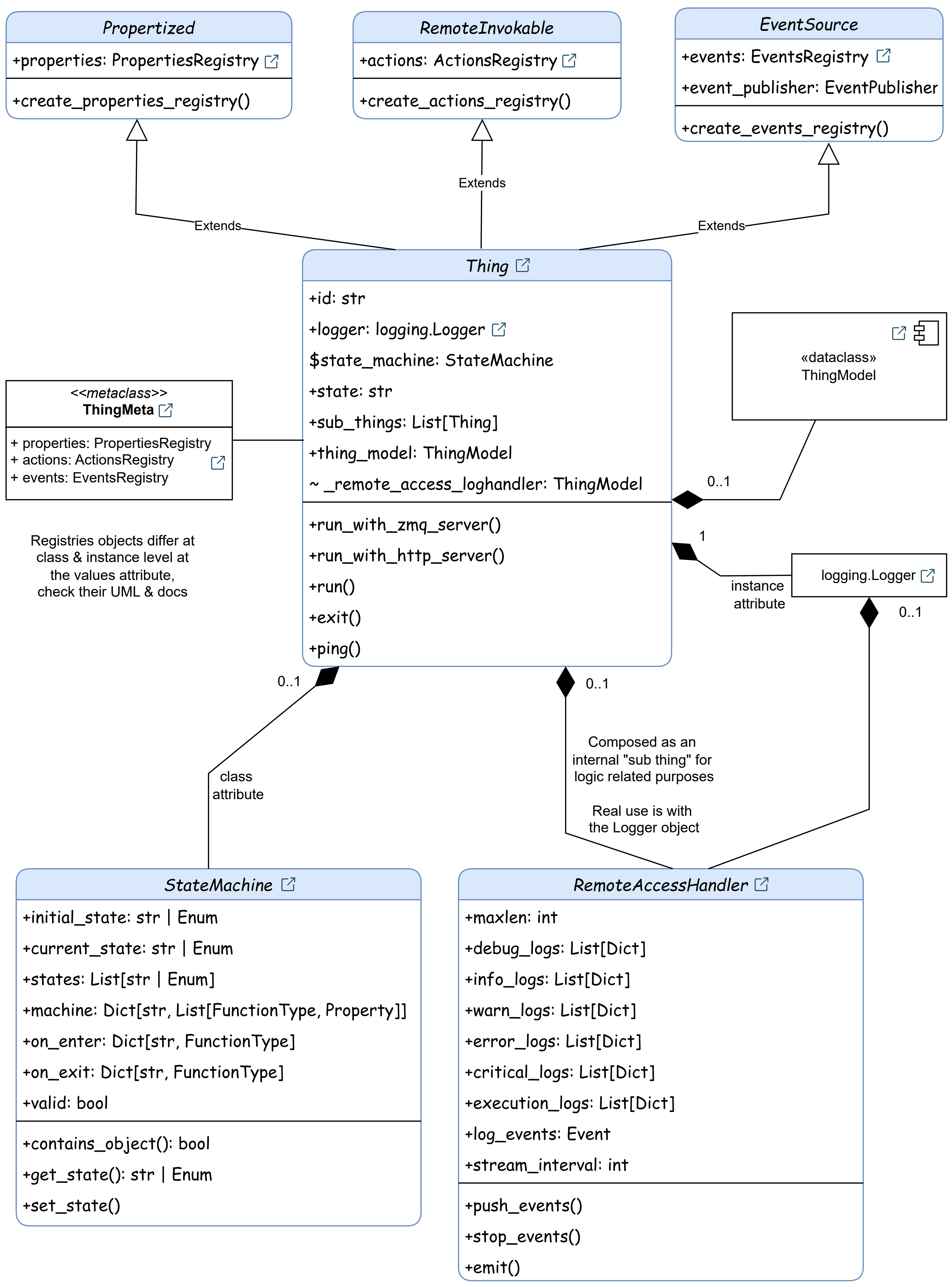
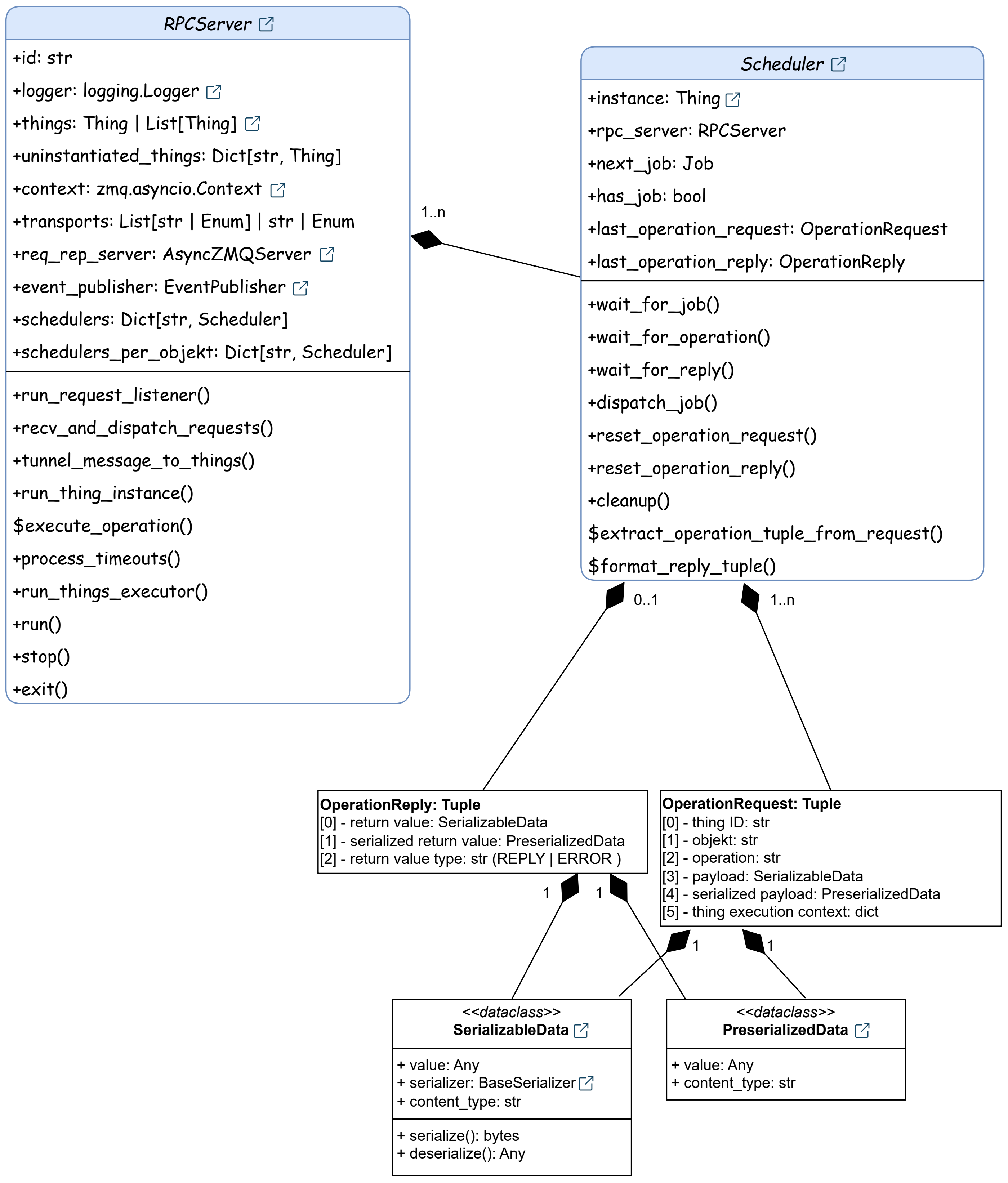
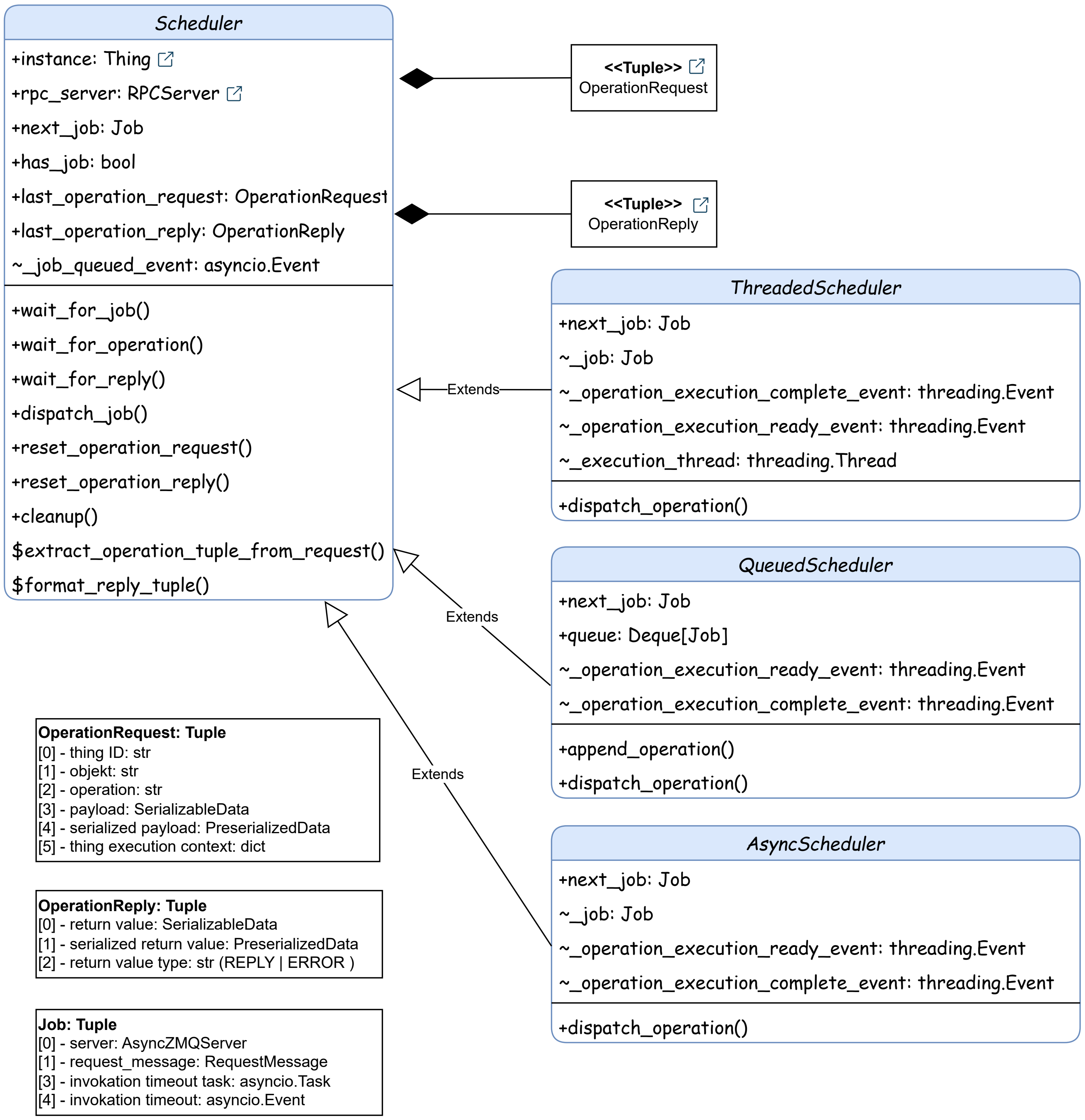
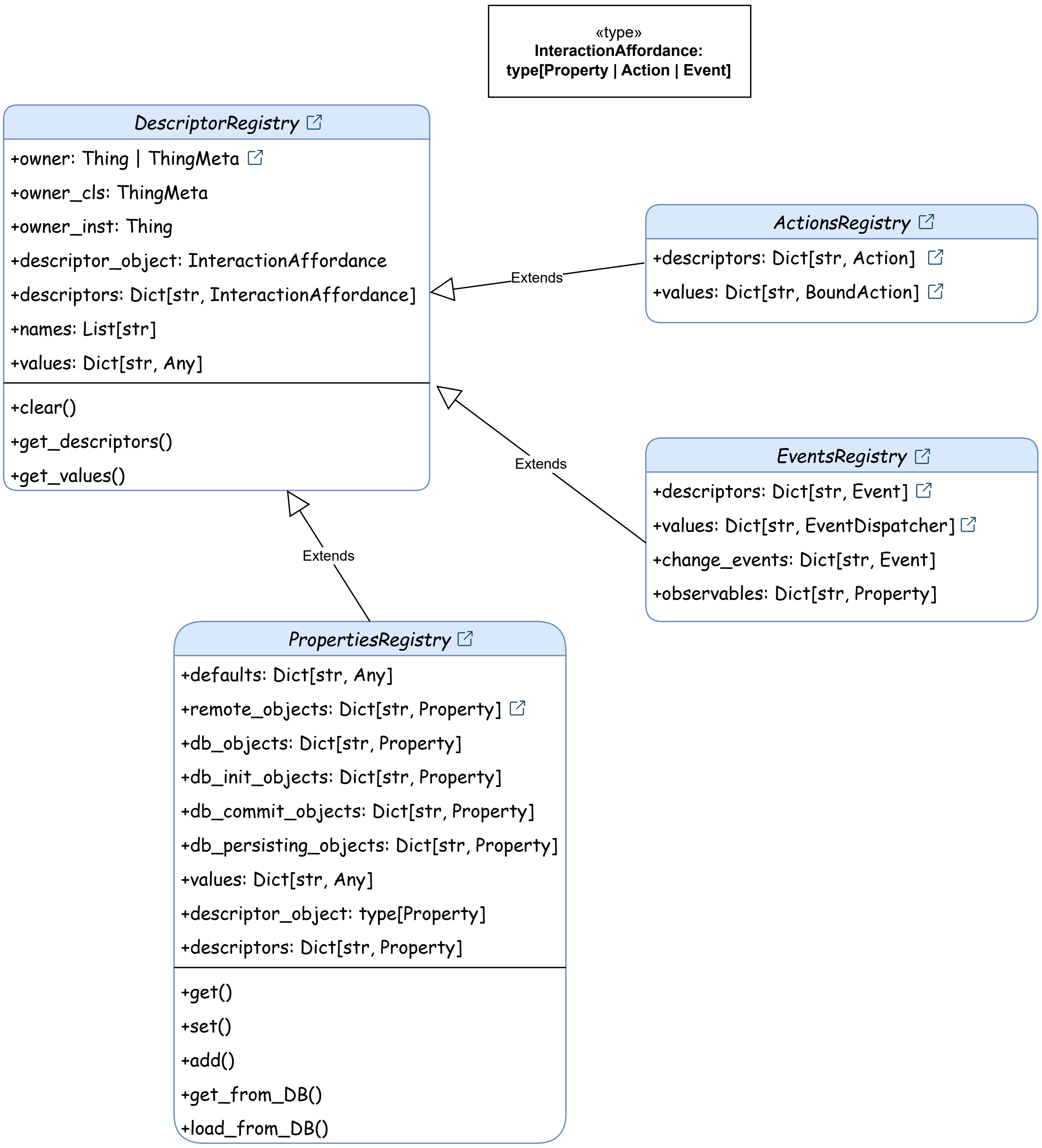


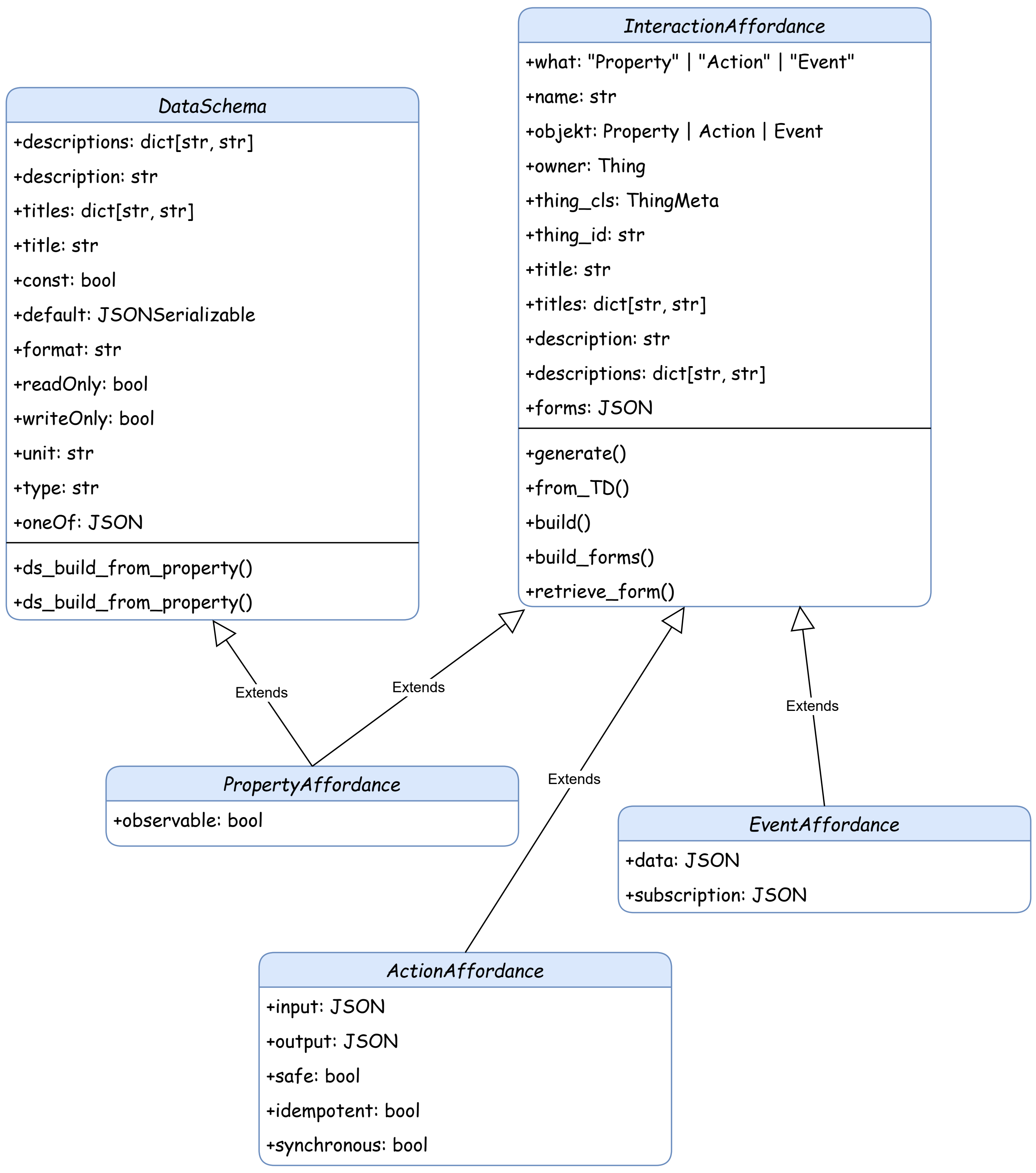
Propertized
+properties: PropertiesRegistry
+create_properties_registry()
RemoteInvokable
+actions: ActionsRegistry
+create_actions_registry()
EventSource
+events: EventsRegistry
+event_publisher: EventPublisher
+create_events_registry()
Extends
Extends
Extends
Thing
+id: str
+logger: logging.Logger
$state_machine: StateMachine
+state: str
+sub_things: List[Thing]
+thing_model: ThingModel
~ _remote_access_loghandler: ThingModel
+run_with_zmq_server()
+run_with_http_server()
+run()
+exit()
+ping()
<<metaclass>>
ThingMeta
+ properties: PropertiesRegistry
+ actions: ActionsRegistry
+ events: EventsRegistry
«dataclass»
ThingModel
0..1
1
instance attribute
logging.Logger
0..1
Registries objects differ at class & instance level at the values attribute, check their UML & docs
0..1
class attribute
0..1
Composed as an internal "sub thing" for logic related purposes
Real use is with the Logger object
StateMachine
+initial_state: str | Enum
+current_state: str | Enum
+states: List[str | Enum]
+machine: Dict[str, List[FunctionType, Property]]
+on_enter: Dict[str, FunctionType]
+on_exit: Dict[str, FunctionType]
+valid: bool
+contains_object(): bool
+get_state(): str | Enum
+set_state()
RemoteAccessHandler
+maxlen: int
+debug_logs: List[Dict]
+info_logs: List[Dict]
+warn_logs: List[Dict]
+error_logs: List[Dict]
+critical_logs: List[Dict]
+execution_logs: List[Dict]
+log_events: Event
+stream_interval: int
+push_events()
+stop_events()
+emit()



RPCServer
+id: str
+logger: logging.Logger
+things: Thing | List[Thing]
+uninstantiated_things: Dict[str, Thing]
+context: zmq.asyncio.Context
+transports: List[str | Enum] | str | Enum
+req_rep_server: AsyncZMQServer
+event_publisher: EventPublisher
+schedulers: Dict[str, Scheduler]
+schedulers_per_objekt: Dict[str, Scheduler]
+run_request_listener()
+recv_and_dispatch_requests()
+tunnel_message_to_things()
+run_thing_instance()
$execute_operation()
+process_timeouts()
+run_things_executor()
+run()
+stop()
+exit()
1..n
Scheduler
+instance: Thing
+rpc_server: RPCServer
+next_job: Job
+has_job: bool
+last_operation_request: OperationRequest
+last_operation_reply: OperationReply
+wait_for_job()
+wait_for_operation()
+wait_for_reply()
+dispatch_job()
+reset_operation_request()
+reset_operation_reply()
+cleanup()
$extract_operation_tuple_from_request()
$format_reply_tuple()
0..1
1..n
OperationReply: Tuple
[0] - return value: SerializableData
[1] - serialized return value: PreserializedData
[2] - return value type: str (REPLY | ERROR)
OperationRequest: Tuple
[0] - thing ID: str
[1] - objekt: str
[2] - operation: str
[3] - payload: SerializableData
[4] - serialized payload: PreserializedData
[5] - thing execution context: dict
1
1
1
1
<<dataclass>>
SerializableData
+ value: Any
+ serializer: BaseSerializer
+ content_type: str
+ serialize(): bytes
+ deserialize(): Any
<<dataclass>>
PreserializedData
+ value: Any
+ content_type: str


Scheduler
+instance: Thing
+rpc_server: RPCServer
+next_job: Job
+has_job: bool
+last_operation_request: OperationRequest
+last_operation_reply: OperationReply
~_job_queued_event: asyncio.Event
+wait_for_job()
+wait_for_operation()
+wait_for_reply()
+dispatch_job()
+reset_operation_request()
+reset_operation_reply()
+cleanup()
$extract_operation_tuple_from_request()
$format_reply_tuple()
<<Tuple>>
OperationRequest
<<Tuple>>
OperationReply
ThreadedScheduler
+next_job: Job
~_job: Job
~_operation_execution_complete_event: threading.Event
~_operation_execution_ready_event: threading.Event
~_execution_thread: threading.Thread
+dispatch_operation()
Extends
QueuedScheduler
+next_job: Job
+queue: Deque[Job]
~_operation_execution_ready_event: threading.Event
~_operation_execution_complete_event: threading.Event
+append_operation()
+dispatch_operation()
Extends
AsyncScheduler
+next_job: Job
~_job: Job
~_operation_execution_ready_event: threading.Event
~_operation_execution_complete_event: threading.Event
+dispatch_operation()
Extends
OperationRequest: Tuple
[0] - thing ID: str
[1] - objekt: str
[2] - operation: str
[3] - payload: SerializableData
[4] - serialized payload: PreserializedData
[5] - thing execution context: dict
OperationReply: Tuple
[0] - return value: SerializableData
[1] - serialized return value: PreserializedData
[2] - return value type: str (REPLY | ERROR)
Job: Tuple
[0] - server: AsyncZMQServer
[1] - request_message: RequestMessage
[3] - invokation timeout task: asyncio.Task
[4] - invokation timeout: asyncio.Event


«type»
InteractionAffordance:
type[Property | Action | Event]
DescriptorRegistry
+owner: Thing | ThingMeta
+owner_cls: ThingMeta
+owner_inst: Thing
+descriptor_object: InteractionAffordance
+descriptors: Dict[str, InteractionAffordance]
+names: List[str]
+values: Dict[str, Any]
+clear()
+get_descriptors()
+get_values()
ActionsRegistry
+descriptors: Dict[str, Action]
+values: Dict[str, BoundAction]
Extends
EventsRegistry
+descriptors: Dict[str, Event]
+values: Dict[str, EventDispatcher]
+change_events: Dict[str, Event]
+observables: Dict[str, Property]
Extends
Extends
PropertiesRegistry
+defaults: Dict[str, Any]
+remote_objects: Dict[str, Property]
+db_objects: Dict[str, Property]
+db_init_objects: Dict[str, Property]
+db_commit_objects: Dict[str, Property]
+db_persisting_objects: Dict[str, Property]
+values: Dict[str, Any]
+descriptor_object: type[Property]
+descriptors: Dict[str, Property]
+get()
+set()
+add()
+get_from_DB()
+load_from_DB()



DataSchema
+descriptions: dict[str, str]
+description: str
+titles: dict[str, str]
+title: str
+const: bool
+default: JSONSerializable
+format: str
+readOnly: bool
+writeOnly: bool
+unit: str
+type: str
+oneOf: JSON
+ds_build_from_property()
+ds_build_from_property()
InteractionAffordance
+what: "Property" | "Action" | "Event"
+name: str
+objekt: Property | Action | Event
+owner: Thing
+thing_cls: ThingMeta
+thing_id: str
+title: str
+titles: dict[str, str]
+description: str
+descriptions: dict[str, str]
+forms: JSON
+generate()
+from_TD()
+build()
+build_forms()
+retrieve_form()
Extends
Extends
Extends
Extends
Extends
PropertyAffordance
+observable: bool
EventAffordance
+data: JSON
+subscription: JSON
ActionAffordance
+input: JSON
+output: JSON
+safe: bool
+idempotent: bool
+synchronous: bool