER_RESOURCE_DATA_UNORM	Unsigned normalized integer.
VKD3D_SHADER_RESOURCE_DATA_SNORM	Signed normalized integer.
VKD3D_SHADER_RESOURCE_DATA_INT	Signed integer.
VKD3D_SHADER_RESOURCE_DATA_UINT	Unsigned integer.
VKD3D_SHADER_RESOURCE_DATA_FLOAT	IEEE single-precision floating-point.
VKD3D_SHADER_RESOURCE_DATA_MIXED	Undefined/type-less.  Since 1.3
VKD3D_SHADER_RESOURCE_DATA_DOUBLE	IEEE double-precision floating-point.  Since 1.3
VKD3D_SHADER_RESOURCE_DATA_CONTINUED	Continuation of the previous component. For example, 64-bit double-precision floating-point data may be returned as two 32-bit components, with the first component (containing the LSB) specified as VKD3D_SHADER_RESOURCE_DATA_DOUBLE, and the second component specified as VKD3D_↔ SHADER_RESOURCE_DATA_CONTINUED.  Since 1.3
	Generated by Doxygen

## 4.3.4.24 vkd3d\_shader\_resource\_type

enum [vkd3d\\_shader\\_resource\\_type](#)

The type of a shader resource, returned as part of struct [vkd3d\\_shader\\_descriptor\\_info](#).

## Enumerator

VKD3D_SHADER_RESOURCE_NONE	The type is invalid or not applicable for this descriptor. This value is returned for samplers.
VKD3D_SHADER_RESOURCE_BUFFER	Dimensionless buffer.
VKD3D_SHADER_RESOURCE_TEXTURE_1D	1-dimensional texture.
VKD3D_SHADER_RESOURCE_TEXTURE_2D	2-dimensional texture.
VKD3D_SHADER_RESOURCE_TEXTURE_2DMS	Multisampled 2-dimensional texture.
VKD3D_SHADER_RESOURCE_TEXTURE_3D	3-dimensional texture.
VKD3D_SHADER_RESOURCE_TEXTURE_CUBE	Cubemap texture.
VKD3D_SHADER_RESOURCE_TEXTURE_1↔ DARRAY	1-dimensional array texture.
VKD3D_SHADER_RESOURCE_TEXTURE_2↔ DARRAY	2-dimensional array texture.
VKD3D_SHADER_RESOURCE_TEXTURE_2↔ DMSARRAY	Multisampled 2-dimensional array texture.
VKD3D_SHADER_RESOURCE_TEXTURE_↔ CUBEARRAY	Cubemap array texture.

## 4.3.4.25 vkd3d\_shader\_source\_type

enum [vkd3d\\_shader\\_source\\_type](#)

The format of a shader to be compiled or scanned.

## Enumerator

VKD3D_SHADER_SOURCE_NONE	The shader has no type or is to be ignored. This is not a valid value for <a href="#">vkd3d_shader_compile()</a> or <a href="#">vkd3d_shader_scan()</a> .
VKD3D_SHADER_SOURCE_DXBC_TPF	A 'Tokenized Program Format' shader embedded in a DXBC container. This is the format used for Direct3D shader model 4 and 5 shaders.
VKD3D_SHADER_SOURCE_HLSL	High-Level Shader Language source code.  Since 1.3
VKD3D_SHADER_SOURCE_D3D_BYTECODE	Legacy Direct3D byte-code. This is the format used for Direct3D shader model 1, 2, and 3 shaders.  Since 1.3

## Enumerator

VKD3D_SHADER_SOURCE_DXBC_DXIL	<p>A 'DirectX Intermediate Language' shader embedded in a DXBC container. This is the format used for Direct3D shader model 6 shaders.</p> <p>Since 1.9</p>
VKD3D_SHADER_SOURCE_FX	<p>Binary format used by Direct3D 9/10.x/11 effects. Input is a raw FX section without container.</p> <p>Since 1.14</p>

## 4.3.4.26 vkd3d\_shader\_spirv\_environment

```
enum vkd3d_shader_spirv_environment
```

## Enumerator

VKD3D_SHADER_SPIRV_ENVIRONMENT_VULKAN_1↔ _1	<p>Since 1.12</p>
--	-----------------------

## 4.3.4.27 vkd3d\_shader\_spirv\_extension

```
enum vkd3d_shader_spirv_extension
```

## Enumerator

VKD3D_SHADER_SPIRV_EXTENSION_EXT_DESCRIPTOR_INDEXING	<p>Since 1.3</p>
VKD3D_SHADER_SPIRV_EXTENSION_EXT_STENCIL_EXPORT	<p>Since 1.3</p>
VKD3D_SHADER_SPIRV_EXTENSION_EXT_VIEWPORT_INDEX_LAYER	<p>Since 1.11</p>

## Enumerator

VKD3D_SHADER_SPIRV_EXTENSION_EXT_FRAGMENT_SHADER_INTERLOCK	Since 1.12
--	---------------

## 4.3.4.28 vkd3d\_shader\_structure\_type

enum [vkd3d\\_shader\\_structure\\_type](#)

The type of a chained structure.

## Enumerator

VKD3D_SHADER_STRUCTURE_TYPE_↔ COMPILE_INFO	The structure is a <a href="#">vkd3d_shader_compile_info</a> structure.
VKD3D_SHADER_STRUCTURE_TYPE_↔ INTERFACE_INFO	The structure is a <a href="#">vkd3d_shader_interface_info</a> structure.
VKD3D_SHADER_STRUCTURE_TYPE_↔ DESCRIPTOR_INFO	The structure is a <a href="#">vkd3d_shader_scan_descriptor_info</a> structure.
VKD3D_SHADER_STRUCTURE_TYPE_↔ DOMAIN_SHADER_TARGET_INFO	The structure is a <a href="#">vkd3d_shader_spirv_domain_shader_target_info</a> structure.
VKD3D_SHADER_STRUCTURE_TYPE_↔ TARGET_INFO	The structure is a <a href="#">vkd3d_shader_spirv_target_info</a> structure.
VKD3D_SHADER_STRUCTURE_TYPE_↔ TRANSFORM_FEEDBACK_INFO	The structure is a <a href="#">vkd3d_shader_transform_feedback_info</a> structure.
VKD3D_SHADER_STRUCTURE_TYPE_↔ HLSL_SOURCE_INFO	The structure is a <a href="#">vkd3d_shader_hlsl_source_info</a> structure.  Since 1.3
VKD3D_SHADER_STRUCTURE_TYPE_↔ PREPROCESS_INFO	The structure is a <a href="#">vkd3d_shader_preprocess_info</a> structure.  Since 1.3
VKD3D_SHADER_STRUCTURE_TYPE_↔ DESCRIPTOR_OFFSET_INFO	The structure is a <a href="#">vkd3d_shader_descriptor_offset_info</a> structure.  Since 1.3
VKD3D_SHADER_STRUCTURE_TYPE_↔ SCAN_SIGNATURE_INFO	The structure is a <a href="#">vkd3d_shader_scan_signature_info</a> structure.  Since 1.9

## Enumerator

VKD3D_SHADER_STRUCTURE_TYPE_↔ VARYING_MAP_INFO	The structure is a <a href="#">vkd3d_shader_varying_map_info</a> structure.  Since 1.9
VKD3D_SHADER_STRUCTURE_TYPE_SCAN_↔ COMBINED_RESOURCE_SAMPLER_INFO	The structure is a <a href="#">vkd3d_shader_scan_combined_resource_sampler_info</a> structure.  Since 1.10
VKD3D_SHADER_STRUCTURE_TYPE_↔ PARAMETER_INFO	The structure is a <a href="#">vkd3d_shader_parameter_info</a> structure.  Since 1.13
VKD3D_SHADER_STRUCTURE_TYPE_SCAN_↔ HULL_SHADER_TESSELLATION_INFO	The structure is a <a href="#">vkd3d_shader_scan_hull_shader_tessellation_info</a> structure.  Since 1.15

## 4.3.4.29 vkd3d\_shader\_sysval\_semantic

```
enum vkd3d_shader_sysval_semantic
```

System value semantic, returned as part of struct [vkd3d\\_shader\\_signature](#).

## Enumerator

VKD3D_SHADER_SV_NONE	No system value.
VKD3D_SHADER_SV_POSITION	Vertex position; SV_Position in Direct3D.
VKD3D_SHADER_SV_CLIP_DISTANCE	Clip distance; SV_ClipDistance in Direct3D.
VKD3D_SHADER_SV_CULL_DISTANCE	Cull distance; SV_CullDistance in Direct3D.
VKD3D_SHADER_SV_RENDER_TARGET_↔ ARRAY_INDEX	Render target layer; SV_RenderTargetArrayIndex in Direct3D.
VKD3D_SHADER_SV_VIEWPORT_ARRAY_INDEX	Viewport index; SV_ViewportArrayIndex in Direct3D.
VKD3D_SHADER_SV_VERTEX_ID	Vertex ID; SV_VertexID in Direct3D.
VKD3D_SHADER_SV_PRIMITIVE_ID	Primitive ID; SV_PrimitiveID in Direct3D.
VKD3D_SHADER_SV_INSTANCE_ID	Instance ID; SV_InstanceID in Direct3D.
VKD3D_SHADER_SV_IS_FRONT_FACE	Whether the triangle is front-facing; SV_IsFrontFace in Direct3D.
VKD3D_SHADER_SV_SAMPLE_INDEX	Sample index; SV_SampleIndex in Direct3D.

## Enumerator

VKD3D_SHADER_SV_TARGET	Render target; SV_Target in Direct3D.  Since 1.9
VKD3D_SHADER_SV_DEPTH	Depth; SV_Depth in Direct3D.  Since 1.9
VKD3D_SHADER_SV_COVERAGE	Sample mask; SV_Coverage in Direct3D.  Since 1.9
VKD3D_SHADER_SV_DEPTH_GREATER_EQUAL	Depth, which is guaranteed to be greater than or equal to the current depth; SV_DepthGreaterEqual in Direct3D.  Since 1.9
VKD3D_SHADER_SV_DEPTH_LESS_EQUAL	Depth, which is guaranteed to be less than or equal to the current depth; SV_DepthLessEqual in Direct3D.  Since 1.9
VKD3D_SHADER_SV_STENCIL_REF	Stencil reference; SV_StencilRef in Direct3D.  Since 1.9

## 4.3.4.30 vkd3d\_shader\_target\_type

```
enum vkd3d_shader_target_type
```

The output format of a compiled shader.

## Enumerator

VKD3D_SHADER_TARGET_NONE	The shader has no type or is to be ignored. This is not a valid value for <a href="#">vkd3d_shader_compile()</a> .
VKD3D_SHADER_TARGET_SPIRV_BINARY	A SPIR-V shader in binary form. This is the format used for Vulkan shaders.
VKD3D_SHADER_TARGET_D3D_ASM	Direct3D shader assembly.  Since 1.3

## Enumerator

VKD3D_SHADER_TARGET_D3D_BYTECODE	Legacy Direct3D byte-code. This is the format used for Direct3D shader model 1, 2, and 3 shaders.  Since 1.3
VKD3D_SHADER_TARGET_DXBC_TPF	A 'Tokenized Program Format' shader embedded in a DXBC container. This is the format used for Direct3D shader model 4 and 5 shaders.  Since 1.3
VKD3D_SHADER_TARGET_GLSL	An 'OpenGL Shading Language' shader.  Since 1.3
VKD3D_SHADER_TARGET_FX	Binary format used by Direct3D 9/10.x/11 effects profiles. Output is a raw FX section without container.  Since 1.11
VKD3D_SHADER_TARGET_MSL	A 'Metal Shading Language' shader.  Since 1.14

## 4.3.4.31 vkd3d\_shader\_visibility

```
enum vkd3d_shader_visibility
```

Describes which shader stages a resource is visible to.

## Enumerator

VKD3D_SHADER_VISIBILITY_ALL	The resource is visible to all shader stages.
VKD3D_SHADER_VISIBILITY_VERTEX	The resource is visible only to the vertex shader.
VKD3D_SHADER_VISIBILITY_HULL	The resource is visible only to the hull shader.
VKD3D_SHADER_VISIBILITY_DOMAIN	The resource is visible only to the domain shader.
VKD3D_SHADER_VISIBILITY_GEOMETRY	The resource is visible only to the geometry shader.
VKD3D_SHADER_VISIBILITY_PIXEL	The resource is visible only to the pixel shader.
VKD3D_SHADER_VISIBILITY_COMPUTE	The resource is visible only to the compute shader.

## 4.3.5 Function Documentation

### 4.3.5.1 vkd3d\_shader\_build\_varying\_map()

```
VKD3D_SHADER_API void vkd3d_shader_build_varying_map (
    const struct vkd3d_shader_signature * output_signature,
    const struct vkd3d_shader_signature * input_signature,
    unsigned int * count,
    struct vkd3d_shader_varying_map * varyings )
```

Build a mapping of output varyings in a shader stage to input varyings in the following shader stage.

This mapping should be used in struct [vkd3d\\_shader\\_varying\\_map\\_info](#) to compile the first shader.

#### Parameters

<i>output_signature</i>	The output signature of the first shader.
<i>input_signature</i>	The input signature of the second shader.
<i>count</i>	On output, contains the number of entries written into "varyings".
<i>varyings</i>	Pointer to an output array of varyings. This must point to space for N varyings, where N is the number of elements in the input signature.

#### Remarks

Valid legacy Direct3D pixel shaders have at most 12 varying inputs: 10 inter-stage varyings, face, and position. Therefore, in practice, it is safe to call this function with a pre-allocated array with a fixed size of 12.

#### Since

1.9

### 4.3.5.2 vkd3d\_shader\_compile()

```
VKD3D_SHADER_API int vkd3d_shader_compile (
    const struct vkd3d_shader_compile_info * compile_info,
    struct vkd3d_shader_code * out,
    char ** messages )
```

Transform a form of GPU shader source code or byte code into another form of source code or byte code.

This version of vkd3d-shader supports the following transformations:

- VKD3D\_SHADER\_SOURCE\_DXBC\_TPF to VKD3D\_SHADER\_TARGET\_SPIRV\_BINARY
- VKD3D\_SHADER\_SOURCE\_DXBC\_TPF to VKD3D\_SHADER\_TARGET\_SPIRV\_TEXT (if vkd3d was compiled with SPIRV-Tools)

- VKD3D\_SHADER\_SOURCE\_DXBC\_TPF to VKD3D\_SHADER\_TARGET\_D3D\_ASM
- VKD3D\_SHADER\_SOURCE\_D3D\_BYTECODE to VKD3D\_SHADER\_TARGET\_SPIRV\_BINARY
- VKD3D\_SHADER\_SOURCE\_D3D\_BYTECODE to VKD3D\_SHADER\_TARGET\_SPIRV\_TEXT (if vkd3d was compiled with SPIRV-Tools)
- VKD3D\_SHADER\_SOURCE\_D3D\_BYTECODE to VKD3D\_SHADER\_TARGET\_D3D\_ASM
- VKD3D\_SHADER\_SOURCE\_HLSL to VKD3D\_SHADER\_TARGET\_SPIRV\_BINARY
- VKD3D\_SHADER\_SOURCE\_HLSL to VKD3D\_SHADER\_TARGET\_SPIRV\_TEXT (if vkd3d was compiled with SPIRV-Tools)
- VKD3D\_SHADER\_SOURCE\_HLSL to VKD3D\_SHADER\_TARGET\_D3D\_ASM
- VKD3D\_SHADER\_SOURCE\_HLSL to VKD3D\_SHADER\_TARGET\_D3D\_BYTECODE
- VKD3D\_SHADER\_SOURCE\_HLSL to VKD3D\_SHADER\_TARGET\_DXBC\_TPF
- VKD3D\_SHADER\_SOURCE\_HLSL to VKD3D\_SHADER\_TARGET\_FX
- VKD3D\_SHADER\_SOURCE\_FX to VKD3D\_SHADER\_TARGET\_D3D\_ASM

Supported transformations can also be detected at runtime with the functions [vkd3d\\_shader\\_get\\_supported\\_source\\_types\(\)](#) and [vkd3d\\_shader\\_get\\_supported\\_target\\_types\(\)](#).

Depending on the source and target types, this function may support the following chained structures:

- [vkd3d\\_shader\\_descriptor\\_offset\\_info](#)
- [vkd3d\\_shader\\_hlsl\\_source\\_info](#)
- [vkd3d\\_shader\\_interface\\_info](#)
- [vkd3d\\_shader\\_parameter\\_info](#)
- [vkd3d\\_shader\\_preprocess\\_info](#)
- [vkd3d\\_shader\\_scan\\_combined\\_resource\\_sampler\\_info](#)
- [vkd3d\\_shader\\_scan\\_descriptor\\_info](#)
- [vkd3d\\_shader\\_scan\\_hull\\_shader\\_tessellation\\_info](#)
- [vkd3d\\_shader\\_scan\\_signature\\_info](#)
- [vkd3d\\_shader\\_spirv\\_domain\\_shader\\_target\\_info](#)
- [vkd3d\\_shader\\_spirv\\_target\\_info](#)
- [vkd3d\\_shader\\_transform\\_feedback\\_info](#)
- [vkd3d\\_shader\\_varying\\_map\\_info](#)

#### Parameters

<i>compile_info</i>	A chained structure containing compilation parameters.
<i>out</i>	A pointer to a <a href="#">vkd3d_shader_code</a> structure in which the compiled code will be stored. The compiled shader is allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_shader_code()</a> when no longer needed.
<i>messages</i>	Optional output location for error or informational messages produced by the compiler. This string is null-terminated and UTF-8 encoded. The messages are allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_messages()</a> when no longer needed.
	The messages returned can be regulated with the <i>log_level</i> member of struct <a href="#">vkd3d_shader_compile_info</a> . Regardless of the requested level, if this parameter is NULL, no compilation messages will be returned. If no messages are produced by the compiler, this parameter may receive NULL instead of a valid string pointer.

## Returns

A member of [vkd3d\\_result](#).

## 4.3.5.3 vkd3d\_shader\_convert\_root\_signature()

```
VKD3D_SHADER_API int vkd3d_shader_convert_root_signature (
    struct vkd3d_shader_versioned_root_signature_desc * dst,
    enum vkd3d_shader_root_signature_version version,
    const struct vkd3d_shader_versioned_root_signature_desc * src )
```

Convert a structural representation of a root signature to a different version of structural representation.

This function corresponds to ID3D12VersionedRootSignatureDeserializer::GetRootSignatureDescAtVersion().

## Parameters

<i>dst</i>	A pointer to a <a href="#">vkd3d_shader_versioned_root_signature_desc</a> structure in which the converted signature will be stored. Members of <i>dst</i> may be allocated by vkd3d-shader. The signature should be freed with <a href="#">vkd3d_shader_free_root_signature()</a> when no longer needed.
<i>version</i>	The desired version to convert <i>src</i> to. This version must not be equal to <i>src-&gt;version</i> .
<i>src</i>	Input root signature description.

## Returns

A member of [vkd3d\\_result](#).

## 4.3.5.4 vkd3d\_shader\_find\_signature\_element()

```
VKD3D_SHADER_API struct vkd3d_shader_signature_element * vkd3d_shader_find_signature_element (
    const struct vkd3d_shader_signature * signature,
    const char * semantic_name,
    unsigned int semantic_index,
    unsigned int stream_index )
```

Find a single element of a parsed input signature.

## Parameters

<i>signature</i>	The parsed input signature. This structure is normally populated by <a href="#">vkd3d_shader_parse_input_signature()</a> .
<i>semantic_name</i>	Semantic name of the desired element. This function performs a case-insensitive comparison with respect to the ASCII plane.
<i>semantic_index</i>	Semantic index of the desired element.
<i>stream_index</i>	Geometry shader stream index of the desired element. If the signature is not a geometry shader output signature, this parameter must be set to 0.

**Returns**

A description of the element matching the requested parameters, or NULL if no such element was found. If not NULL, the return value points into the *signature* parameter and should not be explicitly freed.

**4.3.5.5 vkd3d\_shader\_free\_dxbc()**

```
VKD3D_SHADER_API void vkd3d_shader_free_dxbc (
    struct vkd3d_shader_dxbc_desc * dxbc )
```

Free the contents of a [vkd3d\\_shader\\_dxbc\\_desc](#) structure allocated by another vkd3d-shader function, such as [vkd3d\\_shader\\_parse\\_dxbc\(\)](#).

This function may free the [vkd3d\\_shader\\_dxbc\\_desc::sections](#) member, but does not free the structure itself.

**Parameters**

<i>dxbc</i>	The <a href="#">vkd3d_shader_dxbc_desc</a> structure to free.
-------------	---

**Since**

1.7

**4.3.5.6 vkd3d\_shader\_free\_messages()**

```
VKD3D_SHADER_API void vkd3d_shader_free_messages (
    char * messages )
```

Free shader messages allocated by another vkd3d-shader function, such as [vkd3d\\_shader\\_compile\(\)](#).

**Parameters**

<i>messages</i>	Messages to free. This pointer is optional and may be NULL, in which case no action will be taken.
-----------------	--

**4.3.5.7 vkd3d\_shader\_free\_root\_signature()**

```
VKD3D_SHADER_API void vkd3d_shader_free_root_signature (
    struct vkd3d_shader_versioned_root_signature_desc * root_signature )
```

Free a structural representation of a shader root signature allocated by [vkd3d\\_shader\\_convert\\_root\\_signature\(\)](#) or [vkd3d\\_shader\\_parse\\_root\\_signature\(\)](#).

This function may free members of struct [vkd3d\\_shader\\_versioned\\_root\\_signature\\_desc](#), but does not free the structure itself.

## Parameters

<i>root_signature</i>	Signature description to free.
-----------------------	--------------------------------

**4.3.5.8 vkd3d\_shader\_free\_scan\_combined\_resource\_sampler\_info()**

```
VKD3D_SHADER_API void vkd3d_shader_free_scan_combined_resource_sampler_info (
    struct vkd3d_shader_scan_combined_resource_sampler_info * info )
```

Free members of struct [vkd3d\\_shader\\_scan\\_combined\\_resource\\_sampler\\_info](#) allocated by [vkd3d\\_shader\\_scan\(\)](#).

This function may free members of [vkd3d\\_shader\\_scan\\_combined\\_resource\\_sampler\\_info](#), but does not free the structure itself.

## Parameters

<i>info</i>	Combined resource-sampler information to free.
-------------	--

## Since

1.10

**4.3.5.9 vkd3d\_shader\_free\_scan\_descriptor\_info()**

```
VKD3D_SHADER_API void vkd3d_shader_free_scan_descriptor_info (
    struct vkd3d_shader_scan_descriptor_info * scan_descriptor_info )
```

Free members of struct [vkd3d\\_shader\\_scan\\_descriptor\\_info\(\)](#) allocated by [vkd3d\\_shader\\_scan\(\)](#).

This function may free members of [vkd3d\\_shader\\_scan\\_descriptor\\_info](#), but does not free the structure itself.

## Parameters

<i>scan_descriptor_info</i>	Descriptor information to free.
-----------------------------	---------------------------------

**4.3.5.10 vkd3d\_shader\_free\_scan\_signature\_info()**

```
VKD3D_SHADER_API void vkd3d_shader_free_scan_signature_info (
    struct vkd3d_shader_scan_signature_info * info )
```

Free members of struct [vkd3d\\_shader\\_scan\\_signature\\_info](#) allocated by [vkd3d\\_shader\\_scan\(\)](#).

This function may free members of [vkd3d\\_shader\\_scan\\_signature\\_info](#), but does not free the structure itself.

## Parameters

<i>info</i>	Scan information to free.
-------------	---------------------------

## Since

1.9

**4.3.5.11 vkd3d\_shader\_free\_shader\_code()**

```
VKD3D_SHADER_API void vkd3d_shader_free_shader_code (
    struct vkd3d_shader_code * code )
```

Free shader code allocated by another vkd3d-shader function, such as [vkd3d\\_shader\\_compile\(\)](#).

This function frees the [vkd3d\\_shader\\_code::code](#) member, but does not free the structure itself.

## Parameters

<i>code</i>	Code to free.
-------------	---------------

**4.3.5.12 vkd3d\_shader\_free\_shader\_signature()**

```
VKD3D_SHADER_API void vkd3d_shader_free_shader_signature (
    struct vkd3d_shader_signature * signature )
```

Free a structural representation of a shader input signature allocated by [vkd3d\\_shader\\_parse\\_input\\_signature\(\)](#).

This function may free members of struct [vkd3d\\_shader\\_signature](#), but does not free the structure itself.

## Parameters

<i>signature</i>	Signature description to free.
------------------	--------------------------------

**4.3.5.13 vkd3d\_shader\_get\_supported\_source\_types()**

```
VKD3D_SHADER_API enum vkd3d_shader_source_type * vkd3d_shader_get_supported_source_types (
    unsigned int * count )
```

Returns the source types supported, with any target type, by [vkd3d\\_shader\\_compile\(\)](#).

Future versions of the library may introduce additional source types; callers should ignore unrecognised source types.

Use [vkd3d\\_shader\\_get\\_supported\\_target\\_types\(\)](#) to determine which target types are supported for each source type.

#### Parameters

<i>count</i>	Output location for the size, in elements, of the returned array.
--------------	---

#### Returns

Pointer to an array of source types supported by this version of vkd3d-shader. This array may be a pointer to static data in libvkd3d-shader; it should not be freed.

#### 4.3.5.14 vkd3d\_shader\_get\_supported\_target\_types()

```
VKD3D_SHADER_API enum vkd3d_shader_target_type * vkd3d_shader_get_supported_target_types (
    enum vkd3d_shader_source_type source_type,
    unsigned int * count )
```

Returns the target types supported, with the given source type, by [vkd3d\\_shader\\_compile\(\)](#).

Future versions of the library may introduce additional target types; callers should ignore unrecognised target types.

#### Parameters

<i>source_type</i>	Source type for which to enumerate supported target types.
<i>count</i>	Output location for the size, in elements, of the returned array.

#### Returns

Pointer to an array of target types supported by this version of vkd3d-shader. This array may be a pointer to static data in libvkd3d-shader; it should not be freed.

#### 4.3.5.15 vkd3d\_shader\_get\_version()

```
VKD3D_SHADER_API const char * vkd3d_shader_get_version (
    unsigned int * major,
    unsigned int * minor )
```

Returns the current version of this library.

#### Parameters

<i>major</i>	Output location for the major version of this library.
<i>minor</i>	Output location for the minor version of this library.

**Returns**

A human-readable string describing the library name and version. This string is null-terminated and UTF-8 encoded. This may be a pointer to static data in libvkd3d-shader; it should not be freed.

**4.3.5.16 vkd3d\_shader\_parse\_dxbc()**

```
VKD3D_SHADER_API int vkd3d_shader_parse_dxbc (
    const struct vkd3d_shader_code * dxbc,
    uint32_t flags,
    struct vkd3d_shader_dxbc_desc * desc,
    char ** messages )
```

Parse a DXBC blob contained in a [vkd3d\\_shader\\_code](#) structure.

**Parameters**

<i>dxbc</i>	A <a href="#">vkd3d_shader_code</a> structure containing the DXBC blob to parse.
<i>flags</i>	A combination of zero or more elements of enum <a href="#">vkd3d_shader_parse_dxbc_flags</a> .
<i>desc</i>	A <a href="#">vkd3d_shader_dxbc_desc</a> structure describing the contents of the DXBC blob. Its <a href="#">vkd3d_shader_dxbc_section_desc</a> structures will contain pointers into the input blob; its contents are only valid while the input blob is valid. The contents of this structure should be freed with <a href="#">vkd3d_shader_free_dxbc()</a> when no longer needed.
<i>messages</i>	Optional output location for error or informational messages produced by the parser. This string is null-terminated and UTF-8 encoded. The messages are allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_messages()</a> when no longer needed. If no messages are produced by the parser, this parameter may receive NULL instead of a valid string pointer.

**Returns**

A member of [vkd3d\\_result](#).

**Since**

1.7

**4.3.5.17 vkd3d\_shader\_parse\_input\_signature()**

```
VKD3D_SHADER_API int vkd3d_shader_parse_input_signature (
    const struct vkd3d_shader_code * dxbc,
    struct vkd3d_shader_signature * signature,
    char ** messages )
```

Read the input signature of a compiled DXBC shader, returning a structural description which can be easily parsed by C code.

This function parses a compiled shader. To parse a standalone root signature, use [vkd3d\\_shader\\_parse\\_root\\_signature\(\)](#).

This function only parses DXBC shaders, and only retrieves the input signature. To retrieve signatures from other shader types, or other signature types, use [vkd3d\\_shader\\_scan\(\)](#) and struct [vkd3d\\_shader\\_scan\\_signature\\_info](#). This function returns the same input signature that is returned in struct [vkd3d\\_shader\\_scan\\_signature\\_info](#) for dxbc-tpf shaders, but may return different information for dxbc-dxil shaders.

## Parameters

<i>dxbc</i>	Compiled byte code, in DXBC format.
<i>signature</i>	Output location in which the parsed root signature will be stored. Members of <i>signature</i> may be allocated by vkd3d-shader. The signature should be freed with <a href="#">vkd3d_shader_free_shader_signature()</a> when no longer needed. The signature may contain pointers into the input shader, and should only be accessed while the input shader remains valid.
<i>messages</i>	Optional output location for error or informational messages produced by the parser. This string is null-terminated and UTF-8 encoded. The messages are allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_messages()</a> when no longer needed. If no messages are produced by the parser, this parameter may receive NULL instead of a valid string pointer.

## Returns

A member of [vkd3d\\_result](#).

## 4.3.5.18 vkd3d\_shader\_parse\_root\_signature()

```
VKD3D_SHADER_API int vkd3d_shader_parse_root_signature (
    const struct vkd3d_shader_code * dxbc,
    struct vkd3d_shader_versioned_root_signature_desc * root_signature,
    char ** messages )
```

Convert a byte code description of a shader root signature to a structural description which can be easily parsed by C code.

This function corresponds to ID3D12VersionedRootSignatureDeserializer::GetUnconvertedRootSignatureDesc().

This function performs the reverse transformation of [vkd3d\\_shader\\_serialize\\_root\\_signature\(\)](#).

This function parses a standalone root signature, and should not be confused with [vkd3d\\_shader\\_parse\\_input\\_signature\(\)](#).

## Parameters

<i>dxbc</i>	Compiled byte code, in DXBC format.
<i>root_signature</i>	Output location in which the decompiled root signature will be stored. Members of <i>root_signature</i> may be allocated by vkd3d-shader. The signature should be freed with <a href="#">vkd3d_shader_free_root_signature()</a> when no longer needed.
<i>messages</i>	Optional output location for error or informational messages produced by the parser. This string is null-terminated and UTF-8 encoded. The messages are allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_messages()</a> when no longer needed. If no messages are produced by the parser, this parameter may receive NULL instead of a valid string pointer.

## Returns

A member of [vkd3d\\_result](#).

## 4.3.5.19 vkd3d\_shader\_preprocess()

```
VKD3D_SHADER_API int vkd3d_shader_preprocess (
    const struct vkd3d\_shader\_compile\_info * compile_info,
    struct vkd3d\_shader\_code * out,
    char ** messages )
```

Preprocess the given source code.

This function supports the following chained structures:

- [vkd3d\\_shader\\_preprocess\\_info](#)

## Parameters

<i>compile_info</i>	A chained structure containing compilation parameters.
<i>out</i>	A pointer to a <a href="#">vkd3d_shader_code</a> structure in which the preprocessed code will be stored. The preprocessed shader is allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_shader_code()</a> when no longer needed.
<i>messages</i>	Optional output location for error or informational messages produced by the preprocessor. This string is null-terminated and UTF-8 encoded. The messages are allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_messages()</a> when no longer needed. The messages returned can be regulated with the <i>log_level</i> member of struct <a href="#">vkd3d_shader_compile_info</a> . Regardless of the requested level, if this parameter is NULL, no compilation messages will be returned. If no messages are produced by the preprocessor, this parameter may receive NULL instead of a valid string pointer.

## Returns

A member of [vkd3d\\_result](#).

## Since

1.3

## 4.3.5.20 vkd3d\_shader\_scan()

```
VKD3D_SHADER_API int vkd3d_shader_scan (
    const struct vkd3d\_shader\_compile\_info * compile_info,
    char ** messages )
```

Parse shader source code or byte code, returning various types of requested information.

The *source\_type* member of *compile\_info* must be set to the type of the shader.

The *target\_type* member may be set to VKD3D\_SHADER\_TARGET\_NONE, in which case [vkd3d\\_shader\\_scan\(\)](#) will return information about the shader in isolation. Alternatively, it may be set to a valid compilation target for the shader, in which case [vkd3d\\_shader\\_scan\(\)](#) will return information that reflects the interface for a shader as it will be compiled to that target. In this case other chained structures may be appended to *compile\_info* as they would be passed to [vkd3d\\_shader\\_compile\(\)](#), and interpreted accordingly, such as [vkd3d\\_shader\\_spirv\\_target\\_info](#).

(For a hypothetical example, suppose the source shader distinguishes float and integer texture data, but the target environment does not support integer textures. In this case [vkd3d\\_shader\\_compile\(\)](#) might translate integer operations to float. Accordingly using VKD3D\_SHADER\_TARGET\_NONE would accurately report whether the texture expects integer or float data, but using the relevant specific target type would report VKD3D\_SHADER\_RESOURCE\_DATA\_FLOAT.)

Currently this function supports the following code types:

- VKD3D\_SHADER\_SOURCE\_DXBC\_TPF
- VKD3D\_SHADER\_SOURCE\_D3D\_BYTECODE

#### Parameters

<i>compile_info</i>	<p>A chained structure containing scan parameters. The scanner supports the following chained structures:</p> <ul style="list-style-type: none"> <li>• <a href="#">vkd3d_shader_scan_combined_resource_sampler_info</a></li> <li>• <a href="#">vkd3d_shader_scan_descriptor_info</a></li> <li>• <a href="#">vkd3d_shader_scan_hull_shader_tessellation_info</a></li> <li>• <a href="#">vkd3d_shader_scan_signature_info</a></li> </ul> <p>Although the <i>compile_info</i> parameter is read-only, chained structures passed to this function need not be, and may serve as output parameters, depending on their structure type.</p>
<i>messages</i>	<p>Optional output location for error or informational messages produced by the parser. This string is null-terminated and UTF-8 encoded. The messages are allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_messages()</a> when no longer needed. The messages returned can be regulated with the <i>log_level</i> member of struct <a href="#">vkd3d_shader_compile_info</a>. Regardless of the requested level, if this parameter is NULL, no compilation messages will be returned. If no messages are produced by the parser, this parameter may receive NULL instead of a valid string pointer.</p>

#### Returns

A member of [vkd3d\\_result](#).

#### 4.3.5.21 vkd3d\_shader\_serialize\_dxbc()

```
VKD3D_SHADER_API int vkd3d_shader_serialize_dxbc (
    size_t section_count,
```

```
const struct vkd3d_shader_dxbc_section_desc * sections,
struct vkd3d_shader_code * dxbc,
char ** messages )
```

Serialize a DXBC description into a blob stored in a [vkd3d\\_shader\\_code](#) structure.

#### Parameters

<i>section_count</i>	The number of DXBC sections to serialize.
<i>sections</i>	An array of <a href="#">vkd3d_shader_dxbc_section_desc</a> structures to serialize.
<i>dxbc</i>	A pointer to a <a href="#">vkd3d_shader_code</a> structure in which the serialized blob will be stored. The output blob is allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_shader_code()</a> when no longer needed.
<i>messages</i>	Optional output location for error or informational messages produced by the serializer. This string is null-terminated and UTF-8 encoded. The messages are allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_messages()</a> when no longer needed. If no messages are produced by the serializer, this parameter may receive NULL instead of a valid string pointer.

#### Returns

A member of [vkd3d\\_result](#).

#### Since

1.7

#### 4.3.5.22 vkd3d\_shader\_serialize\_root\_signature()

```
VKD3D_SHADER_API int vkd3d_shader_serialize_root_signature (
    const struct vkd3d_shader_versioned_root_signature_desc * root_signature,
    struct vkd3d_shader_code * dxbc,
    char ** messages )
```

Convert a structural description of a shader root signature to a byte code format capable of being read by ID3D12↵ Device::CreateRootSignature.

The compiled signature is compatible with Microsoft D3D 12.

This function corresponds to D3D12SerializeVersionedRootSignature().

#### Parameters

<i>root_signature</i>	Description of the root signature.
<i>dxbc</i>	A pointer to a <a href="#">vkd3d_shader_code</a> structure in which the compiled code will be stored. The compiled signature is allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_shader_code()</a> when no longer needed.
<i>messages</i>	Optional output location for error or informational messages produced by the serializer. This string is null-terminated and UTF-8 encoded. The messages are allocated by vkd3d-shader and should be freed with <a href="#">vkd3d_shader_free_messages()</a> when no longer needed.
Generated by Doxygen	If no messages are produced by the serializer, this parameter may receive NULL instead of a valid string pointer.

**Returns**

A member of [vkd3d\\_result](#).

**4.3.5.23 vkd3d\_shader\_set\_log\_callback()**

```
VKD3D_SHADER_API void vkd3d_shader_set_log_callback (
    PFN_vkd3d_log callback )
```

Set a callback to be called when vkd3d-shader outputs debug logging.

If NULL, or if this function has not been called, libvkd3d-shader will print all enabled log output to stderr.

**Parameters**

<i>callback</i>	Callback function to set.
-----------------	---------------------------

**Since**

1.4

**4.4 vkd3d\_shader.h**

[Go to the documentation of this file.](#)

```
1 /*
2  * Copyright 2017-2019 Józef Kucia for CodeWeavers
3  *
4  * This library is free software; you can redistribute it and/or
5  * modify it under the terms of the GNU Lesser General Public
6  * License as published by the Free Software Foundation; either
7  * version 2.1 of the License, or (at your option) any later version.
8  *
9  * This library is distributed in the hope that it will be useful,
10 * but WITHOUT ANY WARRANTY; without even the implied warranty of
11 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
12 * Lesser General Public License for more details.
13 *
14 * You should have received a copy of the GNU Lesser General Public
15 * License along with this library; if not, write to the Free Software
16 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
17 */
18
19 #ifndef __VKD3D_SHADER_H
20 #define __VKD3D_SHADER_H
21
22 #include <stdbool.h>
23 #include <stdint.h>
24 #include <stddef.h>
25 #include <vkd3d_types.h>
26
27 #ifdef __cplusplus
28 extern "C" {
29 #endif /* __cplusplus */
30
43 enum vkd3d_shader_api_version
44 {
45     VKD3D_SHADER_API_VERSION_1_0,
46     VKD3D_SHADER_API_VERSION_1_1,
47     VKD3D_SHADER_API_VERSION_1_2,
48     VKD3D_SHADER_API_VERSION_1_3,
49     VKD3D_SHADER_API_VERSION_1_4,
50     VKD3D_SHADER_API_VERSION_1_5,
51     VKD3D_SHADER_API_VERSION_1_6,
52     VKD3D_SHADER_API_VERSION_1_7,
```

```

53     VKD3D_SHADER_API_VERSION_1_8,
54     VKD3D_SHADER_API_VERSION_1_9,
55     VKD3D_SHADER_API_VERSION_1_10,
56     VKD3D_SHADER_API_VERSION_1_11,
57     VKD3D_SHADER_API_VERSION_1_12,
58     VKD3D_SHADER_API_VERSION_1_13,
59     VKD3D_SHADER_API_VERSION_1_14,
60     VKD3D_SHADER_API_VERSION_1_15,
61
62     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_API_VERSION),
63 };
64
65 enum vkd3d_shader_structure_type
66 {
67     VKD3D_SHADER_STRUCTURE_TYPE_COMPILE_INFO,
68     VKD3D_SHADER_STRUCTURE_TYPE_INTERFACE_INFO,
69     VKD3D_SHADER_STRUCTURE_TYPE_SCAN_DESCRIPTOR_INFO,
70     VKD3D_SHADER_STRUCTURE_TYPE_SPIRV_DOMAIN_SHADER_TARGET_INFO,
71     VKD3D_SHADER_STRUCTURE_TYPE_SPIRV_TARGET_INFO,
72     VKD3D_SHADER_STRUCTURE_TYPE_TRANSFORM_FEEDBACK_INFO,
73
74     VKD3D_SHADER_STRUCTURE_TYPE_HLSL_SOURCE_INFO,
75     VKD3D_SHADER_STRUCTURE_TYPE_PREPROCESS_INFO,
76     VKD3D_SHADER_STRUCTURE_TYPE_DESCRIPTOR_OFFSET_INFO,
77     VKD3D_SHADER_STRUCTURE_TYPE_SCAN_SIGNATURE_INFO,
78     VKD3D_SHADER_STRUCTURE_TYPE_VARYING_MAP_INFO,
79     VKD3D_SHADER_STRUCTURE_TYPE_SCAN_COMBINED_RESOURCE_SAMPLER_INFO,
80     VKD3D_SHADER_STRUCTURE_TYPE_PARAMETER_INFO,
81     VKD3D_SHADER_STRUCTURE_TYPE_SCAN_HULL_SHADER_TESSELLATION_INFO,
82
83     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_STRUCTURE_TYPE),
84 };
85
86 enum vkd3d_shader_compile_option_buffer_uav
87 {
88     VKD3D_SHADER_COMPILE_OPTION_BUFFER_UAV_STORAGE_TEXEL_BUFFER = 0x00000000,
89     VKD3D_SHADER_COMPILE_OPTION_BUFFER_UAV_STORAGE_BUFFER = 0x00000001,
90
91     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPILE_OPTION_BUFFER_UAV),
92 };
93
94 enum vkd3d_shader_compile_option_typed_uav
95 {
96     VKD3D_SHADER_COMPILE_OPTION_TYPED_UAV_READ_FORMAT_R32 = 0x00000000,
97     VKD3D_SHADER_COMPILE_OPTION_TYPED_UAV_READ_FORMAT_UNKNOWN = 0x00000001,
98
99     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPILE_OPTION_TYPED_UAV),
100 };
101
102 enum vkd3d_shader_compile_option_formatting_flags
103 {
104     VKD3D_SHADER_COMPILE_OPTION_FORMATTING_NONE = 0x00000000,
105     VKD3D_SHADER_COMPILE_OPTION_FORMATTING_COLOUR = 0x00000001,
106     VKD3D_SHADER_COMPILE_OPTION_FORMATTING_INDENT = 0x00000002,
107     VKD3D_SHADER_COMPILE_OPTION_FORMATTING_OFFSETS = 0x00000004,
108     VKD3D_SHADER_COMPILE_OPTION_FORMATTING_HEADER = 0x00000008,
109     VKD3D_SHADER_COMPILE_OPTION_FORMATTING_RAW_IDS = 0x00000010,
110     VKD3D_SHADER_COMPILE_OPTION_FORMATTING_IO_SIGNATURES = 0x00000020,
111
112     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPILE_OPTION_FORMATTING_FLAGS),
113 };
114
115 enum vkd3d_shader_compile_option_pack_matrix_order
116 {
117     VKD3D_SHADER_COMPILE_OPTION_PACK_MATRIX_ROW_MAJOR = 0x00000001,
118     VKD3D_SHADER_COMPILE_OPTION_PACK_MATRIX_COLUMN_MAJOR = 0x00000002,
119
120     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPILE_OPTION_PACK_MATRIX_ORDER),
121 };
122
123 enum vkd3d_shader_compile_option_backward_compatibility
124 {
125     VKD3D_SHADER_COMPILE_OPTION_BACKCOMPAT_MAP_SEMANTIC_NAMES = 0x00000001,
126     VKD3D_SHADER_COMPILE_OPTION_DOUBLE_AS_FLOAT_ALIAS = 0x00000002,
127
128     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPILE_OPTION_BACKWARD_COMPATIBILITY),
129 };
130
131 enum vkd3d_shader_compile_option_fragment_coordinate_origin
132 {
133     VKD3D_SHADER_COMPILE_OPTION_FRAGMENT_COORDINATE_ORIGIN_UPPER_LEFT = 0x00000000,
134     VKD3D_SHADER_COMPILE_OPTION_FRAGMENT_COORDINATE_ORIGIN_LOWER_LEFT = 0x00000001,
135
136     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPILE_OPTION_FRAGMENT_COORDINATE_ORIGIN),
137 };
138
139 enum vkd3d_shader_compile_option_feature_flags

```

```

234 {
238     VKD3D_SHADER_COMPILE_OPTION_FEATURE_INT64          = 0x00000001,
242     VKD3D_SHADER_COMPILE_OPTION_FEATURE_FLOAT64       = 0x00000002,
251     VKD3D_SHADER_COMPILE_OPTION_FEATURE_WAVE_OPS      = 0x00000004,
255     VKD3D_SHADER_COMPILE_OPTION_FEATURE_ZERO_INITIALIZE_WORKGROUP_MEMORY = 0x00000008,
256
257     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPILE_OPTION_FEATURE_FLAGS),
258 };
259
265 enum vkd3d_shader_parse_dxbc_flags
266 {
269     VKD3D_SHADER_PARSE_DXBC_IGNORE_CHECKSUM            = 0x00000001,
270
271     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_PARSE_DXBC_FLAGS),
272 };
273
274 enum vkd3d_shader_compile_option_name
275 {
283     VKD3D_SHADER_COMPILE_OPTION_STRIP_DEBUG = 0x00000001,
285     VKD3D_SHADER_COMPILE_OPTION_BUFFER_UAV  = 0x00000002,
287     VKD3D_SHADER_COMPILE_OPTION_FORMATTING  = 0x00000003,
289     VKD3D_SHADER_COMPILE_OPTION_API_VERSION = 0x00000004,
291     VKD3D_SHADER_COMPILE_OPTION_TYPED_UAV   = 0x00000005,
303     VKD3D_SHADER_COMPILE_OPTION_WRITE_TESS_GEOM_POINT_SIZE = 0x00000006,
312     VKD3D_SHADER_COMPILE_OPTION_PACK_MATRIX_ORDER = 0x00000007,
320     VKD3D_SHADER_COMPILE_OPTION_BACKWARD_COMPATIBILITY = 0x00000008,
330     VKD3D_SHADER_COMPILE_OPTION_FRAGMENT_COORDINATE_ORIGIN = 0x00000009,
340     VKD3D_SHADER_COMPILE_OPTION_FEATURE = 0x0000000a,
351     VKD3D_SHADER_COMPILE_OPTION_CHILD_EFFECT = 0x0000000b,
362     VKD3D_SHADER_COMPILE_OPTION_WARN_IMPLICIT_TRUNCATION = 0x0000000c,
370     VKD3D_SHADER_COMPILE_OPTION_INCLUDE_EMPTY_BUFFERS_IN_EFFECTS = 0x0000000d,
371
372     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPILE_OPTION_NAME),
373 };
374
380 struct vkd3d_shader_compile_option
381 {
383     enum vkd3d_shader_compile_option_name name;
388     unsigned int value;
389 };
390
392 enum vkd3d_shader_visibility
393 {
395     VKD3D_SHADER_VISIBILITY_ALL = 0,
397     VKD3D_SHADER_VISIBILITY_VERTEX = 1,
399     VKD3D_SHADER_VISIBILITY_HULL = 2,
401     VKD3D_SHADER_VISIBILITY_DOMAIN = 3,
403     VKD3D_SHADER_VISIBILITY_GEOMETRY = 4,
405     VKD3D_SHADER_VISIBILITY_PIXEL = 5,
406
408     VKD3D_SHADER_VISIBILITY_COMPUTE = 1000000000,
409
410     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_VISIBILITY),
411 };
412
414 struct vkd3d_shader_code
415 {
423     const void *code;
425     size_t size;
426 };
427
429 enum vkd3d_shader_descriptor_type
430 {
435     VKD3D_SHADER_DESCRIPTOR_TYPE_SRV = 0x0,
440     VKD3D_SHADER_DESCRIPTOR_TYPE_UAV = 0x1,
445     VKD3D_SHADER_DESCRIPTOR_TYPE_CBV = 0x2,
450     VKD3D_SHADER_DESCRIPTOR_TYPE_SAMPLER = 0x3,
451
452     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_DESCRIPTOR_TYPE),
453 };
454
459 struct vkd3d_shader_descriptor_binding
460 {
465     unsigned int set;
467     unsigned int binding;
473     unsigned int count;
474 };
475
476 enum vkd3d_shader_binding_flag
477 {
478     VKD3D_SHADER_BINDING_FLAG_BUFFER = 0x00000001,
479     VKD3D_SHADER_BINDING_FLAG_IMAGE = 0x00000002,
480
481     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_BINDING_FLAG),
482 };
483
500 enum vkd3d_shader_fog_fragment_mode

```

```

501 {
502     VKD3D_SHADER_FOG_FRAGMENT_NONE = 0x0,
503     VKD3D_SHADER_FOG_FRAGMENT_EXP = 0x1,
504     VKD3D_SHADER_FOG_FRAGMENT_EXP2 = 0x2,
505     VKD3D_SHADER_FOG_FRAGMENT_LINEAR = 0x3,
506 };
507
508 enum vkd3d_shader_fog_source
509 {
510     VKD3D_SHADER_FOG_SOURCE_FOG = 0x0,
511     VKD3D_SHADER_FOG_SOURCE_FOG_OR_SPECULAR_W = 0x1,
512     VKD3D_SHADER_FOG_SOURCE_Z = 0x2,
513     VKD3D_SHADER_FOG_SOURCE_W = 0x3,
514 };
515
516 enum vkd3d_shader_parameter_type
517 {
518     VKD3D_SHADER_PARAMETER_TYPE_UNKNOWN,
519     VKD3D_SHADER_PARAMETER_TYPE_IMMEDIATE_CONSTANT,
520     VKD3D_SHADER_PARAMETER_TYPE_SPECIALIZATION_CONSTANT,
521     VKD3D_SHADER_PARAMETER_TYPE_BUFFER,
522 };
523
524 enum vkd3d_shader_parameter_data_type
525 {
526     VKD3D_SHADER_PARAMETER_DATA_TYPE_UNKNOWN,
527     VKD3D_SHADER_PARAMETER_DATA_TYPE_UINT32,
528     VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32,
529     VKD3D_SHADER_PARAMETER_DATA_TYPE_FLOAT32_VEC4,
530 };
531
532 enum vkd3d_shader_parameter_name
533 {
534     VKD3D_SHADER_PARAMETER_NAME_UNKNOWN,
535     VKD3D_SHADER_PARAMETER_NAME_RASTERIZER_SAMPLE_COUNT,
536     VKD3D_SHADER_PARAMETER_NAME_ALPHA_TEST_FUNC,
537     VKD3D_SHADER_PARAMETER_NAME_ALPHA_TEST_REF,
538     VKD3D_SHADER_PARAMETER_NAME_FLAT_INTERPOLATION,
539     VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_MASK,
540     VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_0,
541     VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_1,
542     VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_2,
543     VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_3,
544     VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_4,
545     VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_5,
546     VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_6,
547     VKD3D_SHADER_PARAMETER_NAME_CLIP_PLANE_7,
548     VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE,
549     VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE_MIN,
550     VKD3D_SHADER_PARAMETER_NAME_POINT_SIZE_MAX,
551     VKD3D_SHADER_PARAMETER_NAME_POINT_SPRITE,
552     VKD3D_SHADER_PARAMETER_NAME_FOG_FRAGMENT_MODE,
553     VKD3D_SHADER_PARAMETER_NAME_FOG_COLOUR,
554     VKD3D_SHADER_PARAMETER_NAME_FOG_END,
555     VKD3D_SHADER_PARAMETER_NAME_FOG_SCALE,
556     VKD3D_SHADER_PARAMETER_NAME_FOG_SOURCE,
557 };
558
559 struct vkd3d_shader_parameter_immediate_constant
560 {
561     union
562     {
563         uint32_t u32;
564         float f32;
565     } u;
566 };
567
568 struct vkd3d_shader_parameter_immediate_constant1
569 {
570     union
571     {
572         uint32_t u32;
573         float f32;
574         float f32_vec4[4];
575         void *_pointer_pad;
576         uint32_t _pad[4];
577     } u;
578 };

```

```

1002 };
1003
1008 struct vkd3d_shader_parameter_specialization_constant
1009 {
1017     uint32_t id;
1018 };
1019
1024 struct vkd3d_shader_parameter_buffer
1025 {
1030     unsigned int set;
1032     unsigned int binding;
1034     uint32_t offset;
1035 };
1036
1049 struct vkd3d_shader_parameter
1050 {
1051     enum vkd3d_shader_parameter_name name;
1052     enum vkd3d_shader_parameter_type type;
1053     enum vkd3d_shader_parameter_data_type data_type;
1054     union
1055     {
1056         struct vkd3d_shader_parameter_immediate_constant immediate_constant;
1057         struct vkd3d_shader_parameter_specialization_constant specialization_constant;
1058     } u;
1059 };
1060
1078 struct vkd3d_shader_parameter1
1079 {
1081     enum vkd3d_shader_parameter_name name;
1083     enum vkd3d_shader_parameter_type type;
1088     enum vkd3d_shader_parameter_data_type data_type;
1089     union
1090     {
1095         struct vkd3d_shader_parameter_immediate_constant1 immediate_constant;
1100         struct vkd3d_shader_parameter_specialization_constant specialization_constant;
1105         struct vkd3d_shader_parameter_buffer buffer;
1106         void *_pointer_pad;
1107         uint32_t _pad[4];
1108     } u;
1109 };
1110
1120 enum vkd3d_shader_d3dbc_constant_register
1121 {
1123     VKD3D_SHADER_D3DBC_FLOAT_CONSTANT_REGISTER = 0x0,
1125     VKD3D_SHADER_D3DBC_INT_CONSTANT_REGISTER = 0x1,
1127     VKD3D_SHADER_D3DBC_BOOL_CONSTANT_REGISTER = 0x2,
1128
1129     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_D3DBC_CONSTANT_REGISTER),
1130 };
1131
1147 struct vkd3d_shader_resource_binding
1148 {
1150     enum vkd3d_shader_descriptor_type type;
1155     unsigned int register_space;
1164     unsigned int register_index;
1166     enum vkd3d_shader_visibility shader_visibility;
1168     unsigned int flags;
1169
1171     struct vkd3d_shader_descriptor_binding binding;
1172 };
1173
1174 #define VKD3D_SHADER_DUMMY_SAMPLER_INDEX ~0u
1175
1182 struct vkd3d_shader_combined_resource_sampler
1183 {
1188     unsigned int resource_space;
1190     unsigned int resource_index;
1195     unsigned int sampler_space;
1197     unsigned int sampler_index;
1199     enum vkd3d_shader_visibility shader_visibility;
1201     unsigned int flags;
1202
1204     struct vkd3d_shader_descriptor_binding binding;
1205 };
1206
1212 struct vkd3d_shader_uav_counter_binding
1213 {
1218     unsigned int register_space;
1220     unsigned int register_index;
1222     enum vkd3d_shader_visibility shader_visibility;
1223
1225     struct vkd3d_shader_descriptor_binding binding;
1226     unsigned int offset;
1227 };
1228
1235 struct vkd3d_shader_push_constant_buffer
1236 {

```

```

1241     unsigned int register_space;
1242     unsigned int register_index;
1243     enum vkd3d_shader_visibility shader_visibility;
1244 };
1245
1246     unsigned int offset;
1247     unsigned int size;
1248 };
1249
1250 struct vkd3d_shader_interface_info
1251 {
1252     enum vkd3d_shader_structure_type type;
1253     const void *next;
1254
1255     const struct vkd3d_shader_resource_binding *bindings;
1256     unsigned int binding_count;
1257
1258     const struct vkd3d_shader_push_constant_buffer *push_constant_buffers;
1259     unsigned int push_constant_buffer_count;
1260
1261     const struct vkd3d_shader_combined_resource_sampler *combined_samplers;
1262     unsigned int combined_sampler_count;
1263
1264     const struct vkd3d_shader_uav_counter_binding *uav_counters;
1265     unsigned int uav_counter_count;
1266 };
1267
1268 struct vkd3d_shader_transform_feedback_element
1269 {
1270     unsigned int stream_index;
1271     const char *semantic_name;
1272     unsigned int semantic_index;
1273     uint8_t component_index;
1274     uint8_t component_count;
1275     uint8_t output_slot;
1276 };
1277
1278 /* Extends vkd3d_shader_interface_info. */
1279 struct vkd3d_shader_transform_feedback_info
1280 {
1281     enum vkd3d_shader_structure_type type;
1282     const void *next;
1283
1284     const struct vkd3d_shader_transform_feedback_element *elements;
1285     unsigned int element_count;
1286     const unsigned int *buffer_strides;
1287     unsigned int buffer_stride_count;
1288 };
1289
1290 struct vkd3d_shader_descriptor_offset
1291 {
1292     unsigned int static_offset;
1293     unsigned int dynamic_offset_index;
1294 };
1295
1296 struct vkd3d_shader_descriptor_offset_info
1297 {
1298     enum vkd3d_shader_structure_type type;
1299     const void *next;
1300
1301     unsigned int descriptor_table_offset;
1302     unsigned int descriptor_table_count;
1303
1304     const struct vkd3d_shader_descriptor_offset *binding_offsets;
1305
1306     const struct vkd3d_shader_descriptor_offset *uav_counter_offsets;
1307 };
1308
1309 enum vkd3d_shader_source_type
1310 {
1311     VKD3D_SHADER_SOURCE_NONE,
1312     VKD3D_SHADER_SOURCE_DXBC_TPF,
1313     VKD3D_SHADER_SOURCE_HLSL,
1314     VKD3D_SHADER_SOURCE_D3D_BYTECODE,
1315     VKD3D_SHADER_SOURCE_DXBC_DXIL,
1316     VKD3D_SHADER_SOURCE_FX,
1317     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_SOURCE_TYPE),
1318 };
1319
1320 enum vkd3d_shader_target_type
1321 {
1322     VKD3D_SHADER_TARGET_NONE,
1323     VKD3D_SHADER_TARGET_SPIRV_BINARY,
1324     VKD3D_SHADER_TARGET_SPIRV_TEXT,
1325     VKD3D_SHADER_TARGET_D3D_ASM,
1326     VKD3D_SHADER_TARGET_D3D_BYTECODE,
1327     VKD3D_SHADER_TARGET_DXBC_TPF,
1328 };

```

```

1468     VKD3D_SHADER_TARGET_GLSL,
1473     VKD3D_SHADER_TARGET_FX,
1477     VKD3D_SHADER_TARGET_MSL,
1478
1479     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_TARGET_TYPE),
1480 };
1481
1482 enum vkd3d_shader_log_level
1483 {
1484     VKD3D_SHADER_LOG_NONE,
1485     VKD3D_SHADER_LOG_ERROR,
1486     VKD3D_SHADER_LOG_WARNING,
1487     VKD3D_SHADER_LOG_INFO,
1488
1489     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_LOG_LEVEL),
1490 };
1491
1492 struct vkd3d_shader_compile_info
1493 {
1494     enum vkd3d_shader_structure_type type;
1495     const void *next;
1496
1497     struct vkd3d_shader_code source;
1498
1499     enum vkd3d_shader_source_type source_type;
1500     enum vkd3d_shader_target_type target_type;
1501
1502     const struct vkd3d_shader_compile_option *options;
1503     unsigned int option_count;
1504
1505     enum vkd3d_shader_log_level log_level;
1506     const char *source_name;
1507 };
1508
1509 enum vkd3d_shader_spirv_environment
1510 {
1511     VKD3D_SHADER_SPIRV_ENVIRONMENT_NONE,
1512     VKD3D_SHADER_SPIRV_ENVIRONMENT_OPENGL_4_5,
1513     VKD3D_SHADER_SPIRV_ENVIRONMENT_VULKAN_1_0, /* default target */
1514     VKD3D_SHADER_SPIRV_ENVIRONMENT_VULKAN_1_1,
1515
1516     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_SPIRV_ENVIRONMENT),
1517 };
1518
1519 enum vkd3d_shader_spirv_extension
1520 {
1521     VKD3D_SHADER_SPIRV_EXTENSION_NONE,
1522     VKD3D_SHADER_SPIRV_EXTENSION_EXT_DEMOTE_TO_HELPER_INVOCATION,
1523     VKD3D_SHADER_SPIRV_EXTENSION_EXT_DESCRIPTOR_INDEXING,
1524     VKD3D_SHADER_SPIRV_EXTENSION_EXT_STENCIL_EXPORT,
1525     VKD3D_SHADER_SPIRV_EXTENSION_EXT_VIEWPORT_INDEX_LAYER,
1526     VKD3D_SHADER_SPIRV_EXTENSION_EXT_FRAGMENT_SHADER_INTERLOCK,
1527
1528     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_SPIRV_EXTENSION),
1529 };
1530
1531 /* Extends vkd3d_shader_compile_info. */
1532 struct vkd3d_shader_spirv_target_info
1533 {
1534     enum vkd3d_shader_structure_type type;
1535     const void *next;
1536
1537     const char *entry_point; /* "main" if NULL. */
1538
1539     enum vkd3d_shader_spirv_environment environment;
1540
1541     const enum vkd3d_shader_spirv_extension *extensions;
1542     unsigned int extension_count;
1543
1544     const struct vkd3d_shader_parameter *parameters;
1545     unsigned int parameter_count;
1546
1547     bool dual_source_blending;
1548     const unsigned int *output_swizzles;
1549     unsigned int output_swizzle_count;
1550 };
1551
1552 enum vkd3d_shader_tessellator_output_primitive
1553 {
1554     VKD3D_SHADER_TESSELLATOR_OUTPUT_POINT = 0x1,
1555     VKD3D_SHADER_TESSELLATOR_OUTPUT_LINE = 0x2,
1556     VKD3D_SHADER_TESSELLATOR_OUTPUT_TRIANGLE_CW = 0x3,
1557     VKD3D_SHADER_TESSELLATOR_OUTPUT_TRIANGLE_CCW = 0x4,
1558
1559     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_TESSELLATOR_OUTPUT_PRIMITIVE),
1560 };
1561

```

```

1605 enum vkd3d_shader_tessellator_partitioning
1606 {
1607     VKD3D_SHADER_TESSELLATOR_PARTITIONING_INTEGER = 0x1,
1608     VKD3D_SHADER_TESSELLATOR_PARTITIONING_POW2 = 0x2,
1609     VKD3D_SHADER_TESSELLATOR_PARTITIONING_FRACTIONAL_ODD = 0x3,
1610     VKD3D_SHADER_TESSELLATOR_PARTITIONING_FRACTIONAL_EVEN = 0x4,
1611
1612     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_TESSELLATOR_PARTITIONING),
1613 };
1614
1615 /* Extends vkd3d_shader_spirv_target_info. */
1616 struct vkd3d_shader_spirv_domain_shader_target_info
1617 {
1618     enum vkd3d_shader_structure_type type;
1619     const void *next;
1620
1621     enum vkd3d_shader_tessellator_output_primitive output_primitive;
1622     enum vkd3d_shader_tessellator_partitioning partitioning;
1623 };
1624
1629 struct vkd3d_shader_macro
1630 {
1636     const char *name;
1642     const char *value;
1643 };
1644
1673 typedef int (*PFN_vkd3d_shader_open_include)(const char *filename, bool local,
1674     const char *parent_data, void *context, struct vkd3d_shader_code *out);
1689 typedef void (*PFN_vkd3d_shader_close_include)(const struct vkd3d_shader_code *code, void *context);
1690
1702 struct vkd3d_shader_preprocess_info
1703 {
1705     enum vkd3d_shader_structure_type type;
1707     const void *next;
1708
1717     const struct vkd3d_shader_macro *macros;
1719     unsigned int macro_count;
1720
1733     PFN_vkd3d_shader_open_include pfn_open_include;
1743     PFN_vkd3d_shader_close_include pfn_close_include;
1748     void *include_context;
1749 };
1750
1762 struct vkd3d_shader_hlsl_source_info
1763 {
1765     enum vkd3d_shader_structure_type type;
1767     const void *next;
1768
1775     const char *entry_point;
1776     struct vkd3d_shader_code secondary_code;
1781     const char *profile;
1782 };
1783
1784 /* root signature 1.0 */
1785 enum vkd3d_shader_filter
1786 {
1787     VKD3D_SHADER_FILTER_MIN_MAG_MIP_POINT = 0x000,
1788     VKD3D_SHADER_FILTER_MIN_MAG_POINT_MIP_LINEAR = 0x001,
1789     VKD3D_SHADER_FILTER_MIN_POINT_MAG_LINEAR_MIP_POINT = 0x004,
1790     VKD3D_SHADER_FILTER_MIN_POINT_MAG_MIP_LINEAR = 0x005,
1791     VKD3D_SHADER_FILTER_MIN_LINEAR_MAG_MIP_POINT = 0x010,
1792     VKD3D_SHADER_FILTER_MIN_LINEAR_MAG_POINT_MIP_LINEAR = 0x011,
1793     VKD3D_SHADER_FILTER_MIN_MAG_LINEAR_MIP_POINT = 0x014,
1794     VKD3D_SHADER_FILTER_MIN_MAG_MIP_LINEAR = 0x015,
1795     VKD3D_SHADER_FILTER_ANISOTROPIC = 0x055,
1796     VKD3D_SHADER_FILTER_COMPARISON_MIN_MAG_MIP_POINT = 0x080,
1797     VKD3D_SHADER_FILTER_COMPARISON_MIN_MAG_POINT_MIP_LINEAR = 0x081,
1798     VKD3D_SHADER_FILTER_COMPARISON_MIN_POINT_MAG_LINEAR_MIP_POINT = 0x084,
1799     VKD3D_SHADER_FILTER_COMPARISON_MIN_POINT_MAG_MIP_LINEAR = 0x085,
1800     VKD3D_SHADER_FILTER_COMPARISON_MIN_LINEAR_MAG_MIP_POINT = 0x090,
1801     VKD3D_SHADER_FILTER_COMPARISON_MIN_LINEAR_MAG_POINT_MIP_LINEAR = 0x091,
1802     VKD3D_SHADER_FILTER_COMPARISON_MIN_MAG_LINEAR_MIP_POINT = 0x094,
1803     VKD3D_SHADER_FILTER_COMPARISON_MIN_MAG_MIP_LINEAR = 0x095,
1804     VKD3D_SHADER_FILTER_COMPARISON_ANISOTROPIC = 0x0d5,
1805     VKD3D_SHADER_FILTER_MINIMUM_MIN_MAG_MIP_POINT = 0x100,
1806     VKD3D_SHADER_FILTER_MINIMUM_MIN_MAG_POINT_MIP_LINEAR = 0x101,
1807     VKD3D_SHADER_FILTER_MINIMUM_MIN_POINT_MAG_LINEAR_MIP_POINT = 0x104,
1808     VKD3D_SHADER_FILTER_MINIMUM_MIN_POINT_MAG_MIP_LINEAR = 0x105,
1809     VKD3D_SHADER_FILTER_MINIMUM_MIN_LINEAR_MAG_MIP_POINT = 0x110,
1810     VKD3D_SHADER_FILTER_MINIMUM_MIN_LINEAR_MAG_POINT_MIP_LINEAR = 0x111,
1811     VKD3D_SHADER_FILTER_MINIMUM_MIN_MAG_LINEAR_MIP_POINT = 0x114,
1812     VKD3D_SHADER_FILTER_MINIMUM_MIN_MAG_MIP_LINEAR = 0x115,
1813     VKD3D_SHADER_FILTER_MINIMUM_ANISOTROPIC = 0x155,
1814     VKD3D_SHADER_FILTER_MAXIMUM_MIN_MAG_MIP_POINT = 0x180,
1815     VKD3D_SHADER_FILTER_MAXIMUM_MIN_MAG_POINT_MIP_LINEAR = 0x181,
1816     VKD3D_SHADER_FILTER_MAXIMUM_MIN_POINT_MAG_LINEAR_MIP_POINT = 0x184,
1817     VKD3D_SHADER_FILTER_MAXIMUM_MIN_POINT_MAG_MIP_LINEAR = 0x185,

```

```

1818 VKD3D_SHADER_FILTER_MAXIMUM_MIN_LINEAR_MAG_MIP_POINT = 0x190,
1819 VKD3D_SHADER_FILTER_MAXIMUM_MIN_LINEAR_MAG_POINT_LINEAR = 0x191,
1820 VKD3D_SHADER_FILTER_MAXIMUM_MIN_MAG_LINEAR_MIP_POINT = 0x194,
1821 VKD3D_SHADER_FILTER_MAXIMUM_MIN_MAG_MIP_LINEAR = 0x195,
1822 VKD3D_SHADER_FILTER_MAXIMUM_ANISOTROPIC = 0x1d5,
1823
1824 VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_FILTER),
1825 };
1826
1827 enum vkd3d_shader_texture_address_mode
1828 {
1829     VKD3D_SHADER_TEXTURE_ADDRESS_MODE_WRAP = 0x1,
1830     VKD3D_SHADER_TEXTURE_ADDRESS_MODE_MIRROR = 0x2,
1831     VKD3D_SHADER_TEXTURE_ADDRESS_MODE_CLAMP = 0x3,
1832     VKD3D_SHADER_TEXTURE_ADDRESS_MODE_BORDER = 0x4,
1833     VKD3D_SHADER_TEXTURE_ADDRESS_MODE_MIRROR_ONCE = 0x5,
1834
1835     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_TEXTURE_ADDRESS_MODE),
1836 };
1837
1838 enum vkd3d_shader_comparison_func
1839 {
1840     VKD3D_SHADER_COMPARISON_FUNC_NEVER = 0x1,
1841     VKD3D_SHADER_COMPARISON_FUNC_LESS = 0x2,
1842     VKD3D_SHADER_COMPARISON_FUNC_EQUAL = 0x3,
1843     VKD3D_SHADER_COMPARISON_FUNC_LESS_EQUAL = 0x4,
1844     VKD3D_SHADER_COMPARISON_FUNC_GREATER = 0x5,
1845     VKD3D_SHADER_COMPARISON_FUNC_NOT_EQUAL = 0x6,
1846     VKD3D_SHADER_COMPARISON_FUNC_GREATER_EQUAL = 0x7,
1847     VKD3D_SHADER_COMPARISON_FUNC_ALWAYS = 0x8,
1848
1849     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPARISON_FUNC),
1850 };
1851
1852 enum vkd3d_shader_static_border_colour
1853 {
1854     VKD3D_SHADER_STATIC_BORDER_COLOUR_TRANSPARENT_BLACK = 0x0,
1855     VKD3D_SHADER_STATIC_BORDER_COLOUR_OPAQUE_BLACK = 0x1,
1856     VKD3D_SHADER_STATIC_BORDER_COLOUR_OPAQUE_WHITE = 0x2,
1857
1858     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_STATIC_BORDER_COLOUR),
1859 };
1860
1861 struct vkd3d_shader_static_sampler_desc
1862 {
1863     enum vkd3d_shader_filter filter;
1864     enum vkd3d_shader_texture_address_mode address_u;
1865     enum vkd3d_shader_texture_address_mode address_v;
1866     enum vkd3d_shader_texture_address_mode address_w;
1867     float mip_lod_bias;
1868     unsigned int max_anisotropy;
1869     enum vkd3d_shader_comparison_func comparison_func;
1870     enum vkd3d_shader_static_border_colour border_colour;
1871     float min_lod;
1872     float max_lod;
1873     unsigned int shader_register;
1874     unsigned int register_space;
1875     enum vkd3d_shader_visibility shader_visibility;
1876 };
1877
1878 struct vkd3d_shader_descriptor_range
1879 {
1880     enum vkd3d_shader_descriptor_type range_type;
1881     unsigned int descriptor_count;
1882     unsigned int base_shader_register;
1883     unsigned int register_space;
1884     unsigned int descriptor_table_offset;
1885 };
1886
1887 struct vkd3d_shader_root_descriptor_table
1888 {
1889     unsigned int descriptor_range_count;
1890     const struct vkd3d_shader_descriptor_range *descriptor_ranges;
1891 };
1892
1893 struct vkd3d_shader_root_constants
1894 {
1895     unsigned int shader_register;
1896     unsigned int register_space;
1897     unsigned int value_count;
1898 };
1899
1900 struct vkd3d_shader_root_descriptor
1901 {
1902     unsigned int shader_register;
1903     unsigned int register_space;
1904 };

```

```

1905
1906 enum vkd3d_shader_root_parameter_type
1907 {
1908     VKD3D_SHADER_ROOT_PARAMETER_TYPE_DESCRIPTOR_TABLE = 0x0,
1909     VKD3D_SHADER_ROOT_PARAMETER_TYPE_32BIT_CONSTANTS = 0x1,
1910     VKD3D_SHADER_ROOT_PARAMETER_TYPE_CBV = 0x2,
1911     VKD3D_SHADER_ROOT_PARAMETER_TYPE_SRV = 0x3,
1912     VKD3D_SHADER_ROOT_PARAMETER_TYPE_UAV = 0x4,
1913
1914     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_ROOT_PARAMETER_TYPE),
1915 };
1916
1917 struct vkd3d_shader_root_parameter
1918 {
1919     enum vkd3d_shader_root_parameter_type parameter_type;
1920     union
1921     {
1922         struct vkd3d_shader_root_descriptor_table descriptor_table;
1923         struct vkd3d_shader_root_constants constants;
1924         struct vkd3d_shader_root_descriptor descriptor;
1925     } u;
1926     enum vkd3d_shader_visibility shader_visibility;
1927 };
1928
1929 enum vkd3d_shader_root_signature_flags
1930 {
1931     VKD3D_SHADER_ROOT_SIGNATURE_FLAG_NONE = 0x00,
1932     VKD3D_SHADER_ROOT_SIGNATURE_FLAG_ALLOW_INPUT_ASSEMBLER_INPUT_LAYOUT = 0x01,
1933     VKD3D_SHADER_ROOT_SIGNATURE_FLAG_DENY_VERTEX_SHADER_ROOT_ACCESS = 0x02,
1934     VKD3D_SHADER_ROOT_SIGNATURE_FLAG_DENY_HULL_SHADER_ROOT_ACCESS = 0x04,
1935     VKD3D_SHADER_ROOT_SIGNATURE_FLAG_DENY_DOMAIN_SHADER_ROOT_ACCESS = 0x08,
1936     VKD3D_SHADER_ROOT_SIGNATURE_FLAG_DENY_GEOMETRY_SHADER_ROOT_ACCESS = 0x10,
1937     VKD3D_SHADER_ROOT_SIGNATURE_FLAG_DENY_PIXEL_SHADER_ROOT_ACCESS = 0x20,
1938     VKD3D_SHADER_ROOT_SIGNATURE_FLAG_ALLOW_STREAM_OUTPUT = 0x40,
1939
1940     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_ROOT_SIGNATURE_FLAGS),
1941 };
1942
1943 struct vkd3d_shader_root_signature_desc
1944 {
1945     unsigned int parameter_count;
1946     const struct vkd3d_shader_root_parameter *parameters;
1947     unsigned int static_sampler_count;
1948     const struct vkd3d_shader_static_sampler_desc *static_samplers;
1949     enum vkd3d_shader_root_signature_flags flags;
1950 };
1951
1952 /* root signature 1.1 */
1953 enum vkd3d_shader_root_descriptor_flags
1954 {
1955     VKD3D_SHADER_ROOT_DESCRIPTOR_FLAG_NONE = 0x0,
1956     VKD3D_SHADER_ROOT_DESCRIPTOR_FLAG_DATA_VOLATILE = 0x2,
1957     VKD3D_SHADER_ROOT_DESCRIPTOR_FLAG_DATA_STATIC_WHILE_SET_AT_EXECUTE = 0x4,
1958     VKD3D_SHADER_ROOT_DESCRIPTOR_FLAG_DATA_STATIC = 0x8,
1959
1960     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_ROOT_DESCRIPTOR_FLAGS),
1961 };
1962
1963 enum vkd3d_shader_descriptor_range_flags
1964 {
1965     VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_NONE = 0x0,
1966     VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_DESCRIPTOR_VOLATILE = 0x1,
1967     VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_DATA_VOLATILE = 0x2,
1968     VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_DATA_STATIC_WHILE_SET_AT_EXECUTE = 0x4,
1969     VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_DATA_STATIC = 0x8,
1970     VKD3D_SHADER_DESCRIPTOR_RANGE_FLAG_DESCRIPTOR_STATIC_KEEPING_BUFFER_BOUNDS_CHECKS = 0x10000,
1971
1972     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_DESCRIPTOR_RANGE_FLAGS),
1973 };
1974
1975
1976 struct vkd3d_shader_descriptor_range1
1977 {
1978     enum vkd3d_shader_descriptor_type range_type;
1979     unsigned int descriptor_count;
1980     unsigned int base_shader_register;
1981     unsigned int register_space;
1982     enum vkd3d_shader_descriptor_range_flags flags;
1983     unsigned int descriptor_table_offset;
1984 };
1985
1986 struct vkd3d_shader_root_descriptor_table1
1987 {
1988     unsigned int descriptor_range_count;
1989     const struct vkd3d_shader_descriptor_range1 *descriptor_ranges;
1990 };
1991
1992 struct vkd3d_shader_root_descriptor1

```

```

1993 {
1994     unsigned int shader_register;
1995     unsigned int register_space;
1996     enum vkd3d_shader_root_descriptor_flags flags;
1997 };
1998
1999 struct vkd3d_shader_root_parameter1
2000 {
2001     enum vkd3d_shader_root_parameter_type parameter_type;
2002     union
2003     {
2004         struct vkd3d_shader_root_descriptor_table1 descriptor_table;
2005         struct vkd3d_shader_root_constants constants;
2006         struct vkd3d_shader_root_descriptor1 descriptor;
2007     } u;
2008     enum vkd3d_shader_visibility shader_visibility;
2009 };
2010
2011 struct vkd3d_shader_root_signature_desc1
2012 {
2013     unsigned int parameter_count;
2014     const struct vkd3d_shader_root_parameter1 *parameters;
2015     unsigned int static_sampler_count;
2016     const struct vkd3d_shader_static_sampler_desc *static_samplers;
2017     enum vkd3d_shader_root_signature_flags flags;
2018 };
2019
2020 enum vkd3d_shader_root_signature_version
2021 {
2022     VKD3D_SHADER_ROOT_SIGNATURE_VERSION_1_0 = 0x1,
2023     VKD3D_SHADER_ROOT_SIGNATURE_VERSION_1_1 = 0x2,
2024
2025     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_ROOT_SIGNATURE_VERSION),
2026 };
2027
2028 struct vkd3d_shader_versioned_root_signature_desc
2029 {
2030     enum vkd3d_shader_root_signature_version version;
2031     union
2032     {
2033         struct vkd3d_shader_root_signature_desc v_1_0;
2034         struct vkd3d_shader_root_signature_desc1 v_1_1;
2035     } u;
2036 };
2037
2042 enum vkd3d_shader_resource_type
2043 {
2044     VKD3D_SHADER_RESOURCE_NONE = 0x0,
2045     VKD3D_SHADER_RESOURCE_BUFFER = 0x1,
2046     VKD3D_SHADER_RESOURCE_TEXTURE_1D = 0x2,
2047     VKD3D_SHADER_RESOURCE_TEXTURE_2D = 0x3,
2048     VKD3D_SHADER_RESOURCE_TEXTURE_2DMS = 0x4,
2049     VKD3D_SHADER_RESOURCE_TEXTURE_3D = 0x5,
2050     VKD3D_SHADER_RESOURCE_TEXTURE_CUBE = 0x6,
2051     VKD3D_SHADER_RESOURCE_TEXTURE_1DARRAY = 0x7,
2052     VKD3D_SHADER_RESOURCE_TEXTURE_2DARRAY = 0x8,
2053     VKD3D_SHADER_RESOURCE_TEXTURE_2DMSARRAY = 0x9,
2054     VKD3D_SHADER_RESOURCE_TEXTURE_CUBEARRAY = 0xa,
2055
2056     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_RESOURCE_TYPE),
2057 };
2072
2077 enum vkd3d_shader_resource_data_type
2078 {
2079     VKD3D_SHADER_RESOURCE_DATA_UNORM = 0x1,
2080     VKD3D_SHADER_RESOURCE_DATA_SNORM = 0x2,
2081     VKD3D_SHADER_RESOURCE_DATA_INT = 0x3,
2082     VKD3D_SHADER_RESOURCE_DATA_UINT = 0x4,
2083     VKD3D_SHADER_RESOURCE_DATA_FLOAT = 0x5,
2084     VKD3D_SHADER_RESOURCE_DATA_MIXED = 0x6,
2085     VKD3D_SHADER_RESOURCE_DATA_DOUBLE = 0x7,
2086     VKD3D_SHADER_RESOURCE_DATA_CONTINUED = 0x8,
2087
2088     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_RESOURCE_DATA_TYPE),
2089 };
2102
2107 enum vkd3d_shader_descriptor_info_flag
2108 {
2109     VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_UAV_COUNTER = 0x00000001,
2110     VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_UAV_READ = 0x00000002,
2111     VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_SAMPLER_COMPARISON_MODE = 0x00000004,
2112     VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_UAV_ATOMICS = 0x00000008,
2113     VKD3D_SHADER_DESCRIPTOR_INFO_FLAG_RAW_BUFFER = 0x00000010,
2114
2115     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_DESCRIPTOR_INFO_FLAG),
2116 };
2126

```

```

2131 struct vkd3d_shader_descriptor_info
2132 {
2133     enum vkd3d_shader_descriptor_type type;
2139     unsigned int register_space;
2141     unsigned int register_index;
2143     enum vkd3d_shader_resource_type resource_type;
2145     enum vkd3d_shader_resource_data_type resource_data_type;
2150     unsigned int flags;
2155     unsigned int count;
2156 };
2157
2191 struct vkd3d_shader_scan_descriptor_info
2192 {
2196     enum vkd3d_shader_structure_type type;
2198     const void *next;
2199
2201     struct vkd3d_shader_descriptor_info *descriptors;
2203     unsigned int descriptor_count;
2204 };
2205
2212 struct vkd3d_shader_combined_resource_sampler_info
2213 {
2214     unsigned int resource_space;
2215     unsigned int resource_index;
2216     unsigned int sampler_space;
2217     unsigned int sampler_index;
2218 };
2219
2240 struct vkd3d_shader_scan_combined_resource_sampler_info
2241 {
2243     enum vkd3d_shader_structure_type type;
2245     const void *next;
2246
2248     struct vkd3d_shader_combined_resource_sampler_info *combined_samplers;
2250     unsigned int combined_sampler_count;
2251 };
2252
2260 struct vkd3d_shader_scan_hull_shader_tessellation_info
2261 {
2263     enum vkd3d_shader_structure_type type;
2265     const void *next;
2266
2268     enum vkd3d_shader_tessellator_output_primitive output_primitive;
2270     enum vkd3d_shader_tessellator_partitioning partitioning;
2271 };
2272
2277 enum vkd3d_shader_component_type
2278 {
2280     VKD3D_SHADER_COMPONENT_VOID = 0x0,
2282     VKD3D_SHADER_COMPONENT_UINT = 0x1,
2284     VKD3D_SHADER_COMPONENT_INT = 0x2,
2286     VKD3D_SHADER_COMPONENT_FLOAT = 0x3,
2288     VKD3D_SHADER_COMPONENT_BOOL = 0x4,
2290     VKD3D_SHADER_COMPONENT_DOUBLE = 0x5,
2292     VKD3D_SHADER_COMPONENT_UINT64 = 0x6,
2294     VKD3D_SHADER_COMPONENT_INT64 = 0x7,
2296     VKD3D_SHADER_COMPONENT_FLOAT16 = 0x8,
2298     VKD3D_SHADER_COMPONENT_UINT16 = 0x9,
2300     VKD3D_SHADER_COMPONENT_INT16 = 0xa,
2301
2302     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_COMPONENT_TYPE),
2303 };
2304
2306 enum vkd3d_shader_sysval_semantic
2307 {
2309     VKD3D_SHADER_SV_NONE = 0x00,
2311     VKD3D_SHADER_SV_POSITION = 0x01,
2313     VKD3D_SHADER_SV_CLIP_DISTANCE = 0x02,
2315     VKD3D_SHADER_SV_CULL_DISTANCE = 0x03,
2317     VKD3D_SHADER_SV_RENDER_TARGET_ARRAY_INDEX = 0x04,
2319     VKD3D_SHADER_SV_VIEWPORT_ARRAY_INDEX = 0x05,
2321     VKD3D_SHADER_SV_VERTEX_ID = 0x06,
2323     VKD3D_SHADER_SV_PRIMITIVE_ID = 0x07,
2325     VKD3D_SHADER_SV_INSTANCE_ID = 0x08,
2327     VKD3D_SHADER_SV_IS_FRONT_FACE = 0x09,
2329     VKD3D_SHADER_SV_SAMPLE_INDEX = 0x0a,
2330     VKD3D_SHADER_SV_TESS_FACTOR_QUADEDGE = 0x0b,
2331     VKD3D_SHADER_SV_TESS_FACTOR_QUADINT = 0x0c,
2332     VKD3D_SHADER_SV_TESS_FACTOR_TRIEDGE = 0x0d,
2333     VKD3D_SHADER_SV_TESS_FACTOR_TRIINT = 0x0e,
2334     VKD3D_SHADER_SV_TESS_FACTOR_LINEDET = 0x0f,
2335     VKD3D_SHADER_SV_TESS_FACTOR_LINEDEN = 0x10,
2337     VKD3D_SHADER_SV_TARGET = 0x40,
2339     VKD3D_SHADER_SV_DEPTH = 0x41,
2341     VKD3D_SHADER_SV_COVERAGE = 0x42,
2346     VKD3D_SHADER_SV_DEPTH_GREATER_EQUAL = 0x43,
2351     VKD3D_SHADER_SV_DEPTH_LESS_EQUAL = 0x44,

```

```

2353     VKD3D_SHADER_SV_STENCIL_REF                = 0x45,
2354
2355     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_SYSVAL_SEMANTIC),
2356 };
2357
2362 enum vkd3d_shader_minimum_precision
2363 {
2364     VKD3D_SHADER_MINIMUM_PRECISION_NONE        = 0,
2366     VKD3D_SHADER_MINIMUM_PRECISION_FLOAT_16    = 1,
2368     VKD3D_SHADER_MINIMUM_PRECISION_FIXED_8_2   = 2,
2370     VKD3D_SHADER_MINIMUM_PRECISION_INT_16      = 4,
2372     VKD3D_SHADER_MINIMUM_PRECISION_UINT_16     = 5,
2373
2374     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_MINIMUM_PRECISION),
2375 };
2376
2380 struct vkd3d_shader_signature_element
2381 {
2383     const char *semantic_name;
2385     unsigned int semantic_index;
2390     unsigned int stream_index;
2395     enum vkd3d_shader_sysval_semantic sysval_semantic;
2397     enum vkd3d_shader_component_type component_type;
2399     unsigned int register_index;
2401     unsigned int mask;
2408     unsigned int used_mask;
2410     enum vkd3d_shader_minimum_precision min_precision;
2411 };
2412
2420 struct vkd3d_shader_signature
2421 {
2423     struct vkd3d_shader_signature_element *elements;
2425     unsigned int element_count;
2426 };
2427
2429 enum vkd3d_shader_swizzle_component
2430 {
2431     VKD3D_SHADER_SWIZZLE_X = 0x0,
2432     VKD3D_SHADER_SWIZZLE_Y = 0x1,
2433     VKD3D_SHADER_SWIZZLE_Z = 0x2,
2434     VKD3D_SHADER_SWIZZLE_W = 0x3,
2435
2436     VKD3D_FORCE_32_BIT_ENUM(VKD3D_SHADER_SWIZZLE_COMPONENT),
2437 };
2438
2444 struct vkd3d_shader_dxbc_section_desc
2445 {
2447     uint32_t tag;
2449     struct vkd3d_shader_code data;
2450 };
2451
2457 struct vkd3d_shader_dxbc_desc
2458 {
2463     uint32_t tag;
2465     uint32_t checksum[4];
2470     unsigned int version;
2472     size_t size;
2474     size_t section_count;
2476     struct vkd3d_shader_dxbc_section_desc *sections;
2477 };
2478
2483 #define VKD3D_SHADER_SWIZZLE_MASK (0xffu)
2485 #define VKD3D_SHADER_SWIZZLE_SHIFT(idx) (8u * (idx))
2486
2496 #define VKD3D_SHADER_SWIZZLE(x, y, z, w) \
2497 (VKD3D_SHADER_SWIZZLE_ ## x « VKD3D_SHADER_SWIZZLE_SHIFT(0) \
2498 | VKD3D_SHADER_SWIZZLE_ ## y « VKD3D_SHADER_SWIZZLE_SHIFT(1) \
2499 | VKD3D_SHADER_SWIZZLE_ ## z « VKD3D_SHADER_SWIZZLE_SHIFT(2) \
2500 | VKD3D_SHADER_SWIZZLE_ ## w « VKD3D_SHADER_SWIZZLE_SHIFT(3))
2501
2503 #define VKD3D_SHADER_NO_SWIZZLE VKD3D_SHADER_SWIZZLE(X, Y, Z, W)
2504
2506 static inline uint32_t vkd3d_shader_create_swizzle(enum vkd3d_shader_swizzle_component x,
2507     enum vkd3d_shader_swizzle_component y, enum vkd3d_shader_swizzle_component z,
2508     enum vkd3d_shader_swizzle_component w)
2509 {
2510     return ((x & VKD3D_SHADER_SWIZZLE_MASK) « VKD3D_SHADER_SWIZZLE_SHIFT(0))
2511         | ((y & VKD3D_SHADER_SWIZZLE_MASK) « VKD3D_SHADER_SWIZZLE_SHIFT(1))
2512         | ((z & VKD3D_SHADER_SWIZZLE_MASK) « VKD3D_SHADER_SWIZZLE_SHIFT(2))
2513         | ((w & VKD3D_SHADER_SWIZZLE_MASK) « VKD3D_SHADER_SWIZZLE_SHIFT(3));
2514 }
2515
2557 struct vkd3d_shader_scan_signature_info
2558 {
2560     enum vkd3d_shader_structure_type type;
2562     const void *next;
2563 }

```

```

2565     struct vkd3d_shader_signature input;
2566
2568     struct vkd3d_shader_signature output;
2569
2571     struct vkd3d_shader_signature patch_constant;
2572 };
2573
2580 struct vkd3d_shader_varying_map
2581 {
2588     unsigned int output_signature_index;
2590     unsigned int input_register_index;
2592     unsigned int input_mask;
2593 };
2594
2619 struct vkd3d_shader_varying_map_info
2620 {
2622     enum vkd3d_shader_structure_type type;
2624     const void *next;
2625
2637     const struct vkd3d_shader_varying_map *varying_map;
2639     unsigned int varying_count;
2640 };
2641
2667 struct vkd3d_shader_parameter_info
2668 {
2670     enum vkd3d_shader_structure_type type;
2672     const void *next;
2673
2675     const struct vkd3d_shader_parameter1 *parameters;
2677     unsigned int parameter_count;
2678 };
2679
2680 #ifdef LIBVKD3D_SHADER_SOURCE
2681 # define VKD3D_SHADER_API VKD3D_EXPORT
2682 #else
2683 # define VKD3D_SHADER_API VKD3D_IMPORT
2684 #endif
2685
2686 #ifndef VKD3D_SHADER_NO_PROTOTYPES
2687
2699 VKD3D_SHADER_API const char *vkd3d_shader_get_version(unsigned int *major, unsigned int *minor);
2715 VKD3D_SHADER_API const enum vkd3d_shader_source_type *vkd3d_shader_get_supported_source_types(unsigned
    int *count);
2730 VKD3D_SHADER_API const enum vkd3d_shader_target_type *vkd3d_shader_get_supported_target_types(
2731     enum vkd3d_shader_source_type source_type, unsigned int *count);
2732
2800 VKD3D_SHADER_API int vkd3d_shader_compile(const struct vkd3d_shader_compile_info *compile_info,
2801     struct vkd3d_shader_code *out, char **messages);
2809 VKD3D_SHADER_API void vkd3d_shader_free_messages(char *messages);
2819 VKD3D_SHADER_API void vkd3d_shader_free_shader_code(struct vkd3d_shader_code *code);
2820
2856 VKD3D_SHADER_API int vkd3d_shader_parse_root_signature(const struct vkd3d_shader_code *dxbc,
2857     struct vkd3d_shader_versioned_root_signature_desc *root_signature, char **messages);
2868 VKD3D_SHADER_API void vkd3d_shader_free_root_signature(
2869     struct vkd3d_shader_versioned_root_signature_desc *root_signature);
2870
2899 VKD3D_SHADER_API int vkd3d_shader_serialize_root_signature(
2900     const struct vkd3d_shader_versioned_root_signature_desc *root_signature,
2901     struct vkd3d_shader_code *dxbc, char **messages);
2922 VKD3D_SHADER_API int vkd3d_shader_convert_root_signature(struct
    vkd3d_shader_versioned_root_signature_desc *dst,
2923     enum vkd3d_shader_root_signature_version version, const struct
    vkd3d_shader_versioned_root_signature_desc *src);
2924
2982 VKD3D_SHADER_API int vkd3d_shader_scan(const struct vkd3d_shader_compile_info *compile_info, char
    **messages);
2992 VKD3D_SHADER_API void vkd3d_shader_free_scan_descriptor_info(
2993     struct vkd3d_shader_scan_descriptor_info *scan_descriptor_info);
2994
3034 VKD3D_SHADER_API int vkd3d_shader_parse_input_signature(const struct vkd3d_shader_code *dxbc,
3035     struct vkd3d_shader_signature *signature, char **messages);
3055 VKD3D_SHADER_API struct vkd3d_shader_signature_element *vkd3d_shader_find_signature_element(
3056     const struct vkd3d_shader_signature *signature, const char *semantic_name,
3057     unsigned int semantic_index, unsigned int stream_index);
3067 VKD3D_SHADER_API void vkd3d_shader_free_shader_signature(struct vkd3d_shader_signature *signature);
3068
3069 /* 1.3 */
3070
3104 VKD3D_SHADER_API int vkd3d_shader_preprocess(const struct vkd3d_shader_compile_info *compile_info,
3105     struct vkd3d_shader_code *out, char **messages);
3106
3117 VKD3D_SHADER_API void vkd3d_shader_set_log_callback(PFN_vkd3d_log callback);
3118
3130 VKD3D_SHADER_API void vkd3d_shader_free_dxbc(struct vkd3d_shader_dxbc_desc *dxbc);
3131
3161 VKD3D_SHADER_API int vkd3d_shader_parse_dxbc(const struct vkd3d_shader_code *dxbc,
3162     uint32_t flags, struct vkd3d_shader_dxbc_desc *desc, char **messages);

```

```

3163
3194 VKD3D_SHADER_API int vkd3d_shader_serialize_dxbc(size_t section_count,
3195             const struct vkd3d_shader_dxbc_section_desc *sections, struct vkd3d_shader_code *dxbc, char
             **messages);
3196
3208 VKD3D_SHADER_API void vkd3d_shader_free_scan_signature_info(struct vkd3d_shader_scan_signature_info
             *info);
3209
3235 VKD3D_SHADER_API void vkd3d_shader_build_varying_map(const struct vkd3d_shader_signature
             *output_signature,
3236             const struct vkd3d_shader_signature *input_signature,
3237             unsigned int *count, struct vkd3d_shader_varying_map *varyings);
3238
3251 VKD3D_SHADER_API void vkd3d_shader_free_scan_combined_resource_sampler_info(
3252             struct vkd3d_shader_scan_combined_resource_sampler_info *info);
3253
3254 #endif /* VKD3D_SHADER_NO_PROTOTYPES */
3255
3257 typedef const char *(*PFN_vkd3d_shader_get_version)(unsigned int *major, unsigned int *minor);
3259 typedef const enum vkd3d_shader_source_type *(*PFN_vkd3d_shader_get_supported_source_types)(unsigned
             int *count);
3261 typedef const enum vkd3d_shader_target_type *(*PFN_vkd3d_shader_get_supported_target_types)(
3262             enum vkd3d_shader_source_type source_type, unsigned int *count);
3263
3265 typedef int (*PFN_vkd3d_shader_compile)(const struct vkd3d_shader_compile_info *compile_info,
3266             struct vkd3d_shader_code *out, char **messages);
3268 typedef void (*PFN_vkd3d_shader_free_messages)(char *messages);
3270 typedef void (*PFN_vkd3d_shader_free_shader_code)(struct vkd3d_shader_code *code);
3271
3273 typedef int (*PFN_vkd3d_shader_parse_root_signature)(const struct vkd3d_shader_code *dxbc,
3274             struct vkd3d_shader_versioned_root_signature_desc *root_signature, char **messages);
3276 typedef void (*PFN_vkd3d_shader_free_root_signature)(struct vkd3d_shader_versioned_root_signature_desc
             *root_signature);
3277
3279 typedef int (*PFN_vkd3d_shader_serialize_root_signature)(
3280             const struct vkd3d_shader_versioned_root_signature_desc *root_signature,
3281             struct vkd3d_shader_code *dxbc, char **messages);
3282
3284 typedef int (*PFN_vkd3d_shader_convert_root_signature)(struct
             vkd3d_shader_versioned_root_signature_desc *dst,
3285             enum vkd3d_shader_root_signature_version version, const struct
             vkd3d_shader_versioned_root_signature_desc *src);
3286
3288 typedef int (*PFN_vkd3d_shader_scan)(const struct vkd3d_shader_compile_info *compile_info, char
             **messages);
3290 typedef void (*PFN_vkd3d_shader_free_scan_descriptor_info)(
3291             struct vkd3d_shader_scan_descriptor_info *scan_descriptor_info);
3292
3294 typedef int (*PFN_vkd3d_shader_parse_input_signature)(const struct vkd3d_shader_code *dxbc,
3295             struct vkd3d_shader_signature *signature, char **messages);
3297 typedef struct vkd3d_shader_signature_element *(*PFN_vkd3d_shader_find_signature_element)(
3298             const struct vkd3d_shader_signature *signature, const char *semantic_name,
3299             unsigned int semantic_index, unsigned int stream_index);
3301 typedef void (*PFN_vkd3d_shader_free_shader_signature)(struct vkd3d_shader_signature *signature);
3302
3304 typedef void (*PFN_vkd3d_shader_preprocess)(struct vkd3d_shader_compile_info *compile_info,
3305             struct vkd3d_shader_code *out, char **messages);
3306
3308 typedef void (*PFN_vkd3d_shader_set_log_callback)(PFN_vkd3d_log callback);
3309
3311 typedef void (*PFN_vkd3d_shader_free_dxbc)(struct vkd3d_shader_dxbc_desc *dxbc);
3313 typedef int (*PFN_vkd3d_shader_parse_dxbc)(const struct vkd3d_shader_code *dxbc,
3314             uint32_t flags, struct vkd3d_shader_dxbc_desc *desc, char **messages);
3316 typedef int (*PFN_vkd3d_shader_serialize_dxbc)(size_t section_count,
3317             const struct vkd3d_shader_dxbc_section_desc *sections, struct vkd3d_shader_code *dxbc, char
             **messages);
3318
3320 typedef void (*PFN_vkd3d_shader_build_varying_map)(const struct vkd3d_shader_signature
             *output_signature,
3321             const struct vkd3d_shader_signature *input_signature,
3322             unsigned int *count, struct vkd3d_shader_varying_map *varyings);
3324 typedef void (*PFN_vkd3d_shader_free_scan_signature_info)(struct vkd3d_shader_scan_signature_info
             *info);
3325
3327 typedef void (*PFN_vkd3d_shader_free_scan_combined_resource_sampler_info)(
3328             struct vkd3d_shader_scan_combined_resource_sampler_info *info);
3329
3330 #ifdef __cplusplus
3331 }
3332 #endif /* __cplusplus */
3333
3334 #endif /* __VKD3D_SHADER_H */

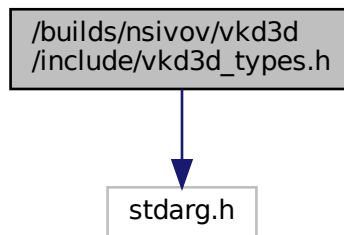
```

## 4.5 /builds/nsivov/vkd3d/include/vkd3d\_types.h File Reference

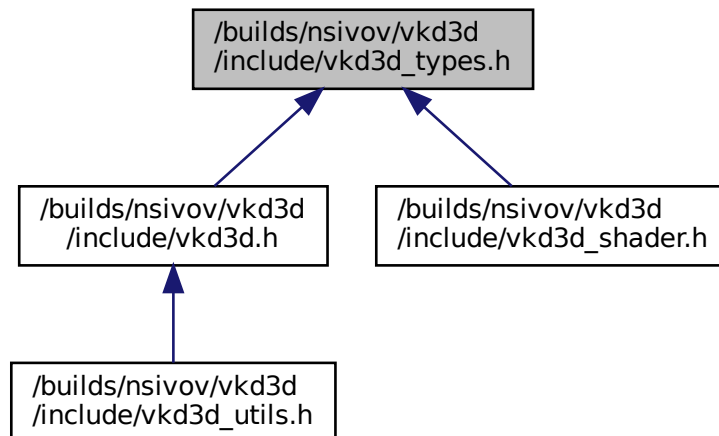
This file contains definitions for basic types used by vkd3d libraries.

```
#include <stdarg.h>
```

Include dependency graph for vkd3d\_types.h:



This graph shows which files directly or indirectly include this file:



### Typedefs

- typedef void(\* **PFN\_vkd3d\_log**) (const char \*format, va\_list args)

## Enumerations

```
enum vkd3d_result {
    VKD3D_OK = 0, VKD3D_FALSE = 1, VKD3D_ERROR = -1, VKD3D_ERROR_OUT_OF_MEMORY = -2,
    VKD3D_ERROR_INVALID_ARGUMENT = -3, VKD3D_ERROR_INVALID_SHADER = -4, VKD3D_ERROR_NOT_IMPLEMENTED = -5,
    VKD3D_ERROR_KEY_ALREADY_EXISTS = -6,
    VKD3D_ERROR_NOT_FOUND = -7, VKD3D_ERROR_MORE_DATA = -8, VKD3D_FORCE_32_BIT_ENUM = (VKD3D_RESULT) }

```

*Result codes returned by some vkd3d functions.*

### 4.5.1 Detailed Description

This file contains definitions for basic types used by vkd3d libraries.

### 4.5.2 Enumeration Type Documentation

#### 4.5.2.1 vkd3d\_result

```
enum vkd3d_result
```

Result codes returned by some vkd3d functions.

Error codes always have negative values; non-error codes never do.

#### Enumerator

VKD3D_OK	Success.
VKD3D_FALSE	Success as a result of there being nothing to do.  <b>Since</b> 1.12
VKD3D_ERROR	An unspecified failure occurred.
VKD3D_ERROR_OUT_OF_MEMORY	There are not enough resources available to complete the operation.
VKD3D_ERROR_INVALID_ARGUMENT	One or more parameters passed to a vkd3d function were invalid.
VKD3D_ERROR_INVALID_SHADER	A shader passed to a vkd3d function was invalid.
VKD3D_ERROR_NOT_IMPLEMENTED	The operation is not implemented in this version of vkd3d.
VKD3D_ERROR_KEY_ALREADY_EXISTS	The object or entry already exists.  <b>Since</b> 1.12
VKD3D_ERROR_NOT_FOUND	The requested object was not found.  <b>Since</b> 1.12
VKD3D_ERROR_MORE_DATA	The output buffer is larger than the requested object.
	<b>Since</b> 1.12.

## 4.6 vkd3d\_types.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright 2016-2018 Józef Kucia for CodeWeavers
3  *
4  * This library is free software; you can redistribute it and/or
5  * modify it under the terms of the GNU Lesser General Public
6  * License as published by the Free Software Foundation; either
7  * version 2.1 of the License, or (at your option) any later version.
8  *
9  * This library is distributed in the hope that it will be useful,
10 * but WITHOUT ANY WARRANTY; without even the implied warranty of
11 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
12 * Lesser General Public License for more details.
13 *
14 * You should have received a copy of the GNU Lesser General Public
15 * License along with this library; if not, write to the Free Software
16 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
17 */
18
19 #ifndef __VKD3D_TYPES_H
20 #define __VKD3D_TYPES_H
21
22 #include <stdarg.h>
23
24 #ifdef __cplusplus
25 extern "C" {
26 #endif /* __cplusplus */
27
28 #define VKD3D_FORCE_32_BIT_ENUM(name) name##_FORCE_32BIT = 0x7fffffff
29
30 enum vkd3d_result
31 {
32     VKD3D_OK = 0,
33     VKD3D_FALSE = 1,
34     VKD3D_ERROR = -1,
35     VKD3D_ERROR_OUT_OF_MEMORY = -2,
36     VKD3D_ERROR_INVALID_ARGUMENT = -3,
37     VKD3D_ERROR_INVALID_SHADER = -4,
38     VKD3D_ERROR_NOT_IMPLEMENTED = -5,
39     VKD3D_ERROR_KEY_ALREADY_EXISTS = -6,
40     VKD3D_ERROR_NOT_FOUND = -7,
41     VKD3D_ERROR_MORE_DATA = -8,
42
43     VKD3D_FORCE_32_BIT_ENUM(VKD3D_RESULT),
44 };
45
46 typedef void (*PFN_vkd3d_log)(const char *format, va_list args);
47
48 #ifdef _WIN32
49 # define VKD3D_IMPORT __declspec(dllimport)
50 # define VKD3D_EXPORT __declspec(dllexport)
51 #elif defined(__GNUC__)
52 # define VKD3D_IMPORT
53 # define VKD3D_EXPORT __attribute__((visibility("default")))
54 #else
55 # define VKD3D_IMPORT
56 # define VKD3D_EXPORT
57 #endif
58
59 #ifdef __cplusplus
60 }
61 #endif /* __cplusplus */
62
63 #endif /* __VKD3D_TYPES_H */

```

## 4.7 /builds/nsivov/vkd3d/include/vkd3d\_utils.h File Reference

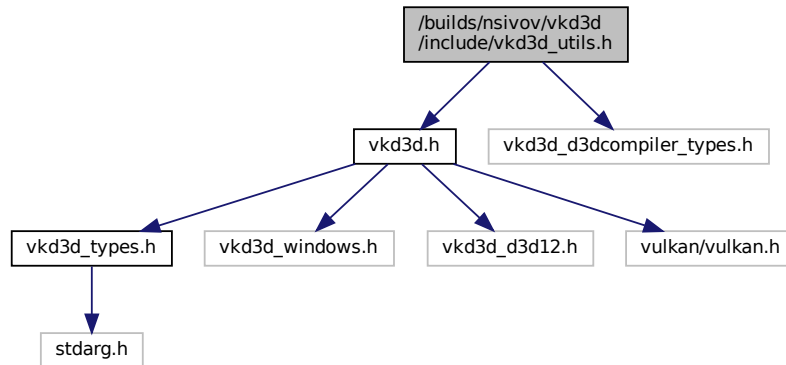
This file contains definitions for the vkd3d-utils library.

```

#include <vkd3d.h>
#include <vkd3d_d3dcompiler_types.h>

```

Include dependency graph for vkd3d\_utils.h:



## Macros

- `#define VKD3D_UTILS_API_VERSION VKD3D_API_VERSION_1_0`
- `#define VKD3D_WAIT_OBJECT_0 (0)`
- `#define VKD3D_WAIT_TIMEOUT (1)`
- `#define VKD3D_WAIT_FAILED (~0u)`
- `#define VKD3D_INFINITE (~0u)`
- `#define VKD3D_UTILS_API VKD3D_IMPORT`
- `#define D3D12CreateDevice(a, b, c, d) D3D12CreateDeviceVKD3D(a, b, c, d, VKD3D_UTILS_API_VERSION)`

## Functions

- VKD3D\_UTILS\_API HANDLE **vkd3d\_create\_event** (void)
- VKD3D\_UTILS\_API HRESULT **vkd3d\_signal\_event** (HANDLE event)
- VKD3D\_UTILS\_API unsigned int **vkd3d\_wait\_event** (HANDLE event, unsigned int milliseconds)
- VKD3D\_UTILS\_API void **vkd3d\_destroy\_event** (HANDLE event)
- VKD3D\_UTILS\_API HRESULT WINAPI **D3D12CreateRootSignatureDeserializer** (const void \*data, SIZE\_T data\_size, REFIID iid, void \*\*deserializer)
- VKD3D\_UTILS\_API HRESULT WINAPI **D3D12GetDebugInterface** (REFIID iid, void \*\*debug)
- VKD3D\_UTILS\_API HRESULT WINAPI **D3D12SerializeRootSignature** (const D3D12\_ROOT\_SIGNATURE\_DESC \*desc, D3D\_ROOT\_SIGNATURE\_VERSION version, ID3DBlob \*\*blob, ID3DBlob \*\*error\_blob)
- VKD3D\_UTILS\_API HRESULT WINAPI **D3D12CreateDeviceVKD3D** (IUnknown \*adapter, D3D\_FEATURE\_LEVEL feature\_level, REFIID iid, void \*\*device, enum vkd3d\_api\_version api\_version)
- VKD3D\_UTILS\_API HRESULT WINAPI **D3D12CreateVersionedRootSignatureDeserializer** (const void \*data, SIZE\_T data\_size, REFIID iid, void \*\*deserializer)
- VKD3D\_UTILS\_API HRESULT WINAPI **D3D12SerializeVersionedRootSignature** (const D3D12\_VERSIONED\_ROOT\_SIGNATURE\_DESC \*desc, ID3DBlob \*\*blob, ID3DBlob \*\*error\_blob)
- VKD3D\_UTILS\_API HRESULT WINAPI **D3DCompile** (const void \*data, SIZE\_T data\_size, const char \*filename, const D3D\_SHADER\_MACRO \*defines, ID3DInclude \*include, const char \*entrypoint, const char \*target, UINT flags, UINT effect\_flags, ID3DBlob \*\*shader, ID3DBlob \*\*error\_messages)

- VKD3D\_UTILS\_API HRESULT WINAPI [D3DCompile2](#) (const void \*data, SIZE\_T data\_size, const char \*filename, const D3D\_SHADER\_MACRO \*defines, ID3DInclude \*include, const char \*entrypoint, const char \*target, UINT flags, UINT effect\_flags, UINT secondary\_flags, const void \*secondary\_data, SIZE\_T secondary\_data\_size, ID3DBlob \*\*shader, ID3DBlob \*\*error\_messages)  
*[D3DCompile2\(\)](#) targets the behaviour of d3dcompiler\_47.dll.*
- VKD3D\_UTILS\_API HRESULT WINAPI **D3DCreateBlob** (SIZE\_T data\_size, ID3DBlob \*\*blob)
- VKD3D\_UTILS\_API HRESULT WINAPI **D3DPreprocess** (const void \*data, SIZE\_T size, const char \*filename, const D3D\_SHADER\_MACRO \*defines, ID3DInclude \*include, ID3DBlob \*\*shader, ID3DBlob \*\*error\_messages)
- VKD3D\_UTILS\_API void [vkd3d\\_utils\\_set\\_log\\_callback](#) (PFN\_vkd3d\_log callback)  
*Set a callback to be called when vkd3d-utils outputs debug logging.*
- VKD3D\_UTILS\_API HRESULT WINAPI [D3DGetBlobPart](#) (const void \*data, SIZE\_T data\_size, D3D\_BLOB\_PART part, UINT flags, ID3DBlob \*\*blob)
- VKD3D\_UTILS\_API HRESULT WINAPI [D3DGetDebugInfo](#) (const void \*data, SIZE\_T data\_size, ID3DBlob \*\*blob)
- VKD3D\_UTILS\_API HRESULT WINAPI [D3DGetInputAndOutputSignatureBlob](#) (const void \*data, SIZE\_T data\_size, ID3DBlob \*\*blob)
- VKD3D\_UTILS\_API HRESULT WINAPI [D3DGetInputSignatureBlob](#) (const void \*data, SIZE\_T data\_size, ID3DBlob \*\*blob)
- VKD3D\_UTILS\_API HRESULT WINAPI [D3DGetOutputSignatureBlob](#) (const void \*data, SIZE\_T data\_size, ID3DBlob \*\*blob)
- VKD3D\_UTILS\_API HRESULT WINAPI [D3DStripShader](#) (const void \*data, SIZE\_T data\_size, UINT flags, ID3DBlob \*\*blob)
- VKD3D\_UTILS\_API HRESULT WINAPI [D3DDisassemble](#) (const void \*data, SIZE\_T data\_size, UINT flags, const char \*comments, ID3DBlob \*\*blob)
- VKD3D\_UTILS\_API HRESULT WINAPI [D3DReflect](#) (const void \*data, SIZE\_T data\_size, REFIID iid, void \*\*reflection)
- VKD3D\_UTILS\_API HRESULT WINAPI [D3DCompile2VKD3D](#) (const void \*data, SIZE\_T data\_size, const char \*filename, const D3D\_SHADER\_MACRO \*defines, ID3DInclude \*include, const char \*entrypoint, const char \*target, UINT flags, UINT effect\_flags, UINT secondary\_flags, const void \*secondary\_data, SIZE\_T secondary\_data\_size, ID3DBlob \*\*shader, ID3DBlob \*\*error\_messages, unsigned int compiler\_version)  
*As [D3DCompile2\(\)](#), but with an extra argument that allows targeting different d3dcompiler versions.*

### 4.7.1 Detailed Description

This file contains definitions for the vkd3d-utils library.

The vkd3d-utils library is a collections of routines to ease the porting of a Direct3D 12 application to vkd3d.

Since

1.0

### 4.7.2 Function Documentation

#### 4.7.2.1 D3DCompile2()

```
VKD3D_UTILS_API HRESULT WINAPI D3DCompile2 (
    const void * data,
    SIZE_T data_size,
    const char * filename,
    const D3D_SHADER_MACRO * defines,
    ID3DInclude * include,
    const char * entrypoint,
    const char * target,
    UINT flags,
    UINT effect_flags,
    UINT secondary_flags,
    const void * secondary_data,
    SIZE_T secondary_data_size,
    ID3DBlob ** shader,
    ID3DBlob ** error_messages )
```

[D3DCompile2\(\)](#) targets the behaviour of d3dcompiler\_47.dll.

To target the behaviour of other d3dcompiler versions, use [D3DCompile2VKD3D\(\)](#).

Since

1.3

#### 4.7.2.2 D3DCompile2VKD3D()

```
VKD3D_UTILS_API HRESULT WINAPI D3DCompile2VKD3D (
    const void * data,
    SIZE_T data_size,
    const char * filename,
    const D3D_SHADER_MACRO * defines,
    ID3DInclude * include,
    const char * entrypoint,
    const char * target,
    UINT flags,
    UINT effect_flags,
    UINT secondary_flags,
    const void * secondary_data,
    SIZE_T secondary_data_size,
    ID3DBlob ** shader,
    ID3DBlob ** error_messages,
    unsigned int compiler_version )
```

As [D3DCompile2\(\)](#), but with an extra argument that allows targeting different d3dcompiler versions.

Parameters

<i>compiler_version</i>	The d3dcompiler version to target. This should be set to the numerical value in the d3dcompiler library name. E.g. to target the behaviour of d3dcompiler_36.dll, set this parameter to 36.
-------------------------	---

Since

1.14

#### 4.7.2.3 D3DDisassemble()

```
VKD3D_UTILS_API HRESULT WINAPI D3DDisassemble (
    const void * data,
    SIZE_T data_size,
    UINT flags,
    const char * comments,
    ID3DBlob ** blob )
```

Since

1.11

#### 4.7.2.4 D3DGetBlobPart()

```
VKD3D_UTILS_API HRESULT WINAPI D3DGetBlobPart (
    const void * data,
    SIZE_T data_size,
    D3D_BLOB_PART part,
    UINT flags,
    ID3DBlob ** blob )
```

Since

1.10

#### 4.7.2.5 D3DGetDebugInfo()

```
VKD3D_UTILS_API HRESULT WINAPI D3DGetDebugInfo (
    const void * data,
    SIZE_T data_size,
    ID3DBlob ** blob )
```

Since

1.10

#### 4.7.2.6 D3DGetInputAndOutputSignatureBlob()

```
VKD3D_UTILS_API HRESULT WINAPI D3DGetInputAndOutputSignatureBlob (
    const void * data,
    SIZE_T data_size,
    ID3DBlob ** blob )
```

Since

1.10

#### 4.7.2.7 D3DGetInputSignatureBlob()

```
VKD3D_UTILS_API HRESULT WINAPI D3DGetInputSignatureBlob (
    const void * data,
    SIZE_T data_size,
    ID3DBlob ** blob )
```

Since

1.10

#### 4.7.2.8 D3DGetOutputSignatureBlob()

```
VKD3D_UTILS_API HRESULT WINAPI D3DGetOutputSignatureBlob (
    const void * data,
    SIZE_T data_size,
    ID3DBlob ** blob )
```

Since

1.10

#### 4.7.2.9 D3DReflect()

```
VKD3D_UTILS_API HRESULT WINAPI D3DReflect (
    const void * data,
    SIZE_T data_size,
    REFIID iid,
    void ** reflection )
```

Since

1.11

#### 4.7.2.10 D3DStripShader()

```
VKD3D_UTILS_API HRESULT WINAPI D3DStripShader (
    const void * data,
    SIZE_T data_size,
    UINT flags,
    ID3DBlob ** blob )
```

Since

1.10

#### 4.7.2.11 vkd3d\_utils\_set\_log\_callback()

```
VKD3D_UTILS_API void vkd3d_utils_set_log_callback (
    PFN_vkd3d_log callback )
```

Set a callback to be called when vkd3d-utils outputs debug logging.

If NULL, or if this function has not been called, libvkd3d-utils will print all enabled log output to stderr.

Calling this function will also set the log callback for libvkd3d and libvkd3d-shader.

Parameters

<i>callback</i>	Callback function to set.
-----------------	---------------------------

Since

1.4

## 4.8 vkd3d\_utils.h

[Go to the documentation of this file.](#)

```
1 /*
2  * Copyright 2016 Józef Kucia for CodeWeavers
3  *
4  * This library is free software; you can redistribute it and/or
5  * modify it under the terms of the GNU Lesser General Public
6  * License as published by the Free Software Foundation; either
7  * version 2.1 of the License, or (at your option) any later version.
8  *
9  * This library is distributed in the hope that it will be useful,
10 * but WITHOUT ANY WARRANTY; without even the implied warranty of
11 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
12 * Lesser General Public License for more details.
13 *
14 * You should have received a copy of the GNU Lesser General Public
15 * License along with this library; if not, write to the Free Software
16 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
17 */
18
19 #ifndef __VKD3D_UTILS_H
20 #define __VKD3D_UTILS_H
21
22 #include <vkd3d.h>
```

```

23 #include <vkd3d_d3dcompiler_types.h>
24
25 #ifndef VKD3D_UTILS_API_VERSION
26 #define VKD3D_UTILS_API_VERSION VKD3D_API_VERSION_1_0
27 #endif
28
29 #ifdef __cplusplus
30 extern "C" {
31 #endif /* __cplusplus */
32
33 #define VKD3D_WAIT_OBJECT_0 (0)
34 #define VKD3D_WAIT_TIMEOUT (1)
35 #define VKD3D_WAIT_FAILED (~0u)
36 #define VKD3D_INFINITE (~0u)
37
38 #ifdef LIBVKD3D_UTILS_SOURCE
39 # define VKD3D_UTILS_API VKD3D_EXPORT
40 #else
41 # define VKD3D_UTILS_API VKD3D_IMPORT
42 #endif
43
44 /* 1.0 */
45 VKD3D_UTILS_API HANDLE vkd3d_create_event(void);
46 VKD3D_UTILS_API HRESULT vkd3d_signal_event(HANDLE event);
47 VKD3D_UTILS_API unsigned int vkd3d_wait_event(HANDLE event, unsigned int milliseconds);
48 VKD3D_UTILS_API void vkd3d_destroy_event(HANDLE event);
49
50 #if __STDC_VERSION__ >= 199901L || __cplusplus >= 201103L
51 # define D3D12CreateDevice(...) D3D12CreateDeviceVKD3D(__VA_ARGS__, VKD3D_UTILS_API_VERSION)
52 #else
53 # define D3D12CreateDevice(a, b, c, d) D3D12CreateDeviceVKD3D(a, b, c, d, VKD3D_UTILS_API_VERSION)
54 #endif
55
56 VKD3D_UTILS_API HRESULT WINAPI D3D12CreateRootSignatureDeserializer(
57     const void *data, SIZE_T data_size, REFIID iid, void **deserializer);
58 VKD3D_UTILS_API HRESULT WINAPI D3D12GetDebugInterface(REFIID iid, void **debug);
59 VKD3D_UTILS_API HRESULT WINAPI D3D12SerializeRootSignature(const D3D12_ROOT_SIGNATURE_DESC *desc,
60     D3D_ROOT_SIGNATURE_VERSION version, ID3DBlob **blob, ID3DBlob **error_blob);
61
62 /* 1.2 */
63 VKD3D_UTILS_API HRESULT WINAPI D3D12CreateDeviceVKD3D(IUnknown *adapter, D3D_FEATURE_LEVEL feature_level,
64     REFIID iid, void **device, enum vkd3d_api_version api_version);
65 VKD3D_UTILS_API HRESULT WINAPI D3D12CreateVersionedRootSignatureDeserializer(const void *data,
66     SIZE_T data_size, REFIID iid, void **deserializer);
67 VKD3D_UTILS_API HRESULT WINAPI D3D12SerializeVersionedRootSignature(const
68     D3D12_VERSIONED_ROOT_SIGNATURE_DESC *desc,
69     ID3DBlob **blob, ID3DBlob **error_blob);
70
71 /* 1.3 */
72 VKD3D_UTILS_API HRESULT WINAPI D3DCompile(const void *data, SIZE_T data_size, const char *filename,
73     const D3D_SHADER_MACRO *defines, ID3DInclude *include, const char *entrypoint,
74     const char *target, UINT flags, UINT effect_flags, ID3DBlob **shader, ID3DBlob **error_messages);
75 VKD3D_UTILS_API HRESULT WINAPI D3DCompile2(const void *data, SIZE_T data_size, const char *filename,
76     const D3D_SHADER_MACRO *defines, ID3DInclude *include, const char *entrypoint,
77     const char *target, UINT flags, UINT effect_flags, UINT secondary_flags,
78     const void *secondary_data, SIZE_T secondary_data_size, ID3DBlob **shader,
79     ID3DBlob **error_messages);
80 VKD3D_UTILS_API HRESULT WINAPI D3DCreateBlob(SIZE_T data_size, ID3DBlob **blob);
81 VKD3D_UTILS_API HRESULT WINAPI D3DPreprocess(const void *data, SIZE_T size, const char *filename,
82     const D3D_SHADER_MACRO *defines, ID3DInclude *include,
83     ID3DBlob **shader, ID3DBlob **error_messages);
84
85 VKD3D_UTILS_API void vkd3d_utils_set_log_callback(PFN_vkd3d_log callback);
86
87 VKD3D_UTILS_API HRESULT WINAPI D3DGetBlobPart(const void *data,
88     SIZE_T data_size, D3D_BLOB_PART part, UINT flags, ID3DBlob **blob);
89 VKD3D_UTILS_API HRESULT WINAPI D3DGetDebugInfo(const void *data, SIZE_T data_size, ID3DBlob **blob);
90 VKD3D_UTILS_API HRESULT WINAPI D3DGetInputAndOutputSignatureBlob(const void *data, SIZE_T data_size,
91     ID3DBlob **blob);
92 VKD3D_UTILS_API HRESULT WINAPI D3DGetInputSignatureBlob(const void *data, SIZE_T data_size, ID3DBlob
93     **blob);
94 VKD3D_UTILS_API HRESULT WINAPI D3DGetOutputSignatureBlob(const void *data, SIZE_T data_size, ID3DBlob
95     **blob);
96 VKD3D_UTILS_API HRESULT WINAPI D3DStripShader(const void *data, SIZE_T data_size, UINT flags, ID3DBlob
97     **blob);
98
99 VKD3D_UTILS_API HRESULT WINAPI D3DDisassemble(const void *data,
100     SIZE_T data_size, UINT flags, const char *comments, ID3DBlob **blob);
101 VKD3D_UTILS_API HRESULT WINAPI D3DReflect(const void *data, SIZE_T data_size, REFIID iid, void
102     **reflection);
103
104 VKD3D_UTILS_API HRESULT WINAPI D3DCompile2VKD3D(const void *data, SIZE_T data_size, const char
105     *filename,
106     const D3D_SHADER_MACRO *defines, ID3DInclude *include, const char *entrypoint,
107     const char *target, UINT flags, UINT effect_flags, UINT secondary_flags,
108     const void *secondary_data, SIZE_T secondary_data_size, ID3DBlob **shader,
109     ID3DBlob **error_messages, unsigned int compiler_version);
110

```

```
151 #ifdef __cplusplus
152 }
153 #endif /* __cplusplus */
154
155 #endif /* __VKD3D_UTILS_H */
```



# Index

[/builds/nsivov/vkd3d/include/vkd3d.h](#), [63](#), [70](#)  
[/builds/nsivov/vkd3d/include/vkd3d\\_shader.h](#), [73](#), [126](#)  
[/builds/nsivov/vkd3d/include/vkd3d\\_types.h](#), [141](#), [143](#)  
[/builds/nsivov/vkd3d/include/vkd3d\\_utils.h](#), [143](#), [149](#)

**adapter\_luid**  
    [vkd3d\\_device\\_create\\_info](#), [10](#)  
**api\_version**  
    [vkd3d\\_application\\_info](#), [8](#)  
**application\_name**  
    [vkd3d\\_application\\_info](#), [8](#)  
**binding\_offsets**  
    [vkd3d\\_shader\\_descriptor\\_offset\\_info](#), [25](#)  
**code**  
    [vkd3d\\_shader\\_code](#), [17](#)  
**count**  
    [vkd3d\\_shader\\_descriptor\\_binding](#), [23](#)  
    [vkd3d\\_shader\\_descriptor\\_info](#), [24](#)  
**D3DCompile2**  
    [vkd3d\\_utils.h](#), [145](#)  
**D3DCompile2VKD3D**  
    [vkd3d\\_utils.h](#), [146](#)  
**D3DDisassemble**  
    [vkd3d\\_utils.h](#), [147](#)  
**D3DGetBlobPart**  
    [vkd3d\\_utils.h](#), [147](#)  
**D3DGetDebugInfo**  
    [vkd3d\\_utils.h](#), [147](#)  
**D3DGetInputAndOutputSignatureBlob**  
    [vkd3d\\_utils.h](#), [147](#)  
**D3DGetInputSignatureBlob**  
    [vkd3d\\_utils.h](#), [148](#)  
**D3DGetOutputSignatureBlob**  
    [vkd3d\\_utils.h](#), [148](#)  
**D3DReflect**  
    [vkd3d\\_utils.h](#), [148](#)  
**D3DStripShader**  
    [vkd3d\\_utils.h](#), [148](#)  
**descriptor\_table\_offset**  
    [vkd3d\\_shader\\_descriptor\\_offset\\_info](#), [26](#)  
**device\_extensions**  
    [vkd3d\\_device\\_create\\_info](#), [10](#)  
**engine\_name**  
    [vkd3d\\_application\\_info](#), [8](#)  
**engine\_version**  
    [vkd3d\\_application\\_info](#), [8](#)  
**entry\_point**

[vkd3d\\_shader\\_hlsl\\_source\\_info](#), [30](#)  
**extensions**  
    [vkd3d\\_optional\\_device\\_extensions\\_info](#), [16](#)  
    [vkd3d\\_optional\\_instance\\_extensions\\_info](#), [17](#)  
**f32**  
    [vkd3d\\_shader\\_parameter\\_immediate\\_constant](#), [36](#)  
**f32\_vec4**  
    [vkd3d\\_shader\\_parameter\\_immediate\\_constant1](#),  
        [37](#)  
**flags**  
    [vkd3d\\_image\\_resource\\_create\\_info](#), [12](#)  
**id**  
    [vkd3d\\_shader\\_parameter\\_specialization\\_constant](#),  
        [39](#)  
**instance**  
    [vkd3d\\_device\\_create\\_info](#), [10](#)  
**instance\_create\_info**  
    [vkd3d\\_device\\_create\\_info](#), [10](#)  
**instance\_extensions**  
    [vkd3d\\_instance\\_create\\_info](#), [14](#)  
**macros**  
    [vkd3d\\_shader\\_preprocess\\_info](#), [40](#)  
**minimum\_feature\_level**  
    [vkd3d\\_device\\_create\\_info](#), [10](#)  
**name**  
    [vkd3d\\_shader\\_macro](#), [32](#)  
**next**  
    [vkd3d\\_shader\\_compile\\_info](#), [21](#)  
**options**  
    [vkd3d\\_shader\\_compile\\_info](#), [21](#)  
**output\_signature\_index**  
    [vkd3d\\_shader\\_varying\\_map](#), [59](#)  
**parent**  
    [vkd3d\\_device\\_create\\_info](#), [10](#)  
**pfn\_close\_include**  
    [vkd3d\\_shader\\_preprocess\\_info](#), [40](#)  
**pfn\_create\_thread**  
    [vkd3d\\_instance\\_create\\_info](#), [14](#)  
**pfn\_join\_thread**  
    [vkd3d\\_instance\\_create\\_info](#), [14](#)  
**pfn\_open\_include**  
    [vkd3d\\_shader\\_preprocess\\_info](#), [41](#)  
**PFN\_vk3d\_queue\_signal\_on\_cpu**  
    [vkd3d.h](#), [66](#)  
**PFN\_vk3d\_set\_log\_callback**

- vkd3d.h, 67
- PFN\_vkd3d\_shader\_build\_varying\_map
  - vkd3d\_shader.h, 85
- PFN\_vkd3d\_shader\_close\_include
  - vkd3d\_shader.h, 85
- PFN\_vkd3d\_shader\_free\_dxbc
  - vkd3d\_shader.h, 85
- PFN\_vkd3d\_shader\_free\_scan\_combined\_resource\_sampler
  - vkd3d\_shader.h, 86
- PFN\_vkd3d\_shader\_free\_scan\_signature\_info
  - vkd3d\_shader.h, 86
- PFN\_vkd3d\_shader\_open\_include
  - vkd3d\_shader.h, 86
- PFN\_vkd3d\_shader\_parse\_dxbc
  - vkd3d\_shader.h, 87
- PFN\_vkd3d\_shader\_preprocess
  - vkd3d\_shader.h, 87
- PFN\_vkd3d\_shader\_serialize\_dxbc
  - vkd3d\_shader.h, 87
- PFN\_vkd3d\_shader\_set\_log\_callback
  - vkd3d\_shader.h, 87
- pfn\_vkGetInstanceProcAddr
  - vkd3d\_instance\_create\_info, 15
- present\_state
  - vkd3d\_image\_resource\_create\_info, 13
- register\_index
  - vkd3d\_shader\_resource\_binding, 43
- register\_space
  - vkd3d\_shader\_push\_constant\_buffer, 42
  - vkd3d\_shader\_resource\_binding, 43
  - vkd3d\_shader\_uav\_counter\_binding, 58
- resource\_space
  - vkd3d\_shader\_combined\_resource\_sampler, 19
- sampler\_space
  - vkd3d\_shader\_combined\_resource\_sampler, 19
- set
  - vkd3d\_shader\_descriptor\_binding, 23
  - vkd3d\_shader\_parameter\_buffer, 35
- source\_name
  - vkd3d\_shader\_compile\_info, 21
- stream\_index
  - vkd3d\_shader\_signature\_element, 54
- sysval\_semantic
  - vkd3d\_shader\_signature\_element, 54
- tag
  - vkd3d\_shader\_dxbc\_desc, 28
- ticks\_per\_second
  - vkd3d\_host\_time\_domain\_info, 12
- uav\_counter\_offsets
  - vkd3d\_shader\_descriptor\_offset\_info, 26
- used\_mask
  - vkd3d\_shader\_signature\_element, 54
- value
  - vkd3d\_shader\_compile\_option, 22
- vkd3d\_shader\_macro, 32
- varying\_map
  - vkd3d\_shader\_varying\_map\_info, 61
- version
  - vkd3d\_shader\_dxbc\_desc, 28
- vk\_physical\_device
  - vkd3d\_device\_create\_info, 11
- vkd3d
  - PFN\_vkd3d\_queue\_signal\_on\_cpu, 66
  - PFN\_vkd3d\_set\_log\_callback, 67
  - vkd3d\_acquire\_vk\_queue, 68
  - vkd3d\_queue\_signal\_on\_cpu, 68
  - vkd3d\_release\_vk\_queue, 69
  - VKD3D\_RESOURCE\_INITIAL\_STATE\_TRANSITION, 66
  - vkd3d\_set\_log\_callback, 69
  - vkd3d\_structure\_type, 67
  - VKD3D\_STRUCTURE\_TYPE\_APPLICATION\_INFO, 68
  - VKD3D\_STRUCTURE\_TYPE\_DEVICE\_CREATE\_INFO, 67
  - VKD3D\_STRUCTURE\_TYPE\_HOST\_TIME\_DOMAIN\_INFO, 68
  - VKD3D\_STRUCTURE\_TYPE\_IMAGE\_RESOURCE\_CREATE\_INFO, 67
  - VKD3D\_STRUCTURE\_TYPE\_INSTANCE\_CREATE\_INFO, 67
  - VKD3D\_STRUCTURE\_TYPE\_OPTIONAL\_DEVICE\_EXTENSIONS, 67
  - VKD3D\_STRUCTURE\_TYPE\_OPTIONAL\_INSTANCE\_EXTENSIONS, 67
- vkd3d\_acquire\_vk\_queue
  - vkd3d.h, 68
- vkd3d\_application\_info, 7
  - api\_version, 8
  - application\_name, 8
  - engine\_name, 8
  - engine\_version, 8
- vkd3d\_device\_create\_info, 9
  - adapter\_luid, 10
  - device\_extensions, 10
  - instance, 10
  - instance\_create\_info, 10
  - minimum\_feature\_level, 10
  - parent, 10
  - vk\_physical\_device, 11
- VKD3D\_ERROR
  - vkd3d\_types.h, 142
- VKD3D\_ERROR\_INVALID\_ARGUMENT
  - vkd3d\_types.h, 142
- VKD3D\_ERROR\_INVALID\_SHADER
  - vkd3d\_types.h, 142
- VKD3D\_ERROR\_KEY\_ALREADY\_EXISTS
  - vkd3d\_types.h, 142
- VKD3D\_ERROR\_MORE\_DATA
  - vkd3d\_types.h, 142
- VKD3D\_ERROR\_NOT\_FOUND
  - vkd3d\_types.h, 142

- VKD3D\_ERROR\_NOT\_IMPLEMENTED
  - vk3d\_types.h, [142](#)
- VKD3D\_ERROR\_OUT\_OF\_MEMORY
  - vk3d\_types.h, [142](#)
- VKD3D\_FALSE
  - vk3d\_types.h, [142](#)
- vk3d\_host\_time\_domain\_info, [11](#)
  - ticks\_per\_second, [12](#)
- vk3d\_image\_resource\_create\_info, [12](#)
  - flags, [12](#)
  - present\_state, [13](#)
- vk3d\_instance\_create\_info, [13](#)
  - instance\_extensions, [14](#)
  - pfn\_create\_thread, [14](#)
  - pfn\_join\_thread, [14](#)
  - pfn\_vkGetInstanceProcAddr, [15](#)
  - wchar\_size, [15](#)
- VKD3D\_OK
  - vk3d\_types.h, [142](#)
- vk3d\_optional\_device\_extensions\_info, [15](#)
  - extensions, [16](#)
- vk3d\_optional\_instance\_extensions\_info, [16](#)
  - extensions, [17](#)
- vk3d\_queue\_signal\_on\_cpu
  - vk3d.h, [68](#)
- vk3d\_release\_vk\_queue
  - vk3d.h, [69](#)
- VKD3D\_RESOURCE\_INITIAL\_STATE\_TRANSITION
  - vk3d.h, [66](#)
- vk3d\_result
  - vk3d\_types.h, [142](#)
- vk3d\_set\_log\_callback
  - vk3d.h, [69](#)
- vk3d\_shader.h
  - PFN\_vk3d\_shader\_build\_varying\_map, [85](#)
  - PFN\_vk3d\_shader\_close\_include, [85](#)
  - PFN\_vk3d\_shader\_free\_dxbc, [85](#)
  - PFN\_vk3d\_shader\_free\_scan\_combined\_resource\_sample, [86](#)
  - PFN\_vk3d\_shader\_free\_scan\_signature\_info, [86](#)
  - PFN\_vk3d\_shader\_open\_include, [86](#)
  - PFN\_vk3d\_shader\_parse\_dxbc, [87](#)
  - PFN\_vk3d\_shader\_preprocess, [87](#)
  - PFN\_vk3d\_shader\_serialize\_dxbc, [87](#)
  - PFN\_vk3d\_shader\_set\_log\_callback, [87](#)
  - vk3d\_shader\_api\_version, [88](#)
  - vk3d\_shader\_build\_varying\_map, [113](#)
  - vk3d\_shader\_compile, [113](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_API\_VERSION, [91](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_BACKCOMPAT\_MATRIX\_ORDER, [88](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_BACKWARD\_COMPATIBILITY, [92](#)
  - vk3d\_shader\_compile\_option\_backward\_compatibility, [88](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_BUFFER\_UAV, [91](#)
  - vk3d\_shader\_compile\_option\_buffer\_uav, [89](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_BUFFER\_UAV\_STORAGE, [89](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_BUFFER\_UAV\_STORAGE, [89](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_CHILD\_EFFECT, [92](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_DOUBLE\_AS\_FLOAT\_ALIAS, [88](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE, [92](#)
  - vk3d\_shader\_compile\_option\_feature\_flags, [89](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE\_FLOAT64, [89](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE\_INT64, [89](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE\_WAVE\_OPS, [90](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE\_ZERO\_INITIALIZATION, [90](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FORMATTING, [91](#)
  - vk3d\_shader\_compile\_option\_formatting\_flags, [90](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FORMATTING\_IO\_SIGNAL, [90](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FRAGMENT\_COORDINATE\_ORIGIN, [92](#)
  - vk3d\_shader\_compile\_option\_fragment\_coordinate\_origin, [90](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FRAGMENT\_COORDINATE\_ORIGIN, [91](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_FRAGMENT\_COORDINATE\_ORIGIN, [91](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_INCLUDE\_EMPTY\_BUFFER, [93](#)
  - vk3d\_shader\_compile\_option\_name, [91](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_PACK\_MATRIX\_ORDER, [92](#)
  - vk3d\_shader\_compile\_option\_pack\_matrix\_order, [93](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_STRIP\_DEBUG, [91](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_TYPED\_UAV, [91](#)
  - vk3d\_shader\_compile\_option\_typed\_uav, [93](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_TYPED\_UAV\_READ\_FORMAT, [93](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_TYPED\_UAV\_READ\_FORMAT, [93](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_WARN\_IMPLICIT\_TRUNCATION, [93](#)
  - VKD3D\_SHADER\_COMPILE\_OPTION\_WRITE\_TESS\_GEOM\_POINTS, [91](#)
  - VKD3D\_SHADER\_COMPONENT\_BOOL, [94](#)
  - VKD3D\_SHADER\_COMPONENT\_DOUBLE, [94](#)
  - VKD3D\_SHADER\_COMPONENT\_FLOAT, [94](#)
  - VKD3D\_SHADER\_COMPONENT\_FLOAT16, [94](#)

- VKD3D\_SHADER\_COMPONENT\_INT, 94
- VKD3D\_SHADER\_COMPONENT\_INT16, 94
- VKD3D\_SHADER\_COMPONENT\_INT64, 94
- vkd3d\_shader\_component\_type, 94
- VKD3D\_SHADER\_COMPONENT\_UINT, 94
- VKD3D\_SHADER\_COMPONENT\_UINT16, 94
- VKD3D\_SHADER\_COMPONENT\_UINT64, 94
- VKD3D\_SHADER\_COMPONENT\_VOID, 94
- vkd3d\_shader\_convert\_root\_signature, 115
- VKD3D\_SHADER\_D3DBC\_BOOL\_CONSTANT\_REGISTER, 95
- vkd3d\_shader\_d3dbc\_constant\_register, 94
- VKD3D\_SHADER\_D3DBC\_FLOAT\_CONSTANT\_REGISTER, 95
- VKD3D\_SHADER\_D3DBC\_INT\_CONSTANT\_REGISTER, 95
- vkd3d\_shader\_descriptor\_info\_flag, 95
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_RAW\_BUFFER, 95
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_SAMPLER\_COMPARISON\_MODE, 95
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_UAV\_ATOMIC, 95
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_UAV\_COUNT, 95
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_UAV\_READ, 95
- VKD3D\_SHADER\_DESCRIPTOR\_RANGE\_FLAG\_DESCRIPTOR\_STATIC\_PARAMETER\_BUFFER\_BOUNDS\_CHECK, 96
- vkd3d\_shader\_descriptor\_range\_flags, 95
- vkd3d\_shader\_descriptor\_type, 96
- VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_CBV, 96
- VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_SAMPLER, 96
- VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_SRV, 96
- VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_UAV, 96
- vkd3d\_shader\_find\_signature\_element, 115
- VKD3D\_SHADER\_FOG\_FRAGMENT\_EXP, 97
- VKD3D\_SHADER\_FOG\_FRAGMENT\_EXP2, 97
- VKD3D\_SHADER\_FOG\_FRAGMENT\_LINEAR, 97
- vkd3d\_shader\_fog\_fragment\_mode, 96
- VKD3D\_SHADER\_FOG\_FRAGMENT\_NONE, 97
- vkd3d\_shader\_fog\_source, 97
- VKD3D\_SHADER\_FOG\_SOURCE\_FOG, 97
- VKD3D\_SHADER\_FOG\_SOURCE\_FOG\_OR\_SPECULAR, 98
- VKD3D\_SHADER\_FOG\_SOURCE\_W, 98
- VKD3D\_SHADER\_FOG\_SOURCE\_Z, 98
- vkd3d\_shader\_free\_dxbc, 116
- vkd3d\_shader\_free\_messages, 116
- vkd3d\_shader\_free\_root\_signature, 116
- vkd3d\_shader\_free\_scan\_combined\_resource\_sampler\_info, 117
- vkd3d\_shader\_free\_scan\_descriptor\_info, 117
- vkd3d\_shader\_free\_scan\_signature\_info, 117
- vkd3d\_shader\_free\_shader\_code, 118
- vkd3d\_shader\_free\_shader\_signature, 118
- vkd3d\_shader\_get\_supported\_source\_types, 118
- vkd3d\_shader\_get\_supported\_target\_types, 119
- vkd3d\_shader\_get\_version, 119
- VKD3D\_SHADER\_LOG\_ERROR, 98
- VKD3D\_SHADER\_LOG\_INFO, 98
- vkd3d\_shader\_log\_level, 98
- VKD3D\_SHADER\_LOG\_NONE, 98
- VKD3D\_SHADER\_LOG\_WARNING, 98
- vkd3d\_shader\_minimum\_precision, 98
- VKD3D\_SHADER\_MINIMUM\_PRECISION\_FIXED\_8\_2, 98
- VKD3D\_SHADER\_MINIMUM\_PRECISION\_FLOAT\_16, 98
- VKD3D\_SHADER\_MINIMUM\_PRECISION\_INT\_16, 98
- VKD3D\_SHADER\_MINIMUM\_PRECISION\_UINT\_16, 98
- vkd3d\_shader\_parameter\_data\_type, 98
- VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_FLOAT32, 98
- VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_FLOAT32\_VEC4, 98
- VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_UINT32, 98
- vkd3d\_shader\_parameter\_name, 99
- VKD3D\_SHADER\_PARAMETER\_NAME\_ALPHA\_TEST\_FUNC, 100
- VKD3D\_SHADER\_PARAMETER\_NAME\_CLIP\_PLANE\_BOUNDS\_CHECK, 100
- VKD3D\_SHADER\_PARAMETER\_NAME\_CLIP\_PLANE\_0, 101
- VKD3D\_SHADER\_PARAMETER\_NAME\_CLIP\_PLANE\_MASK, 101
- VKD3D\_SHADER\_PARAMETER\_NAME\_FLAT\_INTERPOLATION, 100
- VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_COLOUR, 104
- VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_END, 104
- VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_FRAGMENT\_MODE, 104
- VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_SCALE, 105
- VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_SOURCE, 105
- VKD3D\_SHADER\_PARAMETER\_NAME\_POINT\_SIZE, 102
- VKD3D\_SHADER\_PARAMETER\_NAME\_POINT\_SIZE\_MAX, 102
- VKD3D\_SHADER\_PARAMETER\_NAME\_POINT\_SIZE\_MIN, 102
- VKD3D\_SHADER\_PARAMETER\_NAME\_POINT\_SPRITE, 102
- VKD3D\_SHADER\_PARAMETER\_NAME\_RASTERIZER\_SAMPLE, 99
- vkd3d\_shader\_parameter\_type, 105
- VKD3D\_SHADER\_PARAMETER\_TYPE\_BUFFER, 105

- VKD3D\_SHADER\_PARAMETER\_TYPE\_IMMEDIATE\_CONSTANT, [105](#)
- VKD3D\_SHADER\_PARAMETER\_TYPE\_SPECIALIZATION\_CONSTANT, [105](#)
- vkd3d\_shader\_parse\_dxbc, [120](#)
- vkd3d\_shader\_parse\_dxbc\_flags, [106](#)
- VKD3D\_SHADER\_PARSE\_DXBC\_IGNORE\_CHECKSUM, [106](#)
- vkd3d\_shader\_parse\_input\_signature, [120](#)
- vkd3d\_shader\_parse\_root\_signature, [122](#)
- vkd3d\_shader\_preprocess, [123](#)
- VKD3D\_SHADER\_RESOURCE\_BUFFER, [107](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_CONTINUED, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_DOUBLE, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_FLOAT, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_INT, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_MIXED, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_SNORM, [106](#)
- vkd3d\_shader\_resource\_data\_type, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_UINT, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_UNORM, [106](#)
- VKD3D\_SHADER\_RESOURCE\_NONE, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_1D, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_1DARRAY, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_2D, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_2DARRAY, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_2DMS, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_2DMSARRAY, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_3D, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_CUBE, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_CUBEARRAY, [107](#)
- vkd3d\_shader\_resource\_type, [107](#)
- vkd3d\_shader\_scan, [123](#)
- vkd3d\_shader\_serialize\_dxbc, [124](#)
- vkd3d\_shader\_serialize\_root\_signature, [125](#)
- vkd3d\_shader\_set\_log\_callback, [126](#)
- VKD3D\_SHADER\_SOURCE\_D3D\_BYTECODE, [107](#)
- VKD3D\_SHADER\_SOURCE\_DXBC\_DXIL, [108](#)
- VKD3D\_SHADER\_SOURCE\_DXBC\_TPF, [107](#)
- VKD3D\_SHADER\_SOURCE\_FX, [108](#)
- VKD3D\_SHADER\_SOURCE\_HLSL, [107](#)
- VKD3D\_SHADER\_SOURCE\_NONE, [107](#)
- VKD3D\_SHADER\_SOURCE\_SPIRV, [107](#)
- vkd3d\_shader\_spirv\_environment, [108](#)
- VKD3D\_SHADER\_SPIRV\_ENVIRONMENT\_VULKAN\_1\_1, [108](#)
- vkd3d\_shader\_spirv\_extension, [108](#)
- VKD3D\_SHADER\_SPIRV\_EXTENSION\_EXT\_DESCRIPTOR\_INDEXING, [108](#)
- VKD3D\_SHADER\_SPIRV\_EXTENSION\_EXT\_FRAGMENT\_SHADER\_INTERLOCK, [109](#)
- VKD3D\_SHADER\_SPIRV\_EXTENSION\_EXT\_STENCIL\_EXPORT, [108](#)
- VKD3D\_SHADER\_SPIRV\_EXTENSION\_EXT\_VIEWPORT\_INDEX\_0\_1, [108](#)
- vkd3d\_shader\_structure\_type, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_COMPILE\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_DESCRIPTOR\_OFFSET\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_HLSL\_SOURCE\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_INTERFACE\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_PARAMETER\_INFO, [110](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_PREPROCESS\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_COMBINED\_RESOURCES, [110](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_DESCRIPTOR\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_HULL\_SHADER\_THREADS, [110](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_SIGNATURE\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SPIRV\_DOMAIN\_SHADER\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SPIRV\_TARGET\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_TRANSFORM\_FEEDBACK\_INFO, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_VARYING\_MAP\_INFO, [110](#)
- VKD3D\_SHADER\_SV\_CLIP\_DISTANCE, [110](#)
- VKD3D\_SHADER\_SV\_COVERAGE, [111](#)
- VKD3D\_SHADER\_SV\_CULL\_DISTANCE, [110](#)
- VKD3D\_SHADER\_SV\_DEPTH, [111](#)
- VKD3D\_SHADER\_SV\_DEPTH\_GREATER\_EQUAL, [111](#)
- VKD3D\_SHADER\_SV\_DEPTH\_LESS\_EQUAL, [111](#)
- VKD3D\_SHADER\_SV\_INSTANCE\_ID, [110](#)
- VKD3D\_SHADER\_SV\_IS\_FRONT\_FACE, [110](#)
- VKD3D\_SHADER\_SV\_NONE, [110](#)
- VKD3D\_SHADER\_SV\_POSITION, [110](#)
- VKD3D\_SHADER\_SV\_PRIMITIVE\_ID, [110](#)
- VKD3D\_SHADER\_SV\_RENDER\_TARGET\_ARRAY\_INDEX, [110](#)
- VKD3D\_SHADER\_SV\_SAMPLE\_INDEX, [110](#)

- VKD3D\_SHADER\_SV\_STENCIL\_REF, 111
- VKD3D\_SHADER\_SV\_TARGET, 111
- VKD3D\_SHADER\_SV\_VERTEX\_ID, 110
- VKD3D\_SHADER\_SV\_VIEWPORT\_ARRAY\_INDEX, 110
- VKD3D\_SHADER\_SWIZZLE, 84
- VKD3D\_SHADER\_SWIZZLE\_MASK, 84
- vkd3d\_shader\_sysval\_semantic, 110
- VKD3D\_SHADER\_TARGET\_D3D\_ASM, 111
- VKD3D\_SHADER\_TARGET\_D3D\_BYTECODE, 112
- VKD3D\_SHADER\_TARGET\_DXBC\_TPF, 112
- VKD3D\_SHADER\_TARGET\_FX, 112
- VKD3D\_SHADER\_TARGET\_GLSL, 112
- VKD3D\_SHADER\_TARGET\_MSL, 112
- VKD3D\_SHADER\_TARGET\_NONE, 111
- VKD3D\_SHADER\_TARGET\_SPIRV\_BINARY, 111
- vkd3d\_shader\_target\_type, 111
- vkd3d\_shader\_visibility, 112
- VKD3D\_SHADER\_VISIBILITY\_ALL, 112
- VKD3D\_SHADER\_VISIBILITY\_COMPUTE, 112
- VKD3D\_SHADER\_VISIBILITY\_DOMAIN, 112
- VKD3D\_SHADER\_VISIBILITY\_GEOMETRY, 112
- VKD3D\_SHADER\_VISIBILITY\_HULL, 112
- VKD3D\_SHADER\_VISIBILITY\_PIXEL, 112
- VKD3D\_SHADER\_VISIBILITY\_VERTEX, 112
- vkd3d\_shader\_api\_version
  - vkd3d\_shader.h, 88
- vkd3d\_shader\_build\_varying\_map
  - vkd3d\_shader.h, 113
- vkd3d\_shader\_code, 17
  - code, 17
- vkd3d\_shader\_combined\_resource\_sampler, 18
  - resource\_space, 19
  - sampler\_space, 19
- vkd3d\_shader\_combined\_resource\_sampler\_info, 19
- vkd3d\_shader\_compile
  - vkd3d\_shader.h, 113
- vkd3d\_shader\_compile\_info, 20
  - next, 21
  - options, 21
  - source\_name, 21
- vkd3d\_shader\_compile\_option, 21
  - value, 22
- VKD3D\_SHADER\_COMPILE\_OPTION\_API\_VERSION
  - vkd3d\_shader.h, 91
- VKD3D\_SHADER\_COMPILE\_OPTION\_BACKCOMPAT\_COMPATIBILITY
  - vkd3d\_shader.h, 88
- VKD3D\_SHADER\_COMPILE\_OPTION\_BACKWARD\_COMPATIBILITY
  - vkd3d\_shader.h, 92
- vkd3d\_shader\_compile\_option\_backward\_compatibility
  - vkd3d\_shader.h, 88
- VKD3D\_SHADER\_COMPILE\_OPTION\_BUFFER\_UAV
  - vkd3d\_shader.h, 91
- vkd3d\_shader\_compile\_option\_buffer\_uav
  - vkd3d\_shader.h, 89
- VKD3D\_SHADER\_COMPILE\_OPTION\_BUFFER\_UAV\_STORAGE\_TEXEL\_OFFSET
  - vkd3d\_shader.h, 89
- VKD3D\_SHADER\_COMPILE\_OPTION\_BUFFER\_UAV\_STORAGE\_TEXEL\_OFFSET
  - vkd3d\_shader.h, 89
- VKD3D\_SHADER\_COMPILE\_OPTION\_CHILD\_EFFECT
  - vkd3d\_shader.h, 92
- VKD3D\_SHADER\_COMPILE\_OPTION\_DOUBLE\_AS\_FLOAT\_ALIAS
  - vkd3d\_shader.h, 88
- VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE
  - vkd3d\_shader.h, 92
- vkd3d\_shader\_compile\_option\_feature\_flags
  - vkd3d\_shader.h, 89
- VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE\_FLOAT64
  - vkd3d\_shader.h, 89
- VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE\_INT64
  - vkd3d\_shader.h, 89
- VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE\_WAVE\_OPS
  - vkd3d\_shader.h, 90
- VKD3D\_SHADER\_COMPILE\_OPTION\_FEATURE\_ZERO\_INITIALIZE\_ZERO
  - vkd3d\_shader.h, 90
- VKD3D\_SHADER\_COMPILE\_OPTION\_FORMATTING
  - vkd3d\_shader.h, 91
- vkd3d\_shader\_compile\_option\_formatting\_flags
  - vkd3d\_shader.h, 90
- VKD3D\_SHADER\_COMPILE\_OPTION\_FORMATTING\_IO\_SIGNATURES
  - vkd3d\_shader.h, 90
- VKD3D\_SHADER\_COMPILE\_OPTION\_FRAGMENT\_COORDINATE\_ORIGIN
  - vkd3d\_shader.h, 92
- vkd3d\_shader\_compile\_option\_fragment\_coordinate\_origin
  - vkd3d\_shader.h, 90
- VKD3D\_SHADER\_COMPILE\_OPTION\_FRAGMENT\_COORDINATE\_ORIGIN
  - vkd3d\_shader.h, 91
- VKD3D\_SHADER\_COMPILE\_OPTION\_FRAGMENT\_COORDINATE\_ORIGIN
  - vkd3d\_shader.h, 91
- VKD3D\_SHADER\_COMPILE\_OPTION\_INCLUDE\_EMPTY\_BUFFERS\_I
  - vkd3d\_shader.h, 93
- vkd3d\_shader\_compile\_option\_name
  - vkd3d\_shader.h, 91
- VKD3D\_SHADER\_COMPILE\_OPTION\_PACK\_MATRIX\_ORDER
  - vkd3d\_shader.h, 92
- vkd3d\_shader\_compile\_option\_pack\_matrix\_order
  - vkd3d\_shader.h, 93
- VKD3D\_SHADER\_COMPILE\_OPTION\_STRIP\_DEBUG
  - vkd3d\_shader.h, 91
- VKD3D\_SHADER\_COMPILE\_OPTION\_TYPED\_UAV
  - vkd3d\_shader.h, 91
- vkd3d\_shader\_compile\_option\_typed\_uav
  - vkd3d\_shader.h, 93
- VKD3D\_SHADER\_COMPILE\_OPTION\_TYPED\_UAV\_READ\_FORMAT\_I
  - vkd3d\_shader.h, 93
- VKD3D\_SHADER\_COMPILE\_OPTION\_TYPED\_UAV\_READ\_FORMAT\_I
  - vkd3d\_shader.h, 93
- VKD3D\_SHADER\_COMPILE\_OPTION\_WARN\_IMPLICIT\_TRUNCATION
  - vkd3d\_shader.h, 93
- VKD3D\_SHADER\_COMPILE\_OPTION\_WRITE\_TESS\_GEOM\_POINT\_S
  - vkd3d\_shader.h, 91
- VKD3D\_SHADER\_COMPONENT\_BOOL
  - vkd3d\_shader.h, 94
- VKD3D\_SHADER\_COMPONENT\_DOUBLE
  - vkd3d\_shader.h, 94

- VKD3D\_SHADER\_COMPONENT\_FLOAT
  - vk3d\_shader.h, [94](#)
- VKD3D\_SHADER\_COMPONENT\_FLOAT16
  - vk3d\_shader.h, [94](#)
- VKD3D\_SHADER\_COMPONENT\_INT
  - vk3d\_shader.h, [94](#)
- VKD3D\_SHADER\_COMPONENT\_INT16
  - vk3d\_shader.h, [94](#)
- VKD3D\_SHADER\_COMPONENT\_INT64
  - vk3d\_shader.h, [94](#)
- vk3d\_shader\_component\_type
  - vk3d\_shader.h, [94](#)
- VKD3D\_SHADER\_COMPONENT\_UINT
  - vk3d\_shader.h, [94](#)
- VKD3D\_SHADER\_COMPONENT\_UINT16
  - vk3d\_shader.h, [94](#)
- VKD3D\_SHADER\_COMPONENT\_UINT64
  - vk3d\_shader.h, [94](#)
- VKD3D\_SHADER\_COMPONENT\_VOID
  - vk3d\_shader.h, [94](#)
- vk3d\_shader\_convert\_root\_signature
  - vk3d\_shader.h, [115](#)
- VKD3D\_SHADER\_D3DBC\_BOOL\_CONSTANT\_REGISTER
  - vk3d\_shader.h, [95](#)
- vk3d\_shader\_d3dbc\_constant\_register
  - vk3d\_shader.h, [94](#)
- VKD3D\_SHADER\_D3DBC\_FLOAT\_CONSTANT\_REGISTER
  - vk3d\_shader.h, [95](#)
- VKD3D\_SHADER\_D3DBC\_INT\_CONSTANT\_REGISTER
  - vk3d\_shader.h, [95](#)
- vk3d\_shader\_descriptor\_binding, [22](#)
  - count, [23](#)
  - set, [23](#)
- vk3d\_shader\_descriptor\_info, [23](#)
  - count, [24](#)
- vk3d\_shader\_descriptor\_info\_flag
  - vk3d\_shader.h, [95](#)
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_RAW\_BUFFER
  - vk3d\_shader.h, [95](#)
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_SAMPLER\_COMPARISON\_MODE
  - vk3d\_shader.h, [95](#)
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_UAV\_ATOMIC\_COUNTER
  - vk3d\_shader.h, [95](#)
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_UAV\_COUNTER
  - vk3d\_shader.h, [95](#)
- VKD3D\_SHADER\_DESCRIPTOR\_INFO\_FLAG\_UAV\_READ
  - vk3d\_shader.h, [95](#)
- vk3d\_shader\_descriptor\_offset, [24](#)
- vk3d\_shader\_descriptor\_offset\_info, [24](#)
  - binding\_offsets, [25](#)
  - descriptor\_table\_offset, [26](#)
  - uav\_counter\_offsets, [26](#)
- vk3d\_shader\_descriptor\_range, [26](#)
- vk3d\_shader\_descriptor\_range1, [27](#)
- VKD3D\_SHADER\_DESCRIPTOR\_RANGE\_FLAG\_DESCRIPTOR\_BUFFER\_BOUNDS\_CHECKS
  - vk3d\_shader.h, [96](#)
- vk3d\_shader\_descriptor\_range\_flags
  - vk3d\_shader.h, [95](#)
- vk3d\_shader\_descriptor\_type
  - vk3d\_shader.h, [96](#)
- VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_CBV
  - vk3d\_shader.h, [96](#)
- VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_SAMPLER
  - vk3d\_shader.h, [96](#)
- VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_SRV
  - vk3d\_shader.h, [96](#)
- VKD3D\_SHADER\_DESCRIPTOR\_TYPE\_UAV
  - vk3d\_shader.h, [96](#)
- vk3d\_shader\_dxbc\_desc, [27](#)
  - tag, [28](#)
  - version, [28](#)
- vk3d\_shader\_dxbc\_section\_desc, [29](#)
- vk3d\_shader\_find\_signature\_element
  - vk3d\_shader.h, [115](#)
- VKD3D\_SHADER\_FOG\_FRAGMENT\_EXP
  - vk3d\_shader.h, [97](#)
- VKD3D\_SHADER\_FOG\_FRAGMENT\_EXP2
  - vk3d\_shader.h, [97](#)
- VKD3D\_SHADER\_FOG\_FRAGMENT\_LINEAR
  - vk3d\_shader.h, [97](#)
- vk3d\_shader\_fog\_fragment\_mode
  - vk3d\_shader.h, [96](#)
- VKD3D\_SHADER\_FOG\_FRAGMENT\_NONE
  - vk3d\_shader.h, [97](#)
- vk3d\_shader\_fog\_source
  - vk3d\_shader.h, [97](#)
- VKD3D\_SHADER\_FOG\_SOURCE\_FOG
  - vk3d\_shader.h, [97](#)
- VKD3D\_SHADER\_FOG\_SOURCE\_FOG\_OR\_SPECULAR\_W
  - vk3d\_shader.h, [98](#)
- VKD3D\_SHADER\_FOG\_SOURCE\_W
  - vk3d\_shader.h, [98](#)
- VKD3D\_SHADER\_FOG\_SOURCE\_Z
  - vk3d\_shader.h, [98](#)
- vk3d\_shader\_free\_dxbc
  - vk3d\_shader.h, [116](#)
- vk3d\_shader\_free\_messages
  - vk3d\_shader.h, [116](#)
- vk3d\_shader\_free\_root\_signature
  - vk3d\_shader.h, [116](#)
- vk3d\_shader\_free\_scan\_combined\_resource\_sampler\_info
  - vk3d\_shader.h, [117](#)
- vk3d\_shader\_free\_scan\_descriptor\_info
  - vk3d\_shader.h, [117](#)
- vk3d\_shader\_free\_scan\_signature\_info
  - vk3d\_shader.h, [117](#)
- vk3d\_shader\_free\_shader\_code
  - vk3d\_shader.h, [118](#)
- vk3d\_shader\_free\_shader\_signature
  - vk3d\_shader.h, [118](#)
- vk3d\_shader\_get\_supported\_source\_types
  - vk3d\_shader.h, [118](#)
- vk3d\_shader\_get\_supported\_buffer\_types
  - vk3d\_shader.h, [119](#)
- vk3d\_shader\_get\_version
  - vk3d\_shader.h, [119](#)

vkd3d\_shader\_hlsl\_source\_info, 30  
     entry\_point, 30  
 vkd3d\_shader\_interface\_info, 31  
 VKD3D\_SHADER\_LOG\_ERROR  
     vkd3d\_shader.h, 98  
 VKD3D\_SHADER\_LOG\_INFO  
     vkd3d\_shader.h, 98  
 vkd3d\_shader\_log\_level  
     vkd3d\_shader.h, 98  
 VKD3D\_SHADER\_LOG\_NONE  
     vkd3d\_shader.h, 98  
 VKD3D\_SHADER\_LOG\_WARNING  
     vkd3d\_shader.h, 98  
 vkd3d\_shader\_macro, 32  
     name, 32  
     value, 32  
 vkd3d\_shader\_minimum\_precision  
     vkd3d\_shader.h, 98  
 VKD3D\_SHADER\_MINIMUM\_PRECISION\_FIXED\_8\_2  
     vkd3d\_shader.h, 98  
 VKD3D\_SHADER\_MINIMUM\_PRECISION\_FLOAT\_16  
     vkd3d\_shader.h, 98  
 VKD3D\_SHADER\_MINIMUM\_PRECISION\_INT\_16  
     vkd3d\_shader.h, 98  
 VKD3D\_SHADER\_MINIMUM\_PRECISION\_UINT\_16  
     vkd3d\_shader.h, 98  
 vkd3d\_shader\_parameter, 33  
 vkd3d\_shader\_parameter1, 34  
 vkd3d\_shader\_parameter\_buffer, 35  
     set, 35  
 vkd3d\_shader\_parameter\_data\_type  
     vkd3d\_shader.h, 98  
 VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_FLOAT32  
     vkd3d\_shader.h, 99  
 VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_FLOAT32\_VEC4  
     vkd3d\_shader.h, 99  
 VKD3D\_SHADER\_PARAMETER\_DATA\_TYPE\_UINT32  
     vkd3d\_shader.h, 99  
 vkd3d\_shader\_parameter\_immediate\_constant, 36  
     f32, 36  
 vkd3d\_shader\_parameter\_immediate\_constant1, 36  
     f32\_vec4, 37  
 vkd3d\_shader\_parameter\_info, 38  
 vkd3d\_shader\_parameter\_name  
     vkd3d\_shader.h, 99  
 VKD3D\_SHADER\_PARAMETER\_NAME\_ALPHA\_TEST\_REF  
     vkd3d\_shader.h, 100  
 VKD3D\_SHADER\_PARAMETER\_NAME\_ALPHA\_TEST\_REF  
     vkd3d\_shader.h, 100  
 VKD3D\_SHADER\_PARAMETER\_NAME\_CLIP\_PLANE\_0  
     vkd3d\_shader.h, 101  
 VKD3D\_SHADER\_PARAMETER\_NAME\_CLIP\_PLANE\_MASK  
     vkd3d\_shader.h, 101  
 VKD3D\_SHADER\_PARAMETER\_NAME\_FLAT\_INTERPOLATION  
     vkd3d\_shader.h, 100  
 VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_COLOUR  
     vkd3d\_shader.h, 104  
 VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_END  
     vkd3d\_shader.h, 104  
 VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_FRAGMENT\_MODE  
     vkd3d\_shader.h, 104  
 VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_SCALE  
     vkd3d\_shader.h, 105  
 VKD3D\_SHADER\_PARAMETER\_NAME\_FOG\_SOURCE  
     vkd3d\_shader.h, 105  
 VKD3D\_SHADER\_PARAMETER\_NAME\_POINT\_SIZE  
     vkd3d\_shader.h, 102  
 VKD3D\_SHADER\_PARAMETER\_NAME\_POINT\_SIZE\_MAX  
     vkd3d\_shader.h, 102  
 VKD3D\_SHADER\_PARAMETER\_NAME\_POINT\_SIZE\_MIN  
     vkd3d\_shader.h, 102  
 VKD3D\_SHADER\_PARAMETER\_NAME\_POINT\_SPRITE  
     vkd3d\_shader.h, 103  
 VKD3D\_SHADER\_PARAMETER\_NAME\_RASTERIZER\_SAMPLE\_COUNT  
     vkd3d\_shader.h, 99  
 vkd3d\_shader\_parameter\_specialization\_constant, 39  
     id, 39  
 vkd3d\_shader\_parameter\_type  
     vkd3d\_shader.h, 105  
 VKD3D\_SHADER\_PARAMETER\_TYPE\_BUFFER  
     vkd3d\_shader.h, 105  
 VKD3D\_SHADER\_PARAMETER\_TYPE\_IMMEDIATE\_CONSTANT  
     vkd3d\_shader.h, 105  
 VKD3D\_SHADER\_PARAMETER\_TYPE\_SPECIALIZATION\_CONSTANT  
     vkd3d\_shader.h, 105  
 vkd3d\_shader\_parse\_dxbc  
     vkd3d\_shader.h, 120  
 vkd3d\_shader\_parse\_dxbc\_flags  
     vkd3d\_shader.h, 106  
 VKD3D\_SHADER\_PARSE\_DXBC\_IGNORE\_CHECKSUM  
     vkd3d\_shader.h, 106  
 vkd3d\_shader\_parse\_input\_signature  
     vkd3d\_shader.h, 120  
 vkd3d\_shader\_parse\_root\_signature  
     vkd3d\_shader.h, 122  
 vkd3d\_shader\_preprocess  
     vkd3d\_shader.h, 123  
 vkd3d\_shader\_preprocess\_info, 39  
     macros, 40  
     pfn\_close\_include, 40  
     pfn\_open\_include, 41  
 vkd3d\_shader\_push\_constant\_buffer, 41  
     register\_space, 42  
 vkd3d\_shader\_resource\_binding, 42  
     register\_index, 43  
     register\_space, 43  
 VKD3D\_SHADER\_RESOURCE\_BUFFER  
     vkd3d\_shader.h, 107  
 VKD3D\_SHADER\_RESOURCE\_DATA\_CONTINUED  
     vkd3d\_shader.h, 106  
 VKD3D\_SHADER\_RESOURCE\_DATA\_DOUBLE  
     vkd3d\_shader.h, 106  
 VKD3D\_SHADER\_RESOURCE\_DATA\_FLOAT  
     vkd3d\_shader.h, 106  
 VKD3D\_SHADER\_RESOURCE\_DATA\_INT  
     vkd3d\_shader.h, 106

- VKD3D\_SHADER\_RESOURCE\_DATA\_MIXED
  - `vk3d_shader.h`, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_SNORM
  - `vk3d_shader.h`, [106](#)
- `vk3d_shader_resource_data_type`
  - `vk3d_shader.h`, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_UINT
  - `vk3d_shader.h`, [106](#)
- VKD3D\_SHADER\_RESOURCE\_DATA\_UNORM
  - `vk3d_shader.h`, [106](#)
- VKD3D\_SHADER\_RESOURCE\_NONE
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_1D
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_1DARRAY
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_2D
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_2DARRAY
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_2DMS
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_2DMSARRAY
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_3D
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_CUBE
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_RESOURCE\_TEXTURE\_CUBEARRAY
  - `vk3d_shader.h`, [107](#)
- `vk3d_shader_resource_type`
  - `vk3d_shader.h`, [107](#)
- `vk3d_shader_root_constants`, [44](#)
- `vk3d_shader_root_descriptor`, [44](#)
- `vk3d_shader_root_descriptor1`, [44](#)
- `vk3d_shader_root_descriptor_table`, [45](#)
- `vk3d_shader_root_descriptor_table1`, [45](#)
- `vk3d_shader_root_parameter`, [46](#)
- `vk3d_shader_root_parameter1`, [46](#)
- `vk3d_shader_root_signature_desc`, [47](#)
- `vk3d_shader_root_signature_desc1`, [47](#)
- `vk3d_shader_scan`
  - `vk3d_shader.h`, [123](#)
- `vk3d_shader_scan_combined_resource_sampler_info`, [48](#)
- `vk3d_shader_scan_descriptor_info`, [49](#)
- `vk3d_shader_scan_hull_shader_tessellation_info`, [50](#)
- `vk3d_shader_scan_signature_info`, [51](#)
- `vk3d_shader_serialize_dxbc`
  - `vk3d_shader.h`, [124](#)
- `vk3d_shader_serialize_root_signature`
  - `vk3d_shader.h`, [125](#)
- `vk3d_shader_set_log_callback`
  - `vk3d_shader.h`, [126](#)
- `vk3d_shader_signature`, [53](#)
- `vk3d_shader_signature_element`, [53](#)
  - `stream_index`, [54](#)
  - `sysval_semantic`, [54](#)
  - `used_mask`, [54](#)
- VKD3D\_SHADER\_SOURCE\_D3D\_BYTECODE
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_SOURCE\_DXBC\_DXIL
  - `vk3d_shader.h`, [108](#)
- VKD3D\_SHADER\_SOURCE\_DXBC\_TPF
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_SOURCE\_FX
  - `vk3d_shader.h`, [108](#)
- VKD3D\_SHADER\_SOURCE\_HLSL
  - `vk3d_shader.h`, [107](#)
- VKD3D\_SHADER\_SOURCE\_NONE
  - `vk3d_shader.h`, [107](#)
- `vk3d_shader_source_type`
  - `vk3d_shader.h`, [107](#)
- `vk3d_shader_spirv_domain_shader_target_info`, [55](#)
- `vk3d_shader_spirv_environment`
  - `vk3d_shader.h`, [108](#)
- VKD3D\_SHADER\_SPIRV\_ENVIRONMENT\_VULKAN\_1\_1
  - `vk3d_shader.h`, [108](#)
- `vk3d_shader_spirv_extension`
  - `vk3d_shader.h`, [108](#)
- VKD3D\_SHADER\_SPIRV\_EXTENSION\_EXT\_DESCRIPTOR\_INDEXING
  - `vk3d_shader.h`, [108](#)
- VKD3D\_SHADER\_SPIRV\_EXTENSION\_EXT\_FRAGMENT\_SHADER\_INTERPOLATION\_ONLY
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_SPIRV\_EXTENSION\_EXT\_STENCIL\_EXPORT
  - `vk3d_shader.h`, [108](#)
- VKD3D\_SHADER\_SPIRV\_EXTENSION\_EXT\_VIEWPORT\_INDEX\_LAYER
  - `vk3d_shader.h`, [108](#)
- `vk3d_shader_spirv_target_info`, [55](#)
- `vk3d_shader_static_sampler_desc`, [56](#)
- `vk3d_shader_structure_type`
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_COMPILE\_INFO
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_DESCRIPTOR\_OFFSET\_INFO
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_HLSL\_SOURCE\_INFO
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_INTERFACE\_INFO
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_PARAMETER\_INFO
  - `vk3d_shader.h`, [110](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_PREPROCESS\_INFO
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_COMBINED\_RESOURCE
  - `vk3d_shader.h`, [110](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_DESCRIPTOR\_INFO
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_HULL\_SHADER\_TESSELLATION
  - `vk3d_shader.h`, [110](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SCAN\_SIGNATURE\_INFO
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SPIRV\_DOMAIN\_SHADER\_TARGET
  - `vk3d_shader.h`, [109](#)
- VKD3D\_SHADER\_STRUCTURE\_TYPE\_SPIRV\_TARGET\_INFO
  - `vk3d_shader.h`, [109](#)

VKD3D\_SHADER\_STRUCTURE\_TYPE\_TRANSFORM\_FEEDBACK\_SHADER\_TARGET\_SPIRV\_BINARY  
     vkd3d\_shader.h, 109  
 VKD3D\_SHADER\_STRUCTURE\_TYPE\_VARYING\_MAP\_INDEX  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_CLIP\_DISTANCE  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_COVERAGE  
     vkd3d\_shader.h, 111  
 VKD3D\_SHADER\_SV\_CULL\_DISTANCE  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_DEPTH  
     vkd3d\_shader.h, 111  
 VKD3D\_SHADER\_SV\_DEPTH\_GREATER\_EQUAL  
     vkd3d\_shader.h, 111  
 VKD3D\_SHADER\_SV\_DEPTH\_LESS\_EQUAL  
     vkd3d\_shader.h, 111  
 VKD3D\_SHADER\_SV\_INSTANCE\_ID  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_IS\_FRONT\_FACE  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_NONE  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_POSITION  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_PRIMITIVE\_ID  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_RENDER\_TARGET\_ARRAY\_INDEX  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_SAMPLE\_INDEX  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_STENCIL\_REF  
     vkd3d\_shader.h, 111  
 VKD3D\_SHADER\_SV\_TARGET  
     vkd3d\_shader.h, 111  
 VKD3D\_SHADER\_SV\_VERTEX\_ID  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SV\_VIEWPORT\_ARRAY\_INDEX  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_SWIZZLE  
     vkd3d\_shader.h, 84  
 VKD3D\_SHADER\_SWIZZLE\_MASK  
     vkd3d\_shader.h, 84  
 vkd3d\_shader\_sysval\_semantic  
     vkd3d\_shader.h, 110  
 VKD3D\_SHADER\_TARGET\_D3D\_ASM  
     vkd3d\_shader.h, 111  
 VKD3D\_SHADER\_TARGET\_D3D\_BYTECODE  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_TARGET\_DXBC\_TPF  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_TARGET\_FX  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_TARGET\_GLSL  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_TARGET\_MSL  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_TARGET\_NONE  
     vkd3d\_shader.h, 111  
 VKD3D\_SHADER\_TARGET\_SHADER\_TARGET\_SPIRV\_BINARY  
     vkd3d\_shader.h, 111  
 vkd3d\_shader\_target\_type  
     vkd3d\_shader.h, 111  
 vkd3d\_shader\_transform\_feedback\_element, 56  
 vkd3d\_shader\_transform\_feedback\_info, 57  
 vkd3d\_shader\_uav\_counter\_binding, 57  
     register\_space, 58  
 vkd3d\_shader\_varying\_map, 59  
     output\_signature\_index, 59  
 vkd3d\_shader\_varying\_map\_info, 60  
     varying\_map, 61  
 vkd3d\_shader\_versioned\_root\_signature\_desc, 61  
 vkd3d\_shader\_visibility  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_VISIBILITY\_ALL  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_VISIBILITY\_COMPUTE  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_VISIBILITY\_DOMAIN  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_VISIBILITY\_GEOMETRY  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_VISIBILITY\_HULL  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_VISIBILITY\_PIXEL  
     vkd3d\_shader.h, 112  
 VKD3D\_SHADER\_VISIBILITY\_VERTEX  
     vkd3d\_shader.h, 112  
 vkd3d\_structure\_type  
     vkd3d.h, 67  
 VKD3D\_STRUCTURE\_TYPE\_APPLICATION\_INFO  
     vkd3d.h, 68  
 VKD3D\_STRUCTURE\_TYPE\_DEVICE\_CREATE\_INFO  
     vkd3d.h, 67  
 VKD3D\_STRUCTURE\_TYPE\_HOST\_TIME\_DOMAIN\_INFO  
     vkd3d.h, 68  
 VKD3D\_STRUCTURE\_TYPE\_IMAGE\_RESOURCE\_CREATE\_INFO  
     vkd3d.h, 67  
 VKD3D\_STRUCTURE\_TYPE\_INSTANCE\_CREATE\_INFO  
     vkd3d.h, 67  
 VKD3D\_STRUCTURE\_TYPE\_OPTIONAL\_DEVICE\_EXTENSIONS\_INFO  
     vkd3d.h, 67  
 VKD3D\_STRUCTURE\_TYPE\_OPTIONAL\_INSTANCE\_EXTENSIONS\_INFO  
     vkd3d.h, 67  
 vkd3d\_types.h  
     VKD3D\_ERROR, 142  
     VKD3D\_ERROR\_INVALID\_ARGUMENT, 142  
     VKD3D\_ERROR\_INVALID\_SHADER, 142  
     VKD3D\_ERROR\_KEY\_ALREADY\_EXISTS, 142  
     VKD3D\_ERROR\_MORE\_DATA, 142  
     VKD3D\_ERROR\_NOT\_FOUND, 142  
     VKD3D\_ERROR\_NOT\_IMPLEMENTED, 142  
     VKD3D\_ERROR\_OUT\_OF\_MEMORY, 142  
     VKD3D\_FALSE, 142  
     VKD3D\_OK, 142  
     vkd3d\_result, 142  
 vkd3d\_utils.h

- D3DCompile2, [145](#)
- D3DCompile2VKD3D, [146](#)
- D3DDisassemble, [147](#)
- D3DGetBlobPart, [147](#)
- D3DGetDebugInfo, [147](#)
- D3DGetInputAndOutputSignatureBlob, [147](#)
- D3DGetInputSignatureBlob, [148](#)
- D3DGetOutputSignatureBlob, [148](#)
- D3DReflect, [148](#)
- D3DStripShader, [148](#)
- vkd3d\_utils\_set\_log\_callback, [149](#)
- vkd3d\_utils\_set\_log\_callback
  - vkd3d\_utils.h, [149](#)
- wchar\_size
  - vkd3d\_instance\_create\_info, [15](#)