**Compliance with IEEE Standards Policies and Procedures**

**Subclause 5.2.1 of the *IEEE-SA Standards Board Bylaws* states, "While participating in IEEE standards development activities, all participants...shall act in accordance with all applicable laws (nation-based and international), the IEEE Code of Ethics, and with IEEE Standards policies and procedures."**

The contributor acknowledges and accepts that this contribution is subject to

* The IEEE Standards copyright policy as stated in the *IEEE-SA Standards Board Bylaws*, section 7, [http://standards.ieee.org/develop/policies/bylaws/sect6-7.html#7](http://standards.ieee.org/develop/policies/bylaws/sect6-7.html), and the *IEEE-SA Standards Board Operations Manual*, section 6.1, http://standards.ieee.org/develop/policies/opman/sect6.html
* The IEEE Standards patent policy as stated in the *IEEE-SA Standards Board Bylaws*, section 6, [http://standards.ieee.org/guides/bylaws/sect6-7.html#6](http://standards.ieee.org/guides/bylaws/sect6-7.html), and the *IEEE-SA Standards Board Operations Manual*, section 6.3, http://standards.ieee.org/develop/policies/opman/sect6.html

**IEEE P802.15 Working Group  
Wireless Personal Area Network (WPAN) Working Group**

**Robert Heile**

**bheile@ieee.org**

|  |  |  |
| --- | --- | --- |
| Spectrum Characterization and Occupancy Sensing (SCOS) System | | |
| **Date:** 2020-03-03 | | |
| **Author(s):** | | |
| Name | Affiliation | E-mail (Optional) |
| Doug Boulware | NTIA/ITS | dboulware@ntia.gov |
| Mike Cotton | NTIA/ITS | mcotton@ntia.gov |
| Apurva Mody | A5 Systems | apurva.mody@a5systems.com |
| Oliver Holland | Advanced Wireless Technology Group, Ltd. | oliver.holland@awtg.co.uk |
| Gianfranco Miele | University of Cassino and Southern Lazio | g.miele@unicas.it |

Spectrum Characterization and Occupancy Sensing (SCOS) System

1. Overview
   1. Scope

The scope of this document is to describe the NTIA Spectrum Characterization and Occupancy Sensing (SCOS) system. It establishes a high-level architecture, defines functional entities and their interfaces, specifies command and control messages and parameters, and defines metadata describing the nature and configuration of the sensing system and the data it gathers. It also describes operating characteristics and behaviors of the components of the SCOS system, with reference implementations for common use cases. This standard is intended to enable specialization in the following three areas: sensor technology, data acquisition/distribution, and data analysis.

* 1. Purpose

The purpose of the SCOS system is to characterize and assess the occupancy of spectrum resource towards supporting it’s more efficient and effective use. The intent of the SCOS system is to create a high-level architecture to support different spectrum sensing technologies and deployments, to enable specialization and provide incentive, to promote broad adoption of sensing technologies and subsequent economies of scale, and to ultimately achieve broader availability and usage of sensing information from different sources. This will enable clients to acquire and use spectrum sensing information from a multiplicity of predefined independent systems to serve their goals.

Various national regulators and government authorities are developing regulatory and policy frameworks to allow cooperative spectrum sharing approaches in order to optimize spectrum utilization. There is emphasis on greater spectrum efficiencies, spectrum sharing and spectrum utilization, which require systems that can provide spectrum occupancy at a particular location and at a particular time.

More broadly, the Spectrum Characterization and Occupancy Sensing (SCOS) system has many applications which include:

1. Policy and planning
2. Radio planning, management and engineering
3. Regulatory enforcement where systems can detect (RF incursion), locate (source), classify (by type and severity), resolve/remediate
4. Research and technology development
5. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 1003.1-1988, IEEE Standard for Information Technology - Portable Operating System Interface (POSIX)

IETF RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, June 1999.

IETF RFC 8259, The JavaScript Object Notation (JSON) Data Interchange Format, December 2017.

ISO 8601-1:2019. ISO 8601-2:2019

The GNU Radio Foundation, Inc., “The Signal Metadata Format (SigMF)”, v0.0.2, July 2018[[1]](#footnote-1)

1. Definitions

For the purposes of this document, the following terms and definitions apply. The IEEE Standards Dictionary Online should be consulted for terms not defined in this clause. [[2]](#footnote-2)

**Action:** Sensor function that the sensor developer implements and exposes to the API. Actions give the sensor owner control over what the sensor can be tasked to do, e.g., instrument control, data acquisition, data processing.

**Acquisition:** The combination of data and metadata created by an action (though an action does not have to create data).

**Antenna:** Required Sensor hardware components that convert environmental electromagnetic fields into a voltage. An RF cable connects the antenna with the next hardware component.

**API:** Application Programming Interface.

**Association:** The process by which a SCOS entities join the SCOS system.

**Capability:** Available actions, installation specifications (e.g., mobile or stationary), and hardware specification (e.g., frequency range of receiver).

**Data Client:** An external application that may utilize the services exposed from the Manager. For example, a Data Client could be an application that provides long term storage, analysis, and visualization of sensor data.

**GUI:** Graphical User Interface

**Computer:** Required Sensor hardware components that can provide control signals and messages to the preselector and receiver. Computers may also provide data processing, data packaging, and web-accessible services

**HTTP:** Hypertext Transfer Protocol (IETF RFC 2616)

**JSON:** JavaScript Object Notation is a lightweight data interchange format (IETF RFC 8259).

**Manager:** SCOS entity that allows Users and Data Clients to interact with a distributed network of Sensors via web-based Graphical User Interface (GUI) and/or remotely accessible API. It exposes the capabilities of the SCOS system to Users and Data Clients, manages schedule requests by users and Data Clients, provides control over the network of Sensors, and distributes data to Data Clients.

**Message:** a discrete unit of communication utilized to transfer information or invoke a behavior.

**Object:** a data abstraction used to encapsulate related information and behaviors.

**Preselector:** Optional Sensor hardware components that can provide preselection filtering, improved sensitivity via low noise amplification, and calibration signal sources.

**Property:** data with a type and name that is encapsulated within an Object.

**Protocol:** a standard set of rules that allow electronic devices to communicate with each other.

**Schedule:** The collection of all schedule entries (active and inactive) on the sensor.

**Scheduler:** A Sensor process responsible for executing the schedule.

**Schedule Entry:** Input to the sensor schedule that describes action, start and stop times, interval, and priority.

**SCOS:** Spectrum Characterization and Occupancy Sensing

**Sensor:** SCOS entity that provides distributed RF sensing through a remotely accessible API. The API allows Manager Users to discover capabilities, schedule actions, and download data. Sensors package data with SigMF metadata in a SigMF TAR archive.

**SDR:** Software Defined Radio

**SigMF:** Signal Metadata Format (SigMF) is a specification to describe sets of recorded digital signal samples with metadata written in JSON.

**Signal Analyzer:** Required Sensor hardware components, e.g., Software Defined Radios (SDRs), which capture discrete raw data (e.g., baseband representation of the signal) and can apply digital signal processing algorithms to the raw data to achieve a desired metric.

**SLA:** Service Level Agreement

**Subscription:** The process by which Data Clients specify what Acquisitions they will receive from Manager.

**TAR:** a software utility for collecting many files into one archive file.

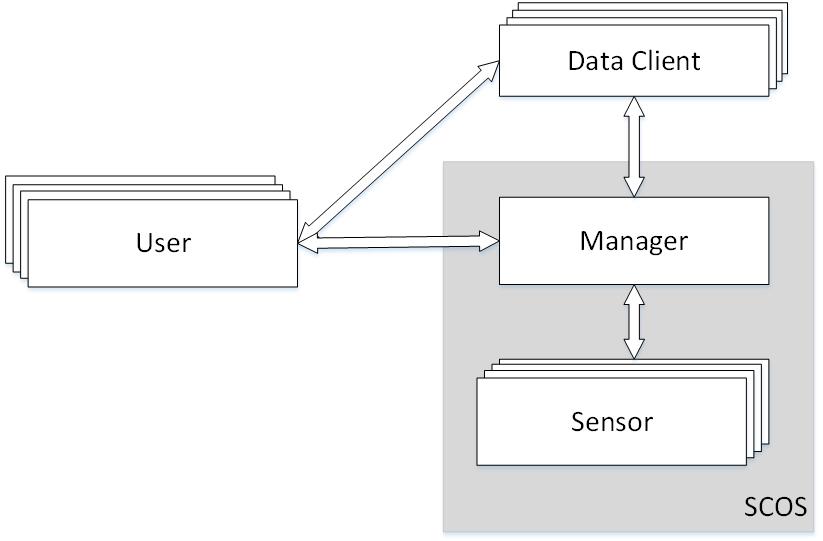
**Task:** An action that was or will be executed at a specific time as part of a Schedule Entry

**Task Result:** A record of the outcome of a task.

**User:** An individual using a Data Client, Manager.

1. System Concept
   1. Context

Figure 1 depicts the boundaries, entities, and high level interactions of the SCOS system. Users and/or Data Clients interact with the system through the Manager to access a distributed network of one or more Sensors to perform RF sensing.



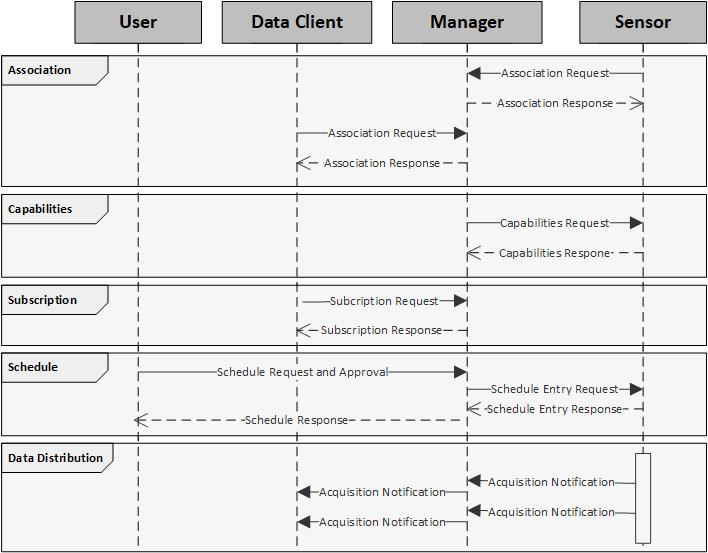
1. —SCOS Context Diagram
   1. Entities

The two entities that comprise the SCOS system are the Manager (server application) and one or more Sensor(s) operating in the field. The Manager API also allows third party applications, referred to as Data Clients, to integrate with the SCOS system.

* **Manager:** SCOS entity that allows Users and Data Clients to interact with a distributed network of Sensors via web-based Graphical User Interface (GUI) and/or remotely accessible API. It exposes the capabilities of the SCOS system to Users and Data Clients, manages schedule requests by Users and Data Clients, provides control over the network of Sensors, and distributes data to Data Clients.
* **Sensor:** SCOS entity that provides distributed RF sensing through a remotely accessible API. The API allows Manager Users to discover capabilities, schedule actions, and download data. Sensors package data with SigMF metadata (The GNU Radio Foundation, Inc) in a SigMF TAR archive (IEEE Std 1003.1-1988).
  1. Interactions

Figure 2 describes the basic interactions between a User, Data Client, Manager, and Sensor. The SCOS system supports the following interactions:

* **Association:** Sensors may associate themselves with a Manager by sending an association request (defined in Table 1), but that does not preclude users from registering a Sensor manually within the Manager. Similarly, a Data Client may associate itself with the Manager with a Data Client association request (defined in Table 15), but that does not preclude a user from manually registering a Data Client within the Manager. The Manager replies to an association request with an association response (defined in Table 2 and Table 16) indicating the status of the request.
* **Capabilities:** Upon registration, or user action, the Manager may request a description of the Sensor’s capabilities (Table 5). Data Clients may also send a capabilities request to the Manager to discover a Sensor’s capabilities. A capabilities response message (Table 6) may be returned from the Sensor to the Manager and from the Manager to a Data Client.
* **Subscription:** A Data Client may send a subscription request (defined in Table 19) to the Manager, but depending upon the implementation a user may configure a Data Client subscription within the Manager as well.
* **Schedule:** After a Sensor has been associated with a Manager, a user may use the Manager to define a schedule request. Different implementations may choose to introduce manual approval processes, but any approval processes are implementation specific and beyond the scope of the standard. After the user specifies the schedule parameters, the Manager sends the request (defined in Table 7) to the Sensor. In response, the Sensor will notify the Manager if the ScheduleEntry was accepted with a ScheduleEntry response (defined in Table 8). If the Sensor accepts the schedule request, the Sensor will begin executing the Action as defined in the schedule entry. Schedule entry requests may also be sent from a Data Client to a Manager.
* **Data Distribution:** Each time the Sensor executes the Action it will result in an Acquisition that may contain sensed data and the Sensor sends the Manager an Acquisition notification (defined in Table 13). If the Manager has Data Clients that have subscribed to receive the Acquisition it will send the Data Clients an Acquisition notification. In addition, the Manager may download archives from Sensors, and provide Users and Data Clients the ability to download archives.



1. —System Interactions
   1. General Requirements

The primary goals of the SCOS architecture are to achieve distributed persistent sensing and data aggregation. To meet these goals, SCOS developers are required to meet general requirements categorized and described in the following subsections.

* + 1. Connectivity, Control, and Access

The following requirements are related to connectivity, sensor control, and data access.

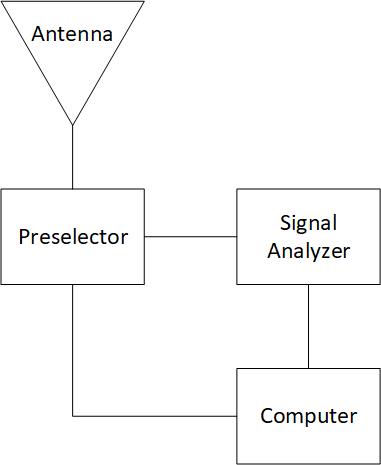
* **Connectivity and access:** Sensors and Manager shall be remotely accessible to authorized users over a network.
* **Control:** SCOS entities shall provide interfaces for distributed control and data acquisition.
* **Discoverable capabilities:** SCOS capabilities and resources shall be discoverable.
* **Data access:** SCOS entities shall make data available to authorized users.
* **Data standardization:** Compliance with a common metadata/data specification is required to allow for collaborative research, large-scale analytics, and sharing of sophisticated tools and methodologies. Sensor data acquisitions shall include shall include SigMF metadata and should use the SigMF extensions defined in Annex A.
  + 1. Design Flexibility

The following requirements support the evolution of SCOS technology and broad deployments.

* **Extensible technologies, algorithms, and metrics:** The SCOS architecture shall support the evolution of centralized and edge processing, sensor technologies, and metrics.
* **Hardware, software, OS, and protocol agnostic:** SCOS standard implementations shall not be bound to specific hardware, software, operating systems or protocols to allow for cost-performance tradeoffs to be considered during the design process and for implementations to be tailored to address unique domain requirements.

1. Sensor
   1. Hardware

Figure 3 provides a block diagram of the basic Sensor hardware model. There may be more sophisticated hardware models, e.g., with multiple antennas and/or multiple signal analyzers connected to a single computer. Each of the components may also be integrated within a single unit (e.g., a mobile phone). Metadata to describe the hardware components is defined in Annex A.



1. —Sensor Hardware Model

* **Antenna:** Required Sensor hardware components that convert environmental electromagnetic fields into a voltage. An RF cable may connect the antenna with the next hardware component.
* **Preselector:** Optional Sensor hardware components that can provide preselection filtering, improved sensitivity via low noise amplification, and calibration signal sources.
* **Signal Analyzer:** Required Sensor hardware components, e.g., Software Defined Radios (SDRs), which capture discrete raw data (e.g., baseband representation of the signal) and can apply digital signal processing algorithms to the raw data to achieve a desired metric.
* **Computer:** Required Sensor hardware components that can provide control signals and messages to the preselector and receiver. Computers may also provide data processing, data packaging, and remotely accessible services.
  1. Software
     1. Functional Requirements

The following are functional requirements of the Sensor software in order to provide a common language for sensor control and data acquisition. Sensors shall:

* Associate with a Manger using the association request defined Table 1.
* Describe Sensor Status, e.g., location, system datetime, last calibration datetime, and execution status as defined in Table 4.
* Advertise Sensor Capabilities, e.g., sensor hardware configuration and available Actions, as defined in Table 6. Actions are functions that the sensor developer implements and exposes to the API. Actions should be used to constrain Sensor tasks to the valid operational ranges of hardware components (e.g., frequency range of antenna, preselector, and signal analyzer).
* Execute Schedule Entries, defined in Table 7, that specify Action, start/stop times, interval, and/or priority.
* Describe schedule and task status as defined in Table 12 in response to the status request defined in Table 11.
* Record and distribute Acquisitions including metadata describing the measurement and security categorization of the data.
  + 1. Messages

The following subsections describe the sensor messages used to provide the core functionality across five key system interactions: association, capabilities, subscription, schedule, and data distribution. Each of the message descriptions below include a column titled R/O/C that indicates whether each property is Required (R), Optional (O), or Conditional (C). Required properties shall be included in the message. Optional properties may be included in the message, and Conditional properties shall be included in the message under specified conditions. In addition, many of the messages below utilize Conditional properties and indicate that the properties shall be used when the underlying protocol does not inherently define the property. The SCOS system remains protocol agnostic and in some protocols it makes more sense to rely on the built in features of the protocol than to rely on custom properties within the messages themselves. For example, HTTP defines verbs that define common operations like create, read, update, and delete and also includes status codes that are useful to indicate common status conditions.

* + - 1. Association

Association messages allow Sensors to register with a Manager. Table 1 and Table 2 describe the association request and response messages.

1. — Sensor Association Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The id of the Manager with which the sensor is registering. The manager\_id shall be included when it is not inherently defined by the underlying protocols. |
| sensor\_id | R | string | The unique id of the sensor. |
| owner | O | string | The id of the sensor owner. |
| name | R | string | The name of the sensor. |
| sensor\_type | O | string | The type of the sensor. When used, the sensor\_type shall equal either sensor or proxy.If the sensor\_type is not included it shall be assumed that it is of type sensor. |
| protocol | O | string | The protocol to use to communicate with the sensor. |
| message\_type | C | string | The type of the request. The message\_type shall be included when it is not inherently defined by the underlying protocols. The message\_type shall equal add\_sensor when used to add a sensor to a Manager, or remove\_sensor when removing a sensor from a Manager. |

1. —Sensor Association Response

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique id of the Manager. The manager\_id shall be included when it is not inherently defined by the underlying protocols. |
| sensor\_id | C | string | The unique id of the sensor. The sensor\_id shall be included when it is not inherently defined by the underlying protocols. |
| status | C | integer | The status response code. The status shall be included when it is not inherently defined by the underlying protocols. |
| detail | O | string | A message providing any extra details that explain the status. |
| message\_type | C | string | The type of the message. The message\_type shall be included when it is not inherently defined by the underlying protocols. The message\_type shall equal sensor\_association\_response when used in the Sensor association response. |

* + - 1. Sensor Status

Status messages support the discovery of Sensor status. Managers request status from a Sensor and Data Clients may request Sensor status from the Manager. Table 3 and Table 4 describe the status request and response messages.

1. —Status Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the Manager. The manager\_id may function as the source or the destination of the message. The manager\_id field shall be included when the underlying transfer protocol does not inherently define it. |
| sensor\_id | C | string | The unique ID of the sensor. The sensor\_id shall be included when the underlying transfer protocol does not inherently define it. |
| client\_id | C | string | The unique ID of the Data Client making the request. The client\_id shall be included when the underlying protocols do not inherently define it and the request originates from a Data Client. |
| message\_type | C | string | The type of the request (status, capabilities…). The message\_type is required when the underlying communication protocols do not inherently define the message type. The message\_type shall equal status\_request for status request messages. |

1. —Status Response

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the Manager making the request. The manager\_id is required when it is not inherently defined by the underlying communication protocols and the status request originated from a Manager. |
| sensor\_id | C | string | The unique ID of the sensor. The sensor\_id is required when is it not inherently defined by the underlying communication protocols. |
| client\_id | C | string | The unique ID of the Data Client making the request. The client\_id shall be included when it is not inherently defined by the underlying communication protocols and the response is being returned to a Data Client. |
| scheduler | O | string | The scheduler status (idle, running, dead) |
| location | O | Location (see Table 21) | Sensor location information. |
| system\_time | R | datetime | The current system time on the sensor |
| calibration\_datetime | O | datetime | The last date/time that the sensor was calibrated. |
| message\_type | C | string | The message type. The message\_type is required when the message type is not inherently defined by the underlying communication protocols. The message\_type shall equal status for status response messages. |

* + - 1. Sensor Capabilities

Capabilities messages enable the discovery of a Sensor’s Capabilities. The Capabilities include a description of the physical Sensor as well as the Actions it provides. Capabilities request and response messages are described in Table 5 and Table 6.

1. —Capabilities Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the Manager. The manager\_id may act as the source or the destination of the message. The manager\_id is required when it is not inherently defined by the underlying communication protocols. |
| sensor\_id | C | string | The unique ID of the sensor. The sensor\_id is required when the sensor id is not inherently defined by the underlying communication protocols. |
| client\_id | C | string | The unique ID of the Data Client. The client\_id shall be included when it is not inherently defined by the underlying communication protocols and the request originated from a Data Client. |
| message\_type | C | string | The type of the request. The message\_type is required when it is not inherently defined by the underlying communication protocols. When the message\_type is used within a capabilities request message it shall have a value of capabilities\_request. |

1. —Capabilities Response

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the Manager that requested the capabilities description. The manager\_id is required when it is not inherently defined by the underlying communication protocols. |
| sensor\_id | C | string | The unique ID of the sensor for which the capabilities are being described. The sensor\_id is required when it is not inherently defined by the underlying communication protocols |
| client\_id | C | string | The unique ID of the Data Client that initiated the capabilities request. The client\_id is required when it is not inherently defined by the underlying communication protocols and the response is to a Data Client. |
| sensor | R | Sensor | A description of the sensor. See Table 37. |
| actions | R | Action[] (see Table 22) | An array of Action objects describing each Action the sensor can perform |
| message\_type | C | string | The message type. The message\_type is required when it is not inherently defined by the underlying communication protocols. When the message\_type is used within a capabilities response message it shall have a value of capabilities\_response. |

* + - 1. Schedule

Schedule messages allow Actions to be scheduled on one or more Sensors and support basic Schedule querying. Scheduling should be performed by the Manager, however schedule requests may be sent from Data Clients to a Manager. Schedule entry request messages, defined in Table 7, may be sent from the Manager to one or more Sensors and/or from a Data Client to a Manager. Individual Sensors may accept or reject schedule entry requests and respond with the Schedule entry response message defined in Table 8. Schedule overview request messages allow the Manage or Data Client to request an overview of all schedule entries (past and present). The schedule overview request and response messages are defined in Table 9 and Table 10.

1. —Schedule Entry Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the Manager. The manager\_id shall be included when it is not defined by the underlying protocols. The manager\_id may be the ID of the Manager making a request or receiving a request depending on the context of the message. |
| sensor\_ids | C | string[] | The unique IDs of the sensors. The sensor\_ids shall be included when not defined by the underlying protocols. |
| client\_id | C | string | The unique ID of the Data Client. The client\_id shall be included when it is not defined by the underlying protocols and the request originates from a Data Client. |
| name | R | string | The name for the schedule entry. |
| schedule\_id | R | string | The unique ID for the schedule entry. |
| action | R | string | The name of the action that will be executed |
| start | O | datetime | Requested time to schedule the first task. If unspecified, the task will start as soon as possible |
| stop | O | datetime | Requested time to end execution of tasks under this schedule. If unspecified, and no relative\_stop is specified then one task will execute and then the schedule will become inactive. |
| relative\_stop | O | Integer | Seconds after start when the schedule will end. If unspecified, and no stop is specified, one task will execute and then the schedule will become inactive. |
| interval | O | integer | Seconds between tasks. If unspecified, a task will run once and then the schedule will become inactive |
| is\_active | R | boolean | Indicates if tasks will continue to be executed for the schedule |
| priority | O | integer | Priority of the entry. Lower numbers indicate higher priority |
| validate\_only | O | boolean | If true the input will only be validated and no schedule entry will be created |
| message\_type | C | string | The message\_type is required when it is not inherently defined by the underlying protocols. When used within a schedule entry request message, the message\_type shall equal create\_schedule, update\_schedule, get\_schedule, or delete\_schedule. |

1. — Schedule Entry Response

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the Manager making the request. The manager\_id shall be included when it is not inherently defined by the underlying protocols. |
| sensor\_id | C | string | The unique ID of the sensor. The sensor\_id shall be included when it is not inherently defined by the underlying protocols. |
| client\_id | C | string | The unique ID of the Data Client that performed the schedule entry request. The client\_id shall be included when it not inherently defined by the underlying protocols and the schedule entry request originated from a Data Client. |
| schedule\_id | R | string | The unique ID of the schedule. |
| name | R | string | The name for the schedule |
| action | R | string | The name of the action that will be executed |
| priority | O | integer | Priority of the entry. Lower numbers indicate higher priority |
| start | O | datetime | Requested time to schedule the first task. If unspecified, the task will start as soon as possible |
| stop | O | datetime | Requested time to end execution of tasks under this schedule. |
| Interval | O | integer | Interval between tasks. If unspecified, the task will run once and then become inactive |
| is\_active | R | boolean | Indicates if tasks will continue to be executed for the schedule |
| validate\_only | O | boolean | If true the input will only be validated and no ScheduleEntry will be created |
| next\_task\_time | O | datetime | The time at which the next task will execute. |
| next\_task\_id | O | integer | The id of the next task. |
| created | O | datetime | The datetime of the original schedule entry request. |
| modified | O | datetime | The datetime at which the schedule entry was last modified. |
| user\_id | O | string | The id of the user that requested the schedule entry. |
| status | C | string | Indicates if the request was accepted or rejected. The status is required when the underlying communication protocols do not define a request status. |
| message\_type | C | string | The type of the message. The message\_type shall be included when it is not inherently defined by the underlying protocols. When the message\_type is used within a schedule entry response it shall be equal to schedule\_response. |

1. —Schedule Overview Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Required | Type | Description |
| manager\_id | C | string | The unique ID of the sensor Manager making the request. The manager\_id is required when it is not inherently defined by the underlying communication protocols. |
| sensor\_id | C | string | The unique ID of the sensor. The sensor\_id is required when it is not inherently defined by the underlying communication protocols. |
| client\_id | C | string | The unique ID of the Data Client making the request. The client\_id shall be included when it is not defined by the underlying communication protocols and the request originates from a Data Client. |
| offset | O | integer | The schedule index at which to start the list of results. |
| limit | O | integer | The number of schedule entries to return. |
| is\_active | O | boolean | Indicates whether or not to only include active Schedule Entries. |
| message\_type | C | string | The message\_type is required when it is not inherently defined by the underlying communication protocols. Schedule request messages support getting an overview of all schedule entries on the Sensor as well as obtaining information on the upcoming tasks that will be executed on the Sensor. When the message\_type is used to request an overview of all schedule entries it shall equal get\_schedule\_overview. |

1. —Schedule Overview Response

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| count | R | integer | The total number of schedule entries on the sensor. |
| results | R | ScheduleEntry[] (see Table 23) | An array of SheduleEntry objects. |
| message\_type | C | string | The type of the message. The message\_type shall be included when it is not inherently defined by the underlying communication protocols. When the message\_type is used within a schedule overview response it shall equal schedule\_overview\_response. |

* + - 1. Task Status

Task status messages support basic querying of task status. A Manager may query a Sensor using a task status request, or a Data Client may query a Manager with a task status request. A task status request may be done at the schedule entry level or at the task level. Task status request and response messages are defined in Table 11 and Table 12.

1. —Task Status Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Required | Type | Description |
| manager\_id | C | string | The unique ID of the sensor Manager making the request. The manager\_id shall be included when it is not inherently defined by the underlying protocols. |
| sensor\_id | C | string | The unique ID of the sensor. The sensor\_id shall be included when it is not inherently defined by the underlying protocols. |
| client\_id | C | string | The unique ID of the Data Client making the request. The client\_id shall be included when it is not defined by the underlying protocols and the request originates from a Data Client. |
| schedule\_id | C | string | The unique id of the schedule. The schedule\_id shall be included when it is not defined by the underlying protocol. |
| task\_id | O | integer | The id of the task in the schedule. The task\_id is an optional parameter and may be specified in an alternative way depending on the underlying protocols. |
| offset | O | integer | The index at which to start the list of results. The offset is an option parameter and it may also be specified in an alternative way depending on the underlying protocols. |
| limit | O | integer | The maximum number or results requested. The limit is an optional parameter and it may be specified in an alternative way depending on the underlying protocols. |
| message\_type | C | string | The message type shall be included when it is not inherently defined by the underlying protocol. When used within a task status request the message\_type shall equal get\_task\_status. |

1. —Task Status Response

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the sensor Manager making the request. The manager\_id shall be included if it not defined by the underlying protocols. |
| sensor\_id | C | string | The unique ID of the sensor. The sensor\_id shall be included if it is not defined by the underlying protocols. |
| client\_id | C | string | The unique ID of the Data Client if the response is in reply to a request that originated from a Data Client. The client\_id shall be include when it is not defined by the underlying protocols and the response is in reply to a request that originated from a Data Client. |
| tasks | R | Task[] (see Table 24) | AcquisitionOverview for each schedule entry dictated by the limit and offset parameters |
| message\_type | C | string | The type of the message. The message\_type shall be included when it is not defined by the underlying communication protocols. When the message\_type is included within an acquisitions overview response it shall equal task\_status\_response. |

* + - 1. Data Distribution

Data distribution messages support the distribution of Acquisitions from a Sensor to a Manager and from a Manager to a Data Client. Sensors send and Acquisition notification message, defined in Table 13, to the Manager upon task completion. The message may or may not contain the measured data regardless of whether or not the task generated any measured data. If the task generated measured data and the data is not included in the Acquisition notification message, an id or locator shall be included in the metadata that allows the Manager or Data Client to retrieve the archive via an archive request.

1. —Acquisition Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the sensor Manager being notified of a new Acquisition. The manager\_id shall be included when it is not defined by the underlying protocols. |
| sensor\_id | C | string | The unique ID of the sensor that generated the acquisition. The sensor\_id shall be included when it is not defined by the underlying protocols. |
| client\_id | C | string | The unique ID of the Data Client being notified of a new Acquisition. The client\_id shall be included when it is not defined by the underlying protocols and the message is destined for a Data Client. |
| task\_id | R | integer | The id of the task that generated the acquisition. |
| started | R | datetime | The datetime stamp for when the task started |
| finished | R | datetime | The datetime stamp for when the task finished |
| duration | O | time | The duration of the action |
| status | R | string | success, fail, in-progress |
| recordings | O | Recording[] (see Table 25) | The recordings generated from the task. |
| schedule\_id | R | string | The ScheduleEntry id |
| detail | O | string | Any additional details regarding the task execution. |
| message\_type | C | string | The message\_type shall be included when it is not inherently defined by the underlying protocols. When the message type is included in an acquisition notification it shall equal acquisition\_notification. |

1. —Archive Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the Manager requesting the archive or the unique ID of the Manager receiving the request. The manager\_id shall be included when it is not defined by the underlying protocols. |
| sensor\_id | C | string | The unique ID of the sensor that generated the acquisition. The sensor\_id shall be included when it is not defined by the underlying protocols. |
| client\_id | C | string | The unique ID of the Data Client requesting the archive. The client\_id shall be included when it is not defined by the underlying protocols and the message is destined for a Data Client. |
| schedule\_entry | C | string | The unique ID of the schedule entry that generated the archive. The schedule\_entry shall be included when it is not defined by the underlying protocols. |
| archive\_id | C | string | The ID of the archive. The archive\_id shall be included when it is not defined by the underlying protocols. |
| message\_type | C | string | The message\_type shall be included if it is not defined by the underlying protocols. When used in an archive request the message\_type shall equal get\_archive. |

1. Manager
   1. Hardware

Manager is hosted on a server or virtual machine. Resources and configuration can be customized to meet design and SLA requirements.

* 1. Software

Manager is a web application/s that simplifies the management and tasking of a network of sensors.

* + 1. Functional Requirements

The following are software requirements for the Manager software:

* **Operations:** Sensor locations, status, and other diagnostics are displayed in operations view to allow for early detection of problems.
* **Association:** Managers shall allow Data Clients and Sensors to associate themselves with the Manager.
* **Distributed sensing actions:** Managers shall support the scheduling of Actions on single or multiple distributed sensors.
* **Subscription:** Managers shall allow Data Clients to subscribe to receive newly acquired Acquisitions based on specified subscription criteria.
* **Data Distribution:** Managers shall notify Data Clients of newly acquired Acquisitions that match specified subscription criteria. In addition, with appropriate permissions, Users and Data Clients may download acquisition archives.
  + 1. Messages

The following subsections describe the messages used by the Manager to provide the core functionality across five key areas: associating Sensors and Data Clients with a Manager, providing Sensor status, providing Sensor capabilities, scheduling actions on a sensor, subscribing Data Clients, and receiving, requesting and distributing Sensor Acquisitions. Each of the message descriptions below include a column titled R/O/C that indicates whether each property is Required (R), Optional (O), or Conditional (C). Required properties shall be included in the message. Optional properties may be included in the message, and Conditional properties shall be included in the message under specified conditions. In addition, many of the messages below utilize Conditional properties and indicate that the properties shall be used when the underlying protocol does not inherently define the property. The SCOS system remains protocol agnostic and in some protocols it makes more sense to rely on the built in features of the protocol than to rely on custom properties within the messages themselves. For example, HTTP defines verbs that define common operations like create, read, update, and delete; and also includes status codes that are useful to indicate common status conditions.

* + - 1. Association

Association messages allow Sensors and Data Clients to associate themselves with a Manager. For sensor association, the Manager consumes the association request message defined in Table 1 and responds with the association response message in Table 2. Data Clients associate with a Manager by sending the association request message in Table 15 and the Manager responds with association response message in Table 16.

1. —Data Client Association Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The id of the sensor Manager with which the sensor is registering. The manager\_id shall be included when it is not inherently defined by the underlying protocols. |
| client\_id | C | string | The unique id of the Data Client. The client\_id shall be included when it is not inherently defined by the underlying protocol. |
| owner | O | string | The id of the Data Client owner. |
| name | R | string | The name of the Data Client. |
| protocol | O | string | The protocol to use to communicate with the Data Client. |
| message\_type | C | string | The type of the request. The message\_type shall be included when it is not inherently defined by the underlying protocols. The message\_type shall equal add\_client when used to add a Data Client to a Manager, or remove\_client when removing a data\_client from a Manager. |

1. —Data Client Association Response

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique id of the Manager. The manager\_id shall be included when it is not inherently defined by the underlying protocols. |
| client\_id | C | string | The unique id of the sensor. The client\_id shall be included when it is not inherently defined by the underlying protocols. |
| status | C | integer | The status response code. The status shall be included when it is not inherently defined by the underlying protocols. |
| message | O | string | A message providing any extra details that explain the status. |

* + - 1. Sensor Status

Status request messages, defined in Table 17, allow Data Clients to identify the sensors that are available from the Manager and to discover the status of individual sensors. The Manager shall return the Sensors response message defined in Table 18 in response to a get\_sensors status request. In addition, the Manager shall return the status response message defined in Table 4 in response to a get\_status status request.

1. —Status Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the sensor Manager making or receiving the request. The manager\_id shall be included when it is not defined by the underlying protocols. |
| sensor\_id | C | string | The unique ID of the sensor for which the status is being requested. The sensor\_id property shall be included in sensor\_status requests when it is not defined by the underlying protocol. |
| client\_id | C | string | The unique ID of the Data Client making the request when the request originates from a Data Client. The client\_id shall be included when it is not defined by the underlying protocols. |
| message\_type | C | string | The type of the request. The message\_type shall be included when it is not inherently defined by the underlying protocols. When the message\_type is used within a Manager status request to identify the sensors that are available the message\_type shall equal get\_sensors. When the message\_type is used within a request to obtain the status of an individual sensor the message\_type shall equal get\_status. |

1. —Sensors Response

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | R | String | The unique ID of the sensor Manager making the request. |
| client\_id | R | String | The unique ID of the Data Client making the request. |
| sensors | R | Sensor[] (see Table 37) | An array of Sensors. |
| message\_type | C | String | The message\_type shall be included when it is not inherently defined by the underlying protocols. When message\_type is used within the sensors response message it shall equal sesnors\_response. |

* + - 1. Sensor Capabilities

Data Clients may send a Manager the capabilities request defined in Table 5 and Manager may send the request to Sensors. Sensors and Managers respond with the capabilities response message defined in Table 6.

* + - 1. Schedule

Data Clients may send a ScheduleEntry request, defined in Table 7, and Manager may send ScheduleEntry requests to Sensors. Sensors and Managers respond with the ScheduleEntry response message defined in Table 8. The ScheduleOverview request and response defined in Table 9 and Table 10 may be used in a similar manner.

* + - 1. Subscription

Subscription messages allow Data Clients to subscribe to receive the acquisitions that result from the tasks performed by sensors. Data Clients may subscribe by Sensor, and or Action as described in Table 19. Subscription response messages are returned from the manger to indicate the status of the subscription request. Once subscriptions have been established, the Data Clients will receive the acquisition notification messages defined in Table 13 for any acquisitions that satisfy the subscription criteria.

1. —Subscription Request

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique ID of the Manager. |
| client\_id | C | string | The unique ID of the Data Client. |
| actions | R | string[] | A list of action names indicating the actions for which the Data Client will receive TaskData messages when executed by a sensor included in the sensors property. The list may have the value any to indicate the Data Client would like to subscribe to any action from the specified sensors. |
| sensor\_ids | R | string[] | A list of sensor ids indicating the sensors from which the Data Client will receive TaskData messages when an action included in the actions property is executed by one of the sensors. The list may include the value any to indicate the Data Client would like to subscribe to the specified actions on any sensor. |

1. —Subscription Response

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| manager\_id | C | string | The unique id of the Manager. The manager\_id shall be included when it is not inherently defined by the underlying protocols. |
| client\_id | C | string | The unique id of the Sensor. The client\_id shall be included when it is not inherently defined by the underlying protocols. |
| status | C | integer | The status code indicating whether the subscription was accepted, or refused. |
| message | O | string | A message explaining the status of the request. |

* + - 1. Data Distribution

The Manager consumes the acquisition notification message defined in Table 13 and sends acquisition notification messages to Data Clients when it has received an acquisition that matches the Data Client’s subscription constraints. The acquisition notification messages use the Recording object defined in Table 25. The Recording defines an optional data field to hold the measured data resulting from the execution of an action. The data field is optional to allow the recipient to utilize an archive request defined in Table 14 to request the data. When the Manager receives an acquisition notification without data and with an archive id\locator in the acquisition notification it performs an archive request. Data Clients may also send a Manager an archive request, defined in Table 14.

1. Metadata and Data Specification
   1. Overview

This section describes the normative metadata used within messages within the SCOS system as well as the SigMF metadata files and datasets produced by a Sensor. To satisfy the requirements of extensibility and data standardization, Sensors shall use the SigMF specification to record and package recorded measurement data and metadata. SigMF specifies that metadata shall be written in JSON and that the entire contents of the metadata shall be contained within a single JSON object that contains three objects named global, captures, and annotations. SigMF allows extension namespaces to define new top-level SigMF objects, name/value pairs, new files, new dataset formats, or new datatypes. SCOS defines normative metadata objects used within control plane messages as well as informative SigMF extensions that should be used within the metadata files. Each of the object descriptions below includes a column titled R/O/C that indicates whether each property is Required (R), Optional (O), or Conditional (C). Required properties shall be included in the object. Optional properties may be included in the object, and Conditional properties shall be included in the object under specified conditions.

* + 1. Message Metadata

This section describes the JSON objects used within the messages in 5.2.2 and 6.2.2. The

* + - 1. Status

This section describes the JSON objects used within status messages.

* + - * 1. Location

The Location object in Table 21 is used within the status response message to convey the location of the Sensor.

1. —Location Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| gps | O | boolean | N/A | Indicates whether the location information was pulled from GPS |
| modified | O | datetime | ISO-8601 (ISO 8601-1:2019) | Time of the last location update |
| description | O | string | N/A | A textual description of the Sensor’s location |
| latitude | O | number | decimal degrees | The Sensor’s latitude |
| longitude | O | number | decimal degrees | The Sensor’s longitude |

* + - * 1. Sensor

The Sensor object, defined in Table 37, provided by the scos-sensor extension in A.3 is used within the Sensors response message defined in Table 18.

* + - 1. Capabilities

This section describes the objects used within the capabilities messages.

* + - * 1. Sensor

The capabilities messages use the Sensor, Preselector, RFPath, and SignalAnalyzer objects provided by the scos-sensor SigMF extension. These objects are defined in Table 37, Table 38, Table 42, and Table 43.

* + - * 1. Action

The Action object is used within the capabilities response message, defined in Table 6, to describe the operations a Sensor may perform. Actions will typically represent RF sensing operations, but may also encapsulate maintenance, administrative, or other utility functionality.

1. —Action Object

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| name | R | string | The name of the action |
| summary | R | string | A short description of the action |
| description | O | string | A detailed description of the action |

* + - 1. Schedule

This section describes the objects used within scheduling messages.

* + - * 1. ScheduleEntry

The ScheduleEntry object is used within the ScheduleOverview response defined in Table 10.

1. —ScheduleEntry

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| id | R | string | The unique id for the ScheduleEntry. |
| name | R | string | The name for the ScheduleEntry. |
| sensor\_ids | C | string[] | The ids of the sensors tasked by the ScheduleEntry. |
| action | R | string | The name of the action that will be executed. |
| start | C | datetime | Requested time to schedule the first task. If unspecified, the task will execute as soon as possible. |
| stop | C | datetime | Requested time to stop scheduling tasks |
| interval | O | integer | Number of seconds between tasks. If unspecified, the task will run exactly once and the schedule will become inactive. |
| is\_active | C | boolean | Indicates if tasks will continue to be executed for the schedule. |
| priority | O | integer | Priority of the entry. Lower numbers indicate higher priority. |
| next\_task\_time | O | datetime | The time at which this task will execute the under this schedule. |
| next\_task\_id | O | integer | The id for the task that will be executed under this schedule. |
| created | R | datetime | The datetime stamp when the schedule was created. |
| modified | R | datetime | The datetime stamp when the schedule was last modified. |
| owner | O | string | The unique ID of the User that created the ScheduleEntry. |

* + - 1. Task Status

This section describes the objects used within Task status messages.

* + - * 1. Task Status

The Task object is used in the Task response message defined in Table 12.

1. —Task Object

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| schedule\_id | R | string | The unique ID or locator for the ScheduleEntry. |
| schedule\_name | R | string | The name of the ScheduleEntry. |
| sensor\_id | O | string | The id of the sensor executing the task. |
| task\_id | C | integer | The id of the task that generated\or will generate the acquisition. The task\_id shall be included when the request includes a schedule\_id parameter. |
| started | R | datetime | The datetime stamp for when the schedule or task started, or will be started. |
| finished | C | datetime | The datetime stamp for when the schedule or task finished. The finished property shall be included when the schedule or task has finished. |
| duration | O | time | The duration of the action |
| status | R | string | The status of the acquisition(s). The status shall equal success, fail, in-progress, or scheduled. |
| archive\_id | C | string | The unique id or locator for the archive of the acquisition. The archive\_id shall be included when the object is describing an acquisition from a task that successfully completed. |

* + - * 1. Recording

The Recording object is used within the Acquisition notification message defined in Table 13.

1. —Recording Object

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| metadata | R | JSON object containing SigMF Metadata | The SigMF metadata associated with the action. |
| data | O | byte[] | The binary sensed data. |

1. Protocol Specific Details
   1. Overview

The SCOS standard is protocol agnostic. However, normative specifications may be adopted as they mature. This section details the normative specifications for the HTTP and MQTT implementations of the SCOS standard.

* 1. HTTP

This section details the normative HTTP based implementation of the standard. The HTTP implementation utilizes the messages defined in 5.2.2 and 6.2.2, however many Conditional properties of the request are instead handled as path parameters and HTTP response codes. All URLs in the HTTP implementation begin with /api/{version}.

* + 1. Sensor

This section describes the HTTP URLs used to request and provide Sensor status and Capabilities, perform scheduling, provide task status, and provide Data Distribution. Rather than describing the URLs according to aforementioned functions, they are organized by endpoint.

* + - 1. Status

Table 26 describes the status endpoint. The status endpoint provides the Sensor status functionality specified in 5.2.2.2.

1. —Status Endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Method | Response Body | Description/Notes |
| /status | GET | status response (see Table 4) | Request Sensor Status |

* + - 1. Capabilities

Table 27 describes the capabilities endpoint. The capabilities endpoint provides the Sensor capabilities functionality specified in 5.2.2.3.

1. —Capabilities Endpoint

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Method | Response Body | Description/Notes |
| /capabilities | GET | capabilities response (see Table 6) | Request Sensor Capabilities |

* + - 1. Schedule

Table 28 describes the schedule endpoint. The schedule endpoint provides the schedule, task status, and archive request functionality described in 5.2.2.4, 5.2.2.5, 5.2.2.6.

1. —Schedule Endpoint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Request | Methods | Request Body | Response Body | Description/Notes |
| /schedule | POST, GET | Schedule entry request (See Table 7) with POST | Schedule overview response (see Table 10) | Create Schedule Entry or view Schedule Entries. Get requests allow option limit and offset URL parameters. |
| /schedule/{schedule\_id} | GET, PUT, PATCH, DELETE | Schedule entry request (See Table 7) with PUT,PATCH | Schedule overview response (see Table 10) | View, Update, or Delete Schedule Entry |
| /schedule /{schedule\_id}/tasks | GET, DELETE | N/A | Task Status Response (Table 12) or HTTP Response Code | View the task status of the schedule’s tasks or delete the schedule’s tasks. |
| /schedule/{schedule\_id}/tasks/{task\_id} | DELETE, GET | N/A | Task Status Response (Table 12) or HTTP Response Code | Delete acquisition for a task or view the acquisition status for a task. |
| /schedule/{schedule\_id}/tasks/{task\_id}/archive | GET | N/A | SigMF TAR archive or HTTP Response Code | Retrieve the acquisition archive for a specific task |

* + 1. Manager

This section describes the Manager’s HTTP URLs used to request and provide Sensor status and Capabilities, perform scheduling, provide Task status, and provide data distribution. Rather than describing the URLs according to aforementioned functions, they are organized by endpoint.

* + - 1. Sensors

1. —Sensors Endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Method | Request Body | Response Body |
| /sensors | POST | Sensor association request (see Table 1) | Sensor association response (see Table 2) |
| /sensors | GET | NA | Sensors response (see Table 18) |
| /sensors/{sensor\_id}/status | GET | NA | status response (see Table 4) |
| /sensors/{sensor\_id}/capabilities | GET | NA | capabilities response (see Table 6) |

* + - 1. Clients

1. —Client Endpoint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| URL | Method | Request Body | Response Body | Description/Notes |
| /clients | POST, DELETE | client association request (see Table 15 | client association response (see Table 16) | Associate or delete a Data Client |

* + - 1. Schedule

1. —Schedule Endpoint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Request | Methods | Request Body | Response Body | Description/Notes |
| /schedule | POST, GET | Schedule entry request (See Table 7) with POST | Schedule overview response (see Table 10) | Create Schedule Entry or view Schedule Entries. Get requests allow option limit and offset URL parameters. |
| /schedule/{schedule\_id} | GET, PUT, PATCH, DELETE, POST | Schedule entry request (See Table 7) with PUT, PATCH. Acquisition notification (see Table 13) in POST. | Schedule overview response (see Table 10) | View, Update, or Delete Schedule Entry |
| /schedule /{schedule\_id}/tasks | GET, DELETE | N/A | Task Status Response (Table 12) or HTTP Response Code | View or delete the tasks that have completed in a schedule. Deleting the tasks will delete the acquisitions generated by the task. |
| /schedule/{schedule\_id}/tasks/{task\_id} | GET, DELETE | N/A | Task Status Response (Table 12) or HTTP Response Code | View or delete a single completed task. Deleting the task will delete the acquisition generated by the task. |
| schedule/{schedule\_id}/acquisitions/{task\_id}/archive | GET | N/A | SigMF TAR archive or HTTP Response Code | Download the archive of the SigMF metadata and data file generated by the task. |

# SigMF Metadata Extensions

## Overview

This section defines the informative SigMF metadata extensions that should be included within the metadata files produced by a Sensor. Some of the objects defined in these extensions are also used within the messages.

## scos-core

The scos-core extension provides generally useful metadata extensions that may be referenced in other namespace extensions.

### Global

The scos-core namespace does not extend the global object, but defines additional objects that may be used in other global object extensions. The Antenna object is utilized within the Sensor object and contains the following name/value pairs.

1. —Antenna Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| antenna\_spec | R | HardwareSpec (see Table 33) | N/A | Metadata describing the physical hardware of the antenna. |
| type | O | string | N/A | Antenna type. E.g. "dipole", "biconical", "monopole", "conical monopole". |
| low\_frequency | O | number | Hz | Low frequency of operational range. |
| high\_frequency | O | number | Hz | High frequency of operational range. |
| polarization | O | number | string | Antenna polarization. E.g. "vertical", "horizontal", "slant-45", "left-hand circular", "right-hand circular". |
| cross\_polar\_discrimination | O | number | N/A | Cross-polar discrimination. |
| gain | O | number | dBi | Nominal gain of the antenna where additional details may be obtained from manufacturing specification. |
| horizontal\_gain\_pattern | O | number[] | dBi | Antenna gain pattern in horizontal plane from 0 to 359 degrees in 1 degree steps. |
| vertical\_gain\_pattern | O | number[] | dBi | Antenna gain pattern in vertical plane from -90 to +90 degrees in 1 degree steps. |
| horizontal\_beam\_width | O | number | degrees | Horizontal 3-dB beamwidth. |
| vertical\_beam\_width | O | number | degrees | Vertical 3-dB beamwidth. |
| voltage\_standing\_wave\_ratio | O | number | volts | Voltage standing wave ratio. |
| cable\_loss | O | number | dB | Cable loss for cable connecting antenna and preselector. |
| steerable | O | boolean | N/A | Defines if the antenna is steerable or not. |

1. —HardwareSpec Object

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| id | R | string | Unique id of hardware. E.g., serial number. |
| model | O | string | Hardware make and model. |
| version | O | string | Hardware version. |
| description | O | string | Description of the hardware. |
| supplemental\_information | O | string | Information about hardware, e.g., URL to on-line data sheets. |

### Captures

The scos-core namespace does not extend the captures object.

### Annotations

The scos-core namespace extends the annotations segment object with the annotation\_type key defined in Table 34. In addition, scos-core defines a new antenna annotation segment type, defined in Table 35, which extends the top level annotation segment object.

1. —Annotation Extension

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| annotation\_type | O | string | The annotation type. |

1. —Antenna Annotation Segment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| id | R | string | N/A | Unique id of an antenna object defined in the global object. |
| azimuth\_angle | O | number | degrees | Angle of main beam in azimuthal plane from North. |
| elevation\_angle | O | number | degrees | Angle of main beam in elevation plane from horizontal. |
| scos-core:annotation\_type | R | string | N/A | AntennaAnnotation |

## scos-sensor

The scos-sensor namespace provides metadata to describe RF sensors.

### Global

The scos-sensor namespace extends the global object with the kev/value pairs defined in Table 36. The sensor extension to the global objects utilizes the Sensor object defined in Table 37, which uses the HardwareSpec (Table 33), Preselector (Table 38), RFPath (Table 42), and SignalAnalyzer (Table 43) objects. These objects are also used in the Sensor capabilities messages defined in 5.2.2.3, which are also used by the Manager as specified in 6.2.2.3.

1. —Global Extensions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| sensor | O | Sensor (see Table 37) | N/A | JSON object describing the Sensor. |
| calibration\_datetime | O | datetime | ISO-8601 (ISO 8601-1:2019) | Time of last calibration. |

1. —Sensor Object

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| sensor\_spec | R | HardwareSpec (see Table 33) | Metadata describing the Sensor hardware. |
| antenna | R | Antenna (see Table 32) | Metadata describing the physical antenna. |
| preselector | O | Preselector (see Table 38) | Metadata describing the preselector. |
| signal\_analyzer | R | SignalAnalyzer (see Table 43) | Metadata describing the signal analyzer. |
| computer\_spec | O | HardwareSpec (see Table 33) | Description of host compute. E.g. Make, model, and configuration. |
| mobile | O | boolean | Indicates if the Sensor is mobile |

1. —Preselector Object

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| preselector\_spec | O | HardwareSpec (see Table 33) | Metadata to describe the physical preselector component. |
| cal\_source | O | CalSource (see Table 41 ) | Metadata to describe/specify the preselector calibration source. |
| amplifiers | O | Amplifier[] (see Table 39) | Metadata to describe/specify the preselector low noise amplifier. |
| filters | O | Filter[] (see Table 33) | Metadata to describe/specify the preselector RF bandpass filters. |
| rf\_paths | O | RFPath[] (see Table 42) | Metadata that describes preselector RF paths. |

1. —Amplifier Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| amplifier\_spec | O | HardwareSpec (see Table 33) | N/A | Metadata to describe the amplifier specification. |
| gain | O | number | dB | Gain of the low noise amplifier. |
| noise\_figure | O | number | dB | Noise figure of the low noise amplifier. |
| max\_power | O | number | dB | The maximum power of the low noise amplifier. |

1. —Filter Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| filter\_spec | O | HardwareSpec (see Table 33) | N/A | Metadata to describe/specify the filter specification. |
| low\_frequency\_passband | O | number | Hz | Low frequency of filter 1-dB passband. |
| high\_frequency\_passband | O | number | Hz | High frequency of filter 1-dB passband. |
| low\_frequency\_stopband | O | number | Hz | Low frequency of filter 60-dB stopband. |
| high\_frequency\_stopband | O | number | Hz | High frequency of filter 60-dB stopband. |

1. —CalSource Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| cal\_source\_spec | O | HardwareSpec (see Table 33) | N/A | Metadata to describe the calibration source. |
| type | O | string | N/A | The type of the calibration source. |
| enr | O | number | dB | The excess noise ration of the calibration source. |

1. —RFPath Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| low\_frequency\_passband\_filter | O | number | Hz | Low frequency of filter 1-dB passband. |
| high\_frequency\_passband\_filter | O | number | Hz | High frequency of filter 1-dB passband. |
| low\_frequency\_stopband\_filter | O | number | Hz | Low frequency of filter 60-dB stopband. |
| high\_frequency\_stopband\_filter | O | number | Hz | High frequency of filter 60-dB stopband. |
| gain\_lna | O | number | dB | Gain of low noise amplifier. |
| noise\_figure\_lna | O | number | dB | Noise figure of low noise amplifier. |
| type\_cal\_source | O | string | N/A | E.g., "calibrated noise source". |

1. —SignalAnalyzer Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| sigan\_spec | O | HardwareSpec (see Table 33) | N/A | Metadata to describe/specify the signal analyzer. |
| low\_frequency | O | number | Hz | Low frequency of operational range of the signal analyzer. |
| high\_frequency | O | number | Hz | High frequency of operational range of the signal analyzer. |
| noise\_figure | O | number | dB | Noise figure of the signal analyzer. |
| max\_power | O | number | dBm | Maximum input power of the signal analyzer. |
| a2d\_bits | O | integer | bits | Number of bits in A/D converter. |

### Captures

The scos-sensor namespace does not extend the captures object.

### Annotations

The scos-sensor namespace provides the SensorAnnotation object, defined in Table 44, to record changes that may have occurred in the sensor while the dataset was being recorded. The Calibration annotation, defined in Table 45, details the calibration settings that were used during the recording.

1. —SensorAnnotation Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| rf\_path\_index | O | integer | N/A | Index of the RFPath object. |
| overload\_sensor | O | boolean | N/A | Indicator of Sensor overload. |
| overload\_sigan | O | boolean | N/A | Indicator of signal analyzer overload. |
| attenuation\_setting\_sigan | O | number | dB | Attenuation setting of the signal analyzer. |
| gain\_setting\_sigan | O | number | dB | Gain setting of the signal analyzer. |
| latitude | O | number | decimal degrees | Latitude. |
| longitude | O | number | decimal degrees | Longitude. |
| altitude | O | number | meters | Height above mean sea level. |
| speed | O | number | m/s | Speed. |
| bearing | O | number | degrees | Direction (angle relative to true North). |
| gps\_nmea | O | string | NMEA | NMEA message from gps receiver. |
| annotation\_type | R | string | N/A | The annotation\_type shall equal SensorAnnotation when used within a Sensor Annotation. |

1. —CalibrationAnnotation Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| gain\_sigan | O | number | dBm | Gain of signal analyzer (may differ with signal analyzer gain setting). |
| noise\_figure\_sigan | O | number | dB | Noise figure of signal analyzer. |
| 1db\_compression\_point\_sigan | O | number | dBm | Maximum input of signal analyzer. |
| enbw\_sigan | O | number | Hz | Equivalent noise bandwidth of signal analyzer. |
| gain\_preselector | O | number | dB | Gain of sensor preselector. |
| noise\_figure\_sensor | O | number | dB | Noise figure of sensor. |
| 1db\_compression\_point\_sensor | O | number | dBm | Maximum input of sensor. |
| enbw\_sensor | O | number | Hz | Equivalent noise bandwidth of sensor. |
| mean\_noise\_power\_sensor | O | number | dBm/Hz | Mean noise power density of sensor. |

## scos-aglorithm

The scos-algorithm extension describes algorithms applied to measurements.

### Global

Thes scos-algorithm namespace extends global with the optional anti\_aliasing\_filter key that uses the DigitalFilter object defined in Table 47.

1. —Global Extensions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| anti\_aliasing\_filter | O | DigitalFilter | N/A | Digital filter applied to data to avoid aliasing |

1. —DigitalFilter Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | Required | Type | Unit | Description |
| filter\_type | O | string | N/A | Description of digital filter, e.g., "FIR", "IIR" |
| FIR\_coefficients | O | number[] | N/A | Coefficients that defines FIR filter. |
| IIR\_numerator\_coefficients | O | number[] | N/A | Coefficients that defines IIR filter. |
| IIR\_denominator\_coefficients | O | number[] | N/A | Coefficients that defines IIR filter. |
| cutoff\_attenuation | O | number | dB | Attenuation that specifies the cutoff\_frequency (typically 3 dB). |
| cutoff\_frequency | O | number | Hz | Frequency that characterizes boundary between passband and stopband. |
| ripple\_passband | O | number | dB | Ripple in passband. |
| attenuation\_stopband | O | number | dB | Attenuation of stopband. |
| frequency\_stopband | O | number | Hz | Point in filter frequency response where stopband starts. |

### Captures

The scos-algorithm namespace does not extend the captures object.

### Annotations

The scos-algorithm namespace provides the TimeDomainDetection, FrequencyDomainDetection, and the DigitalFilterAnnotation annotation extensions defined in Table 48, Table 49, and Table 50. The TimeDomainDetection annotation describes algorithms applied to gap-free IQ time series data captured at a single frequency. The FrequencyDomainDetection annotation describes discrete Fourier transforms of gap-free IQ time series captured at a single frequency. The DigitalFilterAnnoation annotation extension is used to describe changes that were made to anti-aliasing filter as the recordings were captured.

1. —TimeDomainDetection Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| detector | R | string | N/A | E.g. “sample\_iq”, "sample\_power", "mean\_power", "max\_power", "min\_power", "median\_power", "m4s\_power". |
| detection\_domain | R | string | N/A | Domain in which detector is applied, i.e., "time". |
| number\_of\_samples | R | integer | N/A | Number of samples to be integrated over by detector. |
| units | R | string | N/A | Data units, e.g., "dBm", "watts", "volts". |
| reference | O | string | N/A | Data reference point, e.g., "receiver input", "antenna output", "output of isotropic antenna". |
| time | O | number[] | seconds | Time array corresponding to detected data. |
| time\_start | O | number | seconds | Time of the first data point. |
| time\_stop | O | number | seconds | Time of the last data point. |
| time\_step | O | number | seconds | Time step between data points. |
| scos-core:annotation\_type | R | string | N/A | TimeDomainDetection |

1. —FrequencyDomainDetection Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| detector | R | string | N/A | E.g. "fft\_sample\_iq", "fft\_sample\_power", "fft\_mean\_power", "fft\_max\_power", "fft\_min\_power", "fft\_median\_power". |
| detection\_domain | R | string | N/A | Domain in which detector is applied, i.e., "frequency". |
| number\_of\_ffts | R | integer | N/A | Number of FFTs to be integrated over by detector. |
| number\_of\_samples\_in\_fft | R | integer | N/A | Number of samples in FFT to calcluate delta\_f = samplerate/number\_of\_samples\_in\_fft. |
| window | R | string | N/A | E.g. "blackman-harris", "flattop", "gaussian\_a3.5", "gauss top", "hamming", "hanning", "rectangular". |
| equivalent\_noise\_bandwidth | O | number | Hz | Bandwidth of brickwall filter that has same integrated noise power as that of the actual filter. |
| units | R | string | N/A | Data units, e.g., "dBm", "watts", "volts". |
| reference | O | string | N/A | Data reference point, e.g., "receiver input", "antenna output", "output of isotropic antenna". |
| frequency | O | number[] | Hertz | Frequency array corresponding to detected data |
| frequency\_start | O | number | Hertz | Frequency of the first data point. |
| frequency\_stop | O | number | Hertz | Frequency of the last data point. |
| frequency\_step | O | number | Hertz | Frequency step between data points. |
| scos-core:annotation\_type | R | string | N/A | FrequencyDomainDetection |

1. —DigitalFilterAnnotation Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | Required | Type | Unit | Description |
| filter\_type | O | string | N/A | Description of digital filter, e.g., "FIR", "IIR" |
| FIR\_coefficients | O | number[] | N/A | Coefficients that defines FIR filter. |
| IIR\_numerator\_coefficients | O | number[] | N/A | Coefficients that defines FIR filter. |
| IIR\_denominator\_coefficients | O | number[] | N/A | Coefficients that defines FIR filter. |
| cutoff\_attenuation | O | number | dB | Attenuation that specifies the cutoff\_frequency (typically 3 dB). |
| cutoff\_frequency | O | number | Hz | Frequency that characterizes boundary between passband and stopband. |
| ripple\_passband | O | number | dB | Ripple in passband. |
| attenuation\_stopband | O | number | dB | Attenuation of stopband. |
| frequency\_stopband | O | number | Hz | Point in filter frequency response where stopband starts. |
| scos-core:annotation\_type | R | string | N/A | DigitalFilterAnnotation |

## scos-emitter

The scos-emitter namespace provides extensions to describe RF emitters.

### Global

The scos-emitter namespace extension extends the global object with an emitter key and defines an Emitter object to describe the properties of an RF emitter.

1. —Global Object Extensions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| emitters | O | Emitter[](see Table 52) | N/A | Metadata that describe emitters |

1. —Emitter Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| id | R | string | N/A | Unique id of the emitter. |
| description | O | string | N/A | Description of the emitter. |
| power | O | number | dBm | Power referenced to antenna input. |
| antenna | O | Antenna (see Table 32) | N/A | Metadata that describes the antenna. |
| transmitter | O | Transmitter (see Table 53) | N/A | Metadata that describes the transmitter. |

1. —Transmitter Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| model | R | string | N/A | Make and model of the transmitter. E.g. "Agilent E4438C" |

### Captures

ntia-emitter does not extend captures.

### Annotations

The ntia-emitter namespace defines an EmitterAnnotation annotation extension to describe the properties of an RF emitter that may change during the course of a recording.

1. —EmitterAnnotation Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| id | R | string | N/A | Unique id of the emitter. |
| waveform | O | Waveform | N/A | Metadata that describes transmitted waveform. |
| latitude | O | number | decimal degrees | Latitude. |
| longitude | O | number | decimal degrees | Longitude. |
| altitude | O | number | meters | Height above mean sea level. |
| speed | O | number | m/s | Speed. |
| bearing | O | number | degrees | Direction (angle relative to true North). |
| scos-core:annotation\_type | R | string | N/A | EmitterAnnotation |

## scos-waveform

The scos-waveform namespace provides models and parameters for textbook and standardized complex-baseband waveforms. The intention is to provide a library of simulated waaveforms for testing and training signal identification algorithms. The waveform library may also be used for signal generation purposes in system level interference tests.

### Global

Scos-waveform does not directly extend the Global object. Instead, scos-waveform defines the waveform object and additional extensions of the waveform object like the IEEE80211p object. The waveform objects may be utilized within other objects, like the emitter annotation defined in Table 54.

1. —Waveform Object

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| model | R | string | The type of the waveform. E.g. “IEEE80211p”. |

1. —IEEE80211p Object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| info\_bit\_generation | O | string | N/A | Model that defines information bit generation. E.g. "PN". |
| coding\_rate | O | array [k, n] | bits | For every k bits of useful information the coder generates n bits of data. E.g. [1, 2], [2, 3], [3, 4]. |
| packet\_length | O | integer | bits | Packet length. |
| modulation | O | string | N/A | Modulation, e.g., "BPSK", "QPSK", "16QAM", "64QAM". |
| encoder | O | string | N/A | Description of encoder. E.g. "Convolutional". |
| number\_of\_subcarriers | O | integer | N/A | Number of subcarriers. |
| number\_of\_data\_subcarriers | O | integer | N/A | Number of data subcarriers. |
| number\_of\_pilots | O | integer | N/A | Number of pilots. |
| cyclic\_prefix | O | integer | bits | Size of cyclic prefix. |
| short\_inter\_frame\_space | O | number | microseconds | Time required to process a received frame and to respond with a response frame. |
| preamble\_frame | O | array | N/A | Preamble of 0's and 1's used for synchronization and id beginning of frame. |
| number\_of\_info\_bits | false | integer | N/A | Number of information bits. |
| signal\_to\_noise\_ratio | false | float | dB | Signal-to-noise ratio. If unspecified, assumed no noise present. |

## scos-environment

The scos-environment namespace provides SigMF metadata extensions to characterize the environment factors around a sensor and\or emitter.

### Global

The ntia-environment namespace does not extend the global object.

### Captures

The scos-environment namespace does not extend captures.

### Annotations

The scos-environment namespace provides SensorEnvironment and EmitterEnvironment annotations to describe the environment around Sensors and Emitters.

1. —SensorEnvironment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| category | O | string | N/A | Categorical description of the environment where sensor is mounted. E.g. "indoor", "outdoor-urban", "outdoor-rural". |
| temperature | O | number | Celsius | Environmental temperature. |
| humidity | O | number | % | Relative humidity. |
| weather | O | string | N/A | Weather around the sensor. E.g. "rain", "snow".) |
| scos-core:annotation\_type | R | string | N/A | SensorEnvironment |

1. —EmitterEnvironment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | R/O/C | Type | Unit | Description |
| emitter\_id | R | string | N/A | Unique emitter id |
| category | O | string | N/A | Categorical description of the environment where sensor is mounted. E.g. "indoor", "outdoor-urban", "outdoor-rural". |
| temperature | O | number | Celsius | Environmental temperature. |
| humidity | O | number | % | Relative humidity. |
| scos-core:annotation\_type | O | string | N/A | EmitterEnvironment |

## scos-acquisition

The scos-acquisition namespace provides additional extensions that are useful in maintaining the provenance of acquisitions.

### Global

The scos-acquisition namespace extends the global object with the key/value pairs defined in Table 59.

1. —Global Extensions

|  |  |  |  |
| --- | --- | --- | --- |
| Property | R/O/C | Type | Description |
| action | O | string | The name of the action the produced the acquisition. |
| schedule\_entry | O | ScheduleEntry (see Table 23) | The schedule that produced the acquisition. |
| task | O | integer | The id of the task. |
| start\_time | O | datetime | When the action started. |
| end\_time | O | datetime | When the action finished. |
| archive\_path | O | string | The location of the archive file. |

### Captures

The scos-acquisition namespace does not extend captures.

### Annotations

The scos-acquisition namespace does not extend annotations.

1. SigMF specification is available at: <http://sigmf.org> [↑](#footnote-ref-1)
2. IEEE Standards Dictionary Online is available at: <http://dictionary.ieee.org>

   . [↑](#footnote-ref-2)