vxgcloudagent

1.2.34

Generated by Doxygen 1.8.17

Chapter 1

VXG Cloud Agent Library

Chapter 2

Build System

2.0.1 Overview

VXG Cloud Agent library uses Meson build system as a modern, fast and flexible build system that supports easy to set up and maintain a cross-compilation process.

It's recommended to refer to the Meson guide.

2.0.2 C++ Toolchain Requirements

IMPORTANT: This projects requires C++ toolchain with C++11 support

VXG Cloud Agent Library requires modern C++11 so in order to build and use this library the user needs a compiler with C++11 support.
GCC supports C++11 since version 4.8.1 released on May 31, 2013.

> **C++11 Support in GCC**
> GCC 4.8.1 was the first feature-complete implementation of the 2011 C++ standard, previously known as C++0x.
>
> This mode can be selected with the -std=c++11 command-line flag, or -std=gnu++11 to enable GNU extensions as well.

2.0.3 Build system installation

IMPORTANT: This projects requires Meson version >= 0.56.0

It's recommended to use Ubuntu 20.04 LTS distribution in development process but other distributions or operation systems are also supported by Meson.

Please refer to Meson installation guide to get and install Meson, preferable way to install Meson is pip method.

Quick install guide for Ubuntu 20.04. If you have an old version of meson already installed please remove it first.

```
sudo apt-get update
sudo apt-get install -y python3-pip git ninja-build curl tzdata python3-tz
pip3 install git+https://github.com/mesonbuild/meson@0.56.0
# pip3 puts meson main script into the $HOME/.local/bin/ directory, you need to
# add $HOME/.local/bin/ into your PATH environment variable, for bash shell you
# can run the following command and restart the shell session.
echo 'export PATH=$HOME/.local/bin:$PATH' » $HOME/.bashrc
# Check currently installed meson version
meson -v
```

Chapter 3

Application Development

3.1 Overview

An application that uses VXG Cloud Agent Library should implement 3 classes derived from the base classes provided by the library:

- agent::callback - common callbacks class, only on_bye callback is mandatory for implementation
- agent::media::stream class, abstract class for media streams, library provides basic media::rtsp_stream implementation which retransmits RTSP source stream to the endpoint of the VXG Cloud, all callbacks are stubbed. Developer normally should implement own class derived from the media::stream with own vxg::media::Streamer::ISource implementation(vxg::media::ffmpeg::Source class implementation from the ffmpeg_source.cc can be used as a reference), or if RTSP source is acceptable developer can implement own class derived from the media::rtsp_stream but with callbacks implemented.
- agent::event_stream class, abstract class for events generation.

Any callback implementation as well as ISource::init and ISource::finit implementations should be non-blocking, VXG Cloud messages processing is single-threaded which means any VXG Cloud messages are handled sequentially hence no new message will be processed until the callback triggered by the previous message is returned.

The library provides the stub implementation for most of the virtual methods of these classes, the stub implementation prints a log message about this method is not implemented and returns an error, the final application should implement all virtual methods on its own.

Most of the callbacks are just getter/setter for the library's objects.

3.2 Examples

3.2.1 Minimal application example

Headers and namespaces:

```
#include <agent/manager.h>
#include <agent/rtsp-stream.h>
#include <utils/logging.h>
#include <utils/properties.h>
using namespace vxg::cloud;
using namespace vxg::cloud::agent;
```

Common callbacks class, minimal implementation derived from the agent::callback class:

```
using namespace vxg::cloud;
class agent_callback_minimal : public agent::callback {
public:
    virtual void on_bye(proto::bye_reason reason) override {
        vxg::logger::warn("Connection close {}", json(reason).dump());
    }
    virtual void on_registered(const std::string& sid) override {
        // Save Cloud registration session id in the local properties file.
        // This is required for the fast reconnection to the Cloud.
        props.set("prev_sid", sid);
    }
};
```

Create and start agent object agent::manager with one basic media stream agent::media::rtsp_stream

```
    using namespace vxg::cloud::agent;
    // Agent
    manager::ptr agent;
    // VXG Cloud token
    proto::access_token::ptr access_token =
        proto::access_token::parse(vxg_cloud_token);
    // Agent callback
    callback::ptr cb = std::make_unique<agent_callback_minimal>();
    // Media stream
    std::vector<agent::media::stream::ptr> streams;
    media::stream::ptr stream =
        std::make_shared<media::rtsp_stream>(rtsp_url, "DemoStream");
    streams.push_back(stream);
    // Create agent
    if ((agent = agent::manager::create(std::move(cb), access_token,
                                        streams)) == nullptr) {
        vxg::logger::error("Failed to create agent");
        return EXIT_FAILURE;
    }
    if (!quit && !agent->start())
        quit = true;
```

Complete minimal example:

```
#include <signal.h>
#include <args.hxx>
#include <agent/manager.h>
#include <agent/rtsp-stream.h>
#include <utils/logging.h>
#include <utils/properties.h>
using namespace vxg::cloud;
using namespace vxg::cloud::agent;
static bool quit = 0;
static vxg::properties props;
#if !defined(_WIN32)
static void signal_handler(int sig) {
    if (sig == SIGINT || sig == SIGTERM) {
        fprintf(stderr, "\nSIGTERM received\n\n");
        quit = true;
    }
}
#endif
using namespace vxg::cloud;
class agent_callback_minimal : public agent::callback {
public:
    virtual void on_bye(proto::bye_reason reason) override {
        vxg::logger::warn("Connection close {}", json(reason).dump());
    }
    virtual void on_registered(const std::string& sid) override {
        // Save Cloud registration session id in the local properties file.
        // This is required for the fast reconnection to the Cloud.
        props.set("prev_sid", sid);
    }
};
std::string vxg_cloud_token;
std::string rtsp_url;
bool parse_args(int argc, char** argv) {
    args::ArgumentParser parser("This is a test program.", "");
    args::HelpFlag help(parser, "help", "Display this help menu",
                        {'h', "help"});
    args::CompletionFlag completion(parser, {"complete"});
    args::Positional<std::string> token(parser, "vxg_cloud_token",
                                        "VXG Cloud Access Token", "",
                                        args::Options::Required);
    args::Positional<std::string> url(parser, "rtsp_url", "RTSP stream url", "",
                                      args::Options::Required);
    args::Flag secure_connection_arg(
        parser, "",
        "Use secure cloud connetion(enables encryption, cloud agent library "
        "must be compiled with openssl support enabled)",
        {"secure-channel", 's'});
```

```
    try {
        parser.ParseCLI(argc, argv);
        vxg_cloud_token = args::get(token);
        rtsp_url = args::get(url);
        profile::global::instance().insecure_cloud_channel =
            !args::get(secure_connection_arg);
    } catch (const args::RequiredError& e) {
        std::cout << e.what() << std::endl;
        return false;
    } catch (const args::Completion& e) {
        std::cout << e.what();
        return false;
    } catch (const args::Help&) {
        std::cout << parser;
        return false;
    } catch (const args::ParseError& e) {
        std::cerr << e.what() << std::endl;
        std::cerr << parser;
        return false;
    }
    return true;
}
int main(int argc, char** argv) {
    vxg::properties::reset("agent-test.props");
    // Try to load and set previously saved session id.
    // This is required for the fast reconnection to the Cloud.
    if (!props.get("prev_sid").empty())
        profile::global::instance().cm_registration_sid = props.get("prev_sid");
    // Parse args and retreive token and rtsp url
    if (!parse_args(argc, argv))
        return EXIT_FAILURE;
#if !defined(_WIN32)
    // Catch signal
    signal(SIGINT, signal_handler);
    signal(SIGTERM, signal_handler);
    signal(SIGPIPE, signal_handler);
#endif
    vxg::logger::info("VXG Cloud Agent Library Version: {}",
                      vxg::cloud::agent::version());
    using namespace vxg::cloud::agent;
    // Agent
    manager::ptr agent;
    // VXG Cloud token
    proto::access_token::ptr access_token =
        proto::access_token::parse(vxg_cloud_token);
    // Agent callback
    callback::ptr cb = std::make_unique<agent_callback_minimal>();
    // Media stream
    std::vector<agent::media::stream::ptr> streams;
    media::stream::ptr stream =
        std::make_shared<media::rtsp_stream>(rtsp_url, "DemoStream");
    streams.push_back(stream);
    // Create agent
    if ((agent = agent::manager::create(std::move(cb), access_token,
                                        streams)) == nullptr) {
        vxg::logger::error("Failed to create agent");
        return EXIT_FAILURE;
    }
    if (!quit && !agent->start())
        quit = true;
    // Spin main thread until stopped
    while (!quit) {
        std::this_thread::sleep_for(std::chrono::seconds(1));
    }
    agent->stop();
    agent = NULL;
    vxg::logger::info("Agent stopped");
    return EXIT_SUCCESS;
}
```

3.2.2 Complete application example

Common callback class: derived from agent::callback

```
using namespace vxg::cloud;
class my_agent_callback : public agent::callback {
public:
    virtual void on_bye(proto::bye_reason reason) override {
        vxg::logger::error("Error {}", json(reason).dump());
    }
    virtual bool on_raw_msg(std::string client_id, std::string& data) override {
        vxg::logger::info("Raw message {} from client '{}'", data, client_id);
        // Reply json
```

```
        data = "{\"reply\": \"OK\"}";
        return true;
    }
    virtual bool on_get_log(std::string& log_data) override {
        log_data = "log messages...";
        vxg::logger::warn("{} not implemented", __func__);
        return true;
    }
    virtual bool on_start_backward_audio(std::string url) override {
        // Start backward audio playback from url
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_stop_backward_audio(std::string url) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_get_cam_video_config(proto::video_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_set_cam_video_config(
        const proto::video_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_get_cam_audio_config(proto::audio_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_set_cam_audio_config(
        const proto::audio_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_get_ptz_config(proto::ptz_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_cam_ptz(proto::ptz_command& command) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_get_osd_config(proto::osd_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_set_osd_config(const proto::osd_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_get_wifi_config(proto::wifi_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_set_wifi_config(
        const proto::wifi_network& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_get_motion_detection_config(
        proto::motion_detection_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_set_motion_detection_config(
        const proto::motion_detection_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_get_cam_events_config(
        proto::events_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_set_cam_events_config(
        const proto::events_config& config) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_get_timezone(std::string& timezone) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool on_set_timezone(std::string timezone) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
```

```
    }
    virtual bool on_get_memorycard_info(
        proto::event_object::memorycard_info_object& info) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
};
```

Media stream callback class: derived from agent::media::stream

```
class my_media_stream : public media::rtsp_stream {
public:
    my_media_stream(std::string url, std::string name)
        : media::rtsp_stream(url, name) {}
    bool get_supported_stream(proto::supported_stream_config& config) override {
        vxg::logger::warn("{} default implementation should be overriden",
                          __func__);
        config.id = cloud_name();
        config.video = "Video" + std::to_string(0);
        // config.audio = "Audio" + std::to_string(0);
        return true;
    }
    virtual bool get_stream_caps(proto::stream_caps& caps) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool get_stream_config(
        proto::stream_config& streamConfig) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool set_stream_config(
        const proto::stream_config& streamConfig) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool get_snapshot(
        proto::event_object::snapshot_info_object& snapshot) override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual std::vector<proto::video_clip_info> record_get_list(
        vxg::cloud::time begin,
        vxg::cloud::time end,
        bool align) override {
        std::vector<proto::video_clip_info> empty_vector(0);
        vxg::logger::warn("{} not implemented", __func__);
        return empty_vector;
    }
    virtual proto::video_clip_info record_export(
        vxg::cloud::time begin,
        vxg::cloud::time end) override {
        proto::video_clip_info clip;
        vxg::logger::warn("{} not implemented", __func__);
        // empty clip
        return clip;
    }
    virtual bool start_record() override {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool stop_record() override {
        vxg::logger::warn("{} not implemented", __func__);
        return true;
    }
};
```

Event stream callback class: derived from agent::media::event_stream

```
class my_event_stream : public agent::event_stream {
public:
    my_event_stream(std::string name) : agent::event_stream(name) {}
    virtual bool start() {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual void stop() { vxg::logger::warn("{} not implemented", __func__); }
    virtual bool init() {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual void finit() { vxg::logger::warn("{} not implemented", __func__); }
    virtual bool set_trigger_recording(bool enabled, int pre, int post) {
        vxg::logger::warn("{} not implemented", __func__);
        return false;
    }
    virtual bool get_events(std::vector<proto::event_config>& configs) {
```

```
        return false;
    }
    virtual bool set_events(const std::vector<proto::event_config>& config) {
        return false;
    }
};
```

Creating and start agent instance with all callbacks:

```
    using namespace vxg::cloud::agent;
    // Agent
    manager::ptr agent;
    // VXG Cloud token
    proto::access_token::ptr access_token =
        proto::access_token::parse(vxg_cloud_token);
    // Agent callback
    callback::ptr cb = std::make_unique<my_agent_callback>();
    // Media stream
    std::vector<agent::media::stream::ptr> streams;
    media::stream::ptr stream =
        std::make_shared<my_media_stream>(rtsp_url, "MyMediaStream");
    streams.push_back(stream);
    // Event stream
    std::vector<agent::event_stream::ptr> event_streams;
    event_stream::ptr event_stream =
        std::make_shared<my_event_stream>("MyEventStream");
    event_streams.push_back(event_stream);
    // Create agent
    if ((agent = agent::manager::create(std::move(cb), access_token, streams,
                                        event_streams)) == nullptr) {
        vxg::logger::error("Failed to create agent");
        return EXIT_FAILURE;
    }
    if (!quit && !agent->start())
        quit = true;
```

3.2.3 Linking application against the VXG Agent Cloud Library

There are 3 possible ways of how to build and link your application

1. Building the application inside the VXG CLoud Agent library's `Meson project`, the app will be assembled during the library project compilation in this case.
   You need to add a new executable target into the main `meson.build` file, please refer to the example app build target declaration:
   ```
   cloud_agent_minimal = executable('cloud-agent-minimal', 'src/cloud-agent-minimal.cc',
     install : true, dependencies: dep)
   ```
   User must declare own executable target with a list of sources and dependencies, user may need to declare own dependencies if application requires it.

   This method is not recommended as it makes updating of the VXG Cloud Agent library mostly not possible or very difficult for application developer

2. Building your app using your own build system and linking against the installed library.
   Running the `install` step from the compile section installs the binary libraries and headers into the directory you specified during the `setup` step, it also puts the `pkg-config`'s `.pc` files into the prefix directory which could be used by your own build system.

3. Preferred and recommended way of application development is to hold the app as a separate `Meson project` and use the VXG Cloud Agent library as a `Meson subproject` of the application's `Meson project`.
   Using this approach gives the most flexible and convenient workflow for updating the VXG Cloud Library, all library dependencies will be promoted to the main project and will be also accessible by the application.
   How does it work

   - Assuming you have a `Meson` build system installed
   - Start a new `Meson project` with a following command:
     ```
     meson init -l cpp -n your-project-name
     ```

- As a result of this command you should have the following files tree:
```
|-- meson.build
|-- your_project_name.cpp
```
- Add VXG Cloud Agent library as a `Meson subproject`
  All subprojects should be located in the `subprojects` directory so you have to create it first
```
mkdir subprojects
```

Now you have 2 options depending on how you want to store the VXG Cloud Agent library sources:

(a) If you want to store the VXG Cloud Agent library as a files tree locally.

- Create a symlink to the library path inside the subprojects dir:
```
ln -s path/to/vxgcloudagent subprojects/vxgcloudagent
```

Or you can just move vxgcloudagent directory inside the subprojects dir.

- Create a library's `Meson` wrap file inside the subprojects dir, the name of the file should be the same as symlink you created in 1.1 and the content of the file should be:
```
[wrap-file]
directory = vxgcloudagent
[provide]
vxgcloudagent = vxgcloudagent_dep
```

(b) If you want to store the library in a git repository you just need to create a wrap file with the content like below:
```
[wrap-git]
url=https://your-git-repo-url.com/path/vxgcloudagent.git
# You can specify tag, branch or commit hash as revision
revision=master
[provide]
vxgcloudagent = vxgcloudagent_dep
```

You can find the example app `Meson project` in the example/app directory of the VXG Cloud library sources package.

Chapter 4

Library Compilation Guide

4.0.1 Library build process

Here is a compilation quickstart guide:

- First of all you need to have a build system and toolchain installed
- **Setup** the build directory
```
meson setup --prefix=path/to/install --strip -Dbuildtype=debug builddir/
# --prefix=path specifies the installation path
# --strip indicates that final binaries should be stripped
# -Dbuildtype= specifies the debug/release build type, please check the Meson docs about full list of
 the build types.
```
- **Build**
```
meson compile -C builddir
# Or
ninja -C builddir
```
- **Install**
```
meson install -C builddir
# Or
ninja -C builddir/ install
```
  As a result of the `install` step you should have the library compiled and installed into the prefix directory you specified during the `setup` step.
- **Clean**
```
ninja -C builddir clean
```
  Or you can just delete the builddir, you will need to `setup` it again in this case.
```
rm -rf builddir
```

4.0.2 Cross-compilation

- By default `Meson` builds project for the host platform, but it's also possible to cross-compile the library and your application using `Meson`.
- Full `Meson` cross-compilation documentation can be found `here`.
- The difference between the host compilation described above and the cross-compilation is the additional `--cross-file=path/to/cross-file.txt` flag for the Meson `Setup` step, the `Setup` command should look like below:
```
meson setup --prefix=path/to/install --strip -Dbuildtype=debug --cross-file=path/to/cross-file.txt
 builddir/
```
  `cross-file.txt` is the target platform description which in terms of Meson called a `cross-file`.

- `cross-file` example below is for the Debian provided `arm-linux-gnueabihf` toolchain installable using the Ubuntu's package manager command
  `sudo apt install g++-arm-linux-gnueabihf`

- Example of the ARMv7 `cross-file`:

```
[host_machine]
system = 'linux'
cpu_family = 'arm'
cpu = 'armv7-a'
endian = 'little'
[built-in options]
# Example of platform specific CFLAGS and CXXFLAGS
c_args = ['-mfloat-abi=hard', '-march=armv7-a+vfpv3']
cpp_args = c_args
default_library = 'static'
[properties]
# If your toolchain requires specifying the sysroot dir you can setup it like below, sysroot_dir is a
 constant declared in [constants] section of the cross-file
#sys_root = sysroot_dir
# Meson uses pkg-config and cmake to detect external dependencies
# Set the correct path to your cross-compilation pkgconfig directory if your app depends on some
 external dependencies like platform specific libs.
#pkg_config_libdir = sysroot_dir / 'usr/lib/pkgconfig/'
[constants]
cross_prefix = 'arm-linux-gnueabihf-'
#sysroot_dir = '/opt/arm-linux-gnueabihf/sysroot/'
[binaries]
c = cross_prefix + 'gcc'
cpp = cross_prefix + 'g++'
ar = cross_prefix + 'ar'
strip = cross_prefix + 'strip'
# You should specify your platform toolchain pkg-config binary here
#pkgconfig = '/opt/arm-linux-gnueabihf/bin/pkg-config'
```

Chapter 5

Deprecated List

Global vxg::logger::reset (int argc, char ∗∗argv, loglevel l, std::string syslog_ident="VXGCloudAgent↩ Default", std::string crash_logfile_path="", std::string logfile_path="", size_t logfile_max_size=(1024 ∗1024), size_t logfile_max_files=3)

Use reset(const options& opts)

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

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

Chapter 7

Data Structure Index

7.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 8

File Index

8.1 File List

Here is a list of all files with brief descriptions:

Chapter 9

Namespace Documentation

9.1 nlohmann Namespace Reference

9.2 std Namespace Reference

Namespaces

- **chrono**
- **experimental**
- **regex_constants**
- **rel_ops**
- **this_thread**

Data Structures

- class **add_const**
- class **add_cv**
- class **add_lvalue_reference**
- class **add_pointer**
- class **add_rvalue_reference**
- class **add_volatile**
- class **adopt_lock_t**
- class **aligned_storage**
- class **aligned_union**
- class **alignment_of**
- class **allocator**
- class **allocator_arg_t**
- class **allocator_traits**
- class **array**
- class **atomic**
- class **atomic_flag**
- class **auto_ptr**
- class **back_insert_iterator**
- class **bad_alloc**
- class **bad_array_length**
- class **bad_array_new_length**

- class **bad_cast**
- class **bad_exception**
- class **bad_function_call**
- class **bad_optional_access**
- class **bad_typeid**
- class **bad_weak_ptr**
- class **basic_filebuf**
- class **basic_fstream**
- class **basic_ifstream**
- class **basic_ios**
- class **basic_iostream**
- class **basic_istream**
- class **basic_istringstream**
- class **basic_ofstream**
- class **basic_ostream**
- class **basic_ostringstream**
- class **basic_regex**
- class **basic_streambuf**
- class **basic_string**
- class **basic_stringbuf**
- class **basic_stringstream**
- class **bernoulli_distribution**
- class **bidirectional_iterator_tag**
- class **binary_function**
- class **binary_negate**
- class **binomial_distribution**
- class **bit_and**
- class **bit_not**
- class **bit_or**
- class **bitset**
- class **cauchy_distribution**
- class **centi**
- class **cerr**
- class **char_traits**
- class **chi_squared_distribution**
- class **cin**
- class **clock_t**
- class **clog**
- class **cmatch**
- class **codecvt**
- class **codecvt_base**
- class **codecvt_byname**
- class **codecvt_utf16**
- class **codecvt_utf8**
- class **codecvt_utf8_utf16**
- class **collate**
- class **collate_byname**
- class **common_type**
- class **complex**
- class **condition_variable**
- class **condition_variable_any**
- class **conditional**
- class **cout**
- class **cregex_iterator**
- class **cregex_token_iterator**

- class **csub_match**
- class **ctype**
- class **ctype_base**
- class **ctype_byname**
- class **deca**
- class **decay**
- class **deci**
- class **default_delete**
- class **default_random_engine**
- class **defer_lock_t**
- class **deque**
- class **discard_block_engine**
- class **discrete_distribution**
- class **divides**
- class **domain_error**
- class **dynarray**
- class **enable_if**
- class **enable_shared_from_this**
- class **equal_to**
- class **errc**
- class **error_category**
- class **error_code**
- class **error_condition**
- class **exa**
- class **exception**
- class **exception_ptr**
- class **exponential_distribution**
- class **extent**
- class **extreme_value_distribution**
- class **false_type**
- class **femto**
- class **FILE**
- class **filebuf**
- class **fisher_f_distribution**
- class **forward_iterator_tag**
- class **forward_list**
- class **fpos**
- class **fpos_t**
- class **front_insert_iterator**
- class **fstream**
- class **function**
- class **future**
- class **future_error**
- class **gamma_distribution**
- class **geometric_distribution**
- class **giga**
- class **greater**
- class **greater_equal**
- class **has_virtual_destructor**
- class **hash**
- class **hecto**
- class **ifstream**
- class **independent_bits_engine**
- class **initializer_list**
- class **input_iterator_tag**

- class **insert_iterator**
- class **int16_t**
- class **int32_t**
- class **int64_t**
- class **int8_t**
- class **int_fast16_t**
- class **int_fast32_t**
- class **int_fast64_t**
- class **int_fast8_t**
- class **int_least16_t**
- class **int_least32_t**
- class **int_least64_t**
- class **int_least8_t**
- class **integer_sequence**
- class **integral_constant**
- class **intmax_t**
- class **intptr_t**
- class **invalid_argument**
- class **ios_base**
- class **iostream**
- class **is_abstract**
- class **is_arithmetic**
- class **is_array**
- class **is_assignable**
- class **is_base_of**
- class **is_bind_expression**
- class **is_class**
- class **is_compound**
- class **is_const**
- class **is_constructible**
- class **is_convertible**
- class **is_copy_assignable**
- class **is_copy_constructible**
- class **is_default_constructible**
- class **is_destructible**
- class **is_empty**
- class **is_enum**
- class **is_error_code_enum**
- class **is_error_condition_enum**
- class **is_floating_point**
- class **is_function**
- class **is_fundamental**
- class **is_integral**
- class **is_literal_type**
- class **is_lvalue_reference**
- class **is_member_function_pointer**
- class **is_member_object_pointer**
- class **is_member_pointer**
- class **is_move_assignable**
- class **is_move_constructible**
- class **is_nothrow_assignable**
- class **is_nothrow_constructible**
- class **is_nothrow_copy_assignable**
- class **is_nothrow_copy_constructible**
- class **is_nothrow_default_constructible**

- class **is_nothrow_destructible**
- class **is_nothrow_move_assignable**
- class **is_nothrow_move_constructible**
- class **is_object**
- class **is_placeholder**
- class **is_pod**
- class **is_pointer**
- class **is_polymorphic**
- class **is_reference**
- class **is_rvalue_reference**
- class **is_same**
- class **is_scalar**
- class **is_signed**
- class **is_standard_layout**
- class **is_trivial**
- class **is_trivially_assignable**
- class **is_trivially_constructible**
- class **is_trivially_copy_assignable**
- class **is_trivially_copy_constructible**
- class **is_trivially_copyable**
- class **is_trivially_default_constructible**
- class **is_trivially_destructible**
- class **is_trivially_move_assignable**
- class **is_trivially_move_constructible**
- class **is_union**
- class **is_unsigned**
- class **is_void**
- class **is_volatile**
- class **istream**
- class **istream_iterator**
- class **istreambuf_iterator**
- class **istringstream**
- class **istrstream**
- class **iterator**
- class **iterator_traits**
- class **jmp_buf**
- class **kilo**
- class **knuth_b**
- class **lconv**
- class **length_error**
- class **less**
- class **less_equal**
- class **linear_congruential_engine**
- class **list**
- class **locale**
- class **lock_guard**
- class **logic_error**
- class **logical_and**
- class **logical_not**
- class **logical_or**
- class **lognormal_distribution**
- class **make_signed**
- class **make_unsigned**
- class **map**
- class **match_results**

- class **max_align_t**
- class **mbstate_t**
- class **mega**
- class **mersenne_twister_engine**
- class **messages**
- class **messages_base**
- class **messages_byname**
- class **micro**
- class **milli**
- class **minstd_rand**
- class **minstd_rand0**
- class **minus**
- class **modulus**
- class **money_base**
- class **money_get**
- class **money_put**
- class **moneypunct**
- class **moneypunct_byname**
- class **move_iterator**
- class **mt19937**
- class **mt19937_64**
- class **multimap**
- class **multiplies**
- class **multiset**
- class **mutex**
- class **nano**
- class **negate**
- class **negative_binomial_distribution**
- class **nested_exception**
- class **new_handler**
- class **normal_distribution**
- class **not_equal_to**
- class **nothrow_t**
- class **nullptr_t**
- class **num_get**
- class **num_put**
- class **numeric_limits**
- class **numpunct**
- class **numpunct_byname**
- class **ofstream**
- class **once_flag**
- class **ostream**
- class **ostream_iterator**
- class **ostreambuf_iterator**
- class **ostringstream**
- class **ostrstream**
- class **out_of_range**
- class **output_iterator_tag**
- class **overflow_error**
- class **owner_less**
- class **packaged_task**
- class **pair**
- class **peta**
- class **pico**
- class **piecewise_constant_distribution**

- class **piecewise_construct_t**
- class **piecewise_linear_distribution**
- class **placeholders**
- class **plus**
- class **pointer_safety**
- class **pointer_traits**
- class **poisson_distribution**
- class **priority_queue**
- class **promise**
- class **ptrdiff_t**
- class **queue**
- class **random_access_iterator_tag**
- class **random_device**
- class **range_error**
- class **rank**
- class **ranlux24**
- class **ranlux24_base**
- class **ranlux48**
- class **ranlux48_base**
- class **ratio**
- class **ratio_add**
- class **ratio_divide**
- class **ratio_equal**
- class **ratio_greater**
- class **ratio_greater_equal**
- class **ratio_less**
- class **ratio_less_equal**
- class **ratio_multiply**
- class **ratio_not_equal**
- class **ratio_subtract**
- class **raw_storage_iterator**
- class **recursive_mutex**
- class **recursive_timed_mutex**
- class **reference_wrapper**
- class **regex**
- class **regex_error**
- class **regex_iterator**
- class **regex_token_iterator**
- class **regex_traits**
- class **remove_all_extents**
- class **remove_const**
- class **remove_cv**
- class **remove_extent**
- class **remove_pointer**
- class **remove_reference**
- class **remove_volatile**
- class **result_of**
- class **reverse_iterator**
- class **runtime_error**
- class **scoped_allocator_adaptor**
- class **seed_seq**
- class **set**
- class **shared_future**
- class **shared_lock**
- class **shared_ptr**

- class **shared_timed_mutex**
- class **shuffle_order_engine**
- class **sig_atomic_t**
- class **size_t**
- class **smatch**
- class **sregex_iterator**
- class **sregex_token_iterator**
- class **ssub_match**
- class **stack**
- class **streambuf**
- class **streamoff**
- class **streampos**
- class **streamsize**
- class **string**
- class **stringbuf**
- class **stringstream**
- class **strstream**
- class **strstreambuf**
- class **student_t_distribution**
- class **sub_match**
- class **subtract_with_carry_engine**
- class **system_error**
- class **tera**
- class **terminate_handler**
- class **thread**
- class **time_base**
- class **time_get**
- class **time_get_byname**
- class **time_put**
- class **time_put_byname**
- class **time_t**
- class **timed_mutex**
- class **tm**
- class **true_type**
- class **try_to_lock_t**
- class **tuple**
- class **type_index**
- class **type_info**
- class **u16streampos**
- class **u16string**
- class **u32streampos**
- class **u32string**
- class **uint16_t**
- class **uint32_t**
- class **uint64_t**
- class **uint8_t**
- class **uint_fast16_t**
- class **uint_fast32_t**
- class **uint_fast64_t**
- class **uint_fast8_t**
- class **uint_least16_t**
- class **uint_least32_t**
- class **uint_least64_t**
- class **uint_least8_t**
- class **uintmax_t**

- class **uintptr_t**
- class **unary_function**
- class **unary_negate**
- class **underflow_error**
- class **underlying_type**
- class **unexpected_handler**
- class **uniform_int_distribution**
- class **uniform_real_distribution**
- class **unique_lock**
- class **unique_ptr**
- class **unordered_map**
- class **unordered_multimap**
- class **unordered_multiset**
- class **unordered_set**
- class **uses_allocator**
- class **valarray**
- class **vector**
- class **wbuffer_convert**
- class **wcerr**
- class **wcin**
- class **wclog**
- class **wcmatch**
- class **wcout**
- class **wcregex_iterator**
- class **wcregex_token_iterator**
- class **wcsub_match**
- class **weak_ptr**
- class **weibull_distribution**
- class **wfilebuf**
- class **wfstream**
- class **wifstream**
- class **wiostream**
- class **wistream**
- class **wistringstream**
- class **wofstream**
- class **wostream**
- class **wostringstream**
- class **wregex**
- class **wsmatch**
- class **wsregex_iterator**
- class **wsregex_token_iterator**
- class **wssub_match**
- class **wstreambuf**
- class **wstreampos**
- class **wstring**
- class **wstring_convert**
- class **wstringbuf**
- class **wstringstream**
- class **yocto**
- class **yotta**
- class **zetta**

Functions

- T **atomic_fetch_and_explicit** (T... args)
- T **atomic_fetch_xor_explicit** (T... args)
- T **set_unexpected** (T... args)
- T **fputs** (T... args)
- T **modf** (T... args)
- T **not2** (T... args)
- T **strlen** (T... args)
- T **exp2** (T... args)
- T **setiosflags** (T... args)
- T **adjacent_difference** (T... args)
- T **cos** (T... args)
- T **fwscanf** (T... args)
- T **atomic_init** (T... args)
- T **forward_as_tuple** (T... args)
- T **abort** (T... args)
- T **wcsncmp** (T... args)
- T **set_intersection** (T... args)
- T **atomic_signal_fence** (T... args)
- T **llabs** (T... args)
- T **make_move_iterator** (T... args)
- T **scanf** (T... args)
- T **nextafter** (T... args)
- T **stol** (T... args)
- T **strcspn** (T... args)
- T **ungetwc** (T... args)
- T **transform** (T... args)
- T **putc** (T... args)
- T **iswdigit** (T... args)
- T **rint** (T... args)
- T **memset** (T... args)
- T **isgraph** (T... args)
- T **replace_copy_if** (T... args)
- T **scalbn** (T... args)
- T **partial_sort_copy** (T... args)
- T **make_exception_ptr** (T... args)
- T **frexp** (T... args)
- T **isxdigit** (T... args)
- T **atomic_exchange_explicit** (T... args)
- T **wprintf** (T... args)
- T **fdim** (T... args)
- T **wctype** (T... args)
- T **mbrtoc32** (T... args)
- T **setw** (T... args)
- T **get_temporary_buffer** (T... args)
- T **fmax** (T... args)
- T **atomic_thread_fence** (T... args)
- T **atomic_exchange** (T... args)
- T **fgetwc** (T... args)
- T **swprintf** (T... args)
- T **prev_permutation** (T... args)
- T **max_element** (T... args)
- T **set_symmetric_difference** (T... args)
- T **wcscpy** (T... args)

- T **const_pointer_cast** (T... args)
- T **minmax_element** (T... args)
- T **wcstok** (T... args)
- T **ref** (T... args)
- T **feupdateenv** (T... args)
- T **endl** (T... args)
- T **end** (T... args)
- T **wmemmove** (T... args)
- T **fmin** (T... args)
- T **uninitialized_fill_n** (T... args)
- T **nouppercase** (T... args)
- T **noshowpos** (T... args)
- T **ctime** (T... args)
- T **wmemset** (T... args)
- T **iswpunct** (T... args)
- T **pop_heap** (T... args)
- T **sprintf** (T... args)
- T **fixed** (T... args)
- T **make_shared** (T... args)
- T **make_heap** (T... args)
- T **fmod** (T... args)
- T **atol** (T... args)
- T **uninitialized_copy** (T... args)
- T **dynamic_pointer_cast** (T... args)
- T **set_union** (T... args)
- T **hexfloat** (T... args)
- T **vswprintf** (T... args)
- T **asctime** (T... args)
- T **iswspace** (T... args)
- T **nan** (T... args)
- T **sort** (T... args)
- T **quick_exit** (T... args)
- T **log10** (T... args)
- T **mbstowcs** (T... args)
- T **isspace** (T... args)
- T **strncat** (T... args)
- T **isinf** (T... args)
- T **atof** (T... args)
- T **erf** (T... args)
- T **is_sorted_until** (T... args)
- T **cbrt** (T... args)
- T **log1p** (T... args)
- T **return_temporary_buffer** (T... args)
- T **mbsrtowcs** (T... args)
- T **feraiseexcept** (T... args)
- T **fseek** (T... args)
- T **atomic_fetch_or_explicit** (T... args)
- T **log** (T... args)
- T **putchar** (T... args)
- T **make_tuple** (T... args)
- T **expm1** (T... args)
- T **fma** (T... args)
- T **remove_copy_if** (T... args)
- T **showpoint** (T... args)
- T **fscanf** (T... args)

- T **stable_partition** (T... args)
- T **fill_n** (T... args)
- T **remove_copy** (T... args)
- T **atomic_compare_exchange_strong_explicit** (T... args)
- T **wctomb** (T... args)
- T **fgets** (T... args)
- T **remainder** (T... args)
- T **allocate_shared** (T... args)
- T **unique** (T... args)
- T **includes** (T... args)
- T **iswalnum** (T... args)
- T **exit** (T... args)
- T **put_time** (T... args)
- T **to_string** (T... args)
- T **is_heap_until** (T... args)
- T **wcstold** (T... args)
- T **stold** (T... args)
- T **ftell** (T... args)
- T **copy_backward** (T... args)
- T **wcstoll** (T... args)
- T **perror** (T... args)
- T **vwscanf** (T... args)
- T **stable_sort** (T... args)
- T **generic_category** (T... args)
- T **abs(int)** (T... args)
- T **fgetws** (T... args)
- T **showpos** (T... args)
- T **exp** (T... args)
- T **fill** (T... args)
- T **isalpha** (T... args)
- T **lgamma** (T... args)
- T **feclearexcept** (T... args)
- T **wcsncpy** (T... args)
- T **undeclare_reachable** (T... args)
- T **oct** (T... args)
- T **strspn** (T... args)
- T **realloc** (T... args)
- T **copy** (T... args)
- T **binary_search** (T... args)
- T **system_category** (T... args)
- T **mbrtowc** (T... args)
- T **strtof** (T... args)
- T **mem_fn** (T... args)
- T **distance** (T... args)
- T **lock** (T... args)
- T **strcmp** (T... args)
- T **tmpfile** (T... args)
- T **hypot** (T... args)
- T **getenv** (T... args)
- T **strrchr** (T... args)
- T **count** (T... args)
- T **tan** (T... args)
- T **strftime** (T... args)
- T **stod** (T... args)
- T **towupper** (T... args)

- T **atoll** (T... args)
- T **atomic_store** (T... args)
- T **stoi** (T... args)
- T **rethrow_exception** (T... args)
- T **sin** (T... args)
- T **atomic_fetch_sub_explicit** (T... args)
- T **unexpected** (T... args)
- T **mbtowc** (T... args)
- T **get_time** (T... args)
- T **partition** (T... args)
- T **next** (T... args)
- T **isfinite** (T... args)
- T **boolalpha** (T... args)
- T **fetestexcept** (T... args)
- T **mbrlen** (T... args)
- T **iswgraph** (T... args)
- T **time** (T... args)
- T **atomic_compare_exchange_strong** (T... args)
- T **wcschr** (T... args)
- T **uppercase** (T... args)
- T **lower_bound** (T... args)
- T **copy_if** (T... args)
- T **isnan** (T... args)
- T **has_facet** (T... args)
- T **kill_dependency** (T... args)
- T **uninitialized_copy_n** (T... args)
- T **feholdexcept** (T... args)
- T **div** (T... args)
- T **at_quick_exit** (T... args)
- T **wcspbrk** (T... args)
- T **search** (T... args)
- T **find_first_of** (T... args)
- T **iota** (T... args)
- T **declare_reachable** (T... args)
- T **atomic_compare_exchange_weak** (T... args)
- T **strtod** (T... args)
- T **accumulate** (T... args)
- T **wcsrchr** (T... args)
- T **min_element** (T... args)
- T **clearerr** (T... args)
- T **random_shuffle** (T... args)
- T **iswalpha** (T... args)
- T **atomic_fetch_and** (T... args)
- T **wmemchr** (T... args)
- T **bsearch** (T... args)
- T **ilogb** (T... args)
- T **unique_copy** (T... args)
- T **_Exit** (T... args)
- T **move** (T... args)
- T **find_end** (T... args)
- T **fesetexceptflag** (T... args)
- T **nth_element** (T... args)
- T **gets** (T... args)
- T **lexicographical_compare** (T... args)
- T **nearbyint** (T... args)

- T **memcpy** (T... args)
- T **fwrite** (T... args)
- T **unitbuf** (T... args)
- T **iswlower** (T... args)
- T **mblen** (T... args)
- T **swscanf** (T... args)
- T **wcstoimax** (T... args)
- T **fprintf** (T... args)
- T **find_if** (T... args)
- T **strtoimax** (T... args)
- T **isalnum** (T... args)
- T **atomic_fetch_add_explicit** (T... args)
- T **push_heap** (T... args)
- T **min** (T... args)
- T **fwprintf** (T... args)
- T **uncaught_exception** (T... args)
- T **strtoll** (T... args)
- T **throw_with_nested** (T... args)
- T **shuffle** (T... args)
- T **isprint** (T... args)
- T **get_new_handler** (T... args)
- T **call_once** (T... args)
- T **trunc** (T... args)
- T **wcscspn** (T... args)
- T **mbrtoc16** (T... args)
- T **lround** (T... args)
- T **pow** (T... args)
- T **tgamma** (T... args)
- T **erfc** (T... args)
- T **llround** (T... args)
- T **abs(float)** (T... args)
- T **asinh** (T... args)
- T **feof** (T... args)
- T **noskipws** (T... args)
- T **find** (T... args)
- T **atoi** (T... args)
- T **not1** (T... args)
- T **vfscanf** (T... args)
- T **stof** (T... args)
- T **regex_search** (T... args)
- T **rotate_copy** (T... args)
- T **set_new_handler** (T... args)
- T **undeclare_no_pointers** (T... args)
- T **async** (T... args)
- T **partition_point** (T... args)
- T **vsscanf** (T... args)
- T **fesetround** (T... args)
- T **atomic_is_lock_free** (T... args)
- T **tanh** (T... args)
- T **ldiv** (T... args)
- T **setbase** (T... args)
- T **remove** (T... args)
- T **strtol** (T... args)
- T **strpbrk** (T... args)
- T **signbit** (T... args)

- T **wcsncat** (T... args)
- T **get_money** (T... args)
- T **set_difference** (T... args)
- T **cref** (T... args)
- T **getline** (T... args)
- T **to_wstring** (T... args)
- T **system** (T... args)
- T **static_pointer_cast** (T... args)
- T **wcstoumax** (T... args)
- T **memmove** (T... args)
- T **getwchar** (T... args)
- T **scientific** (T... args)
- T **wcsftime** (T... args)
- T **begin** (T... args)
- T **ceil** (T... args)
- T **sinh** (T... args)
- T **is_permutation** (T... args)
- T **generate_n** (T... args)
- T **acosh** (T... args)
- T **advance** (T... args)
- T **flush** (T... args)
- T **atomic_fetch_xor** (T... args)
- T **ws** (T... args)
- T **signal** (T... args)
- T **noshowbase** (T... args)
- T **generate** (T... args)
- T **ldexp** (T... args)
- T **vsnprintf** (T... args)
- T **remove_if** (T... args)
- T **stoull** (T... args)
- T **fegetexceptflag** (T... args)
- T **find_if_not** (T... args)
- T **merge** (T... args)
- T **free** (T... args)
- T **count_if** (T... args)
- T **clock** (T... args)
- T **mktime** (T... args)
- T **inserter** (T... args)
- T **puts** (T... args)
- T **asin** (T... args)
- T **iscntrl** (T... args)
- T **difftime** (T... args)
- T **terminate** (T... args)
- T **memcmp** (T... args)
- T **uninitialized_fill** (T... args)
- T **hex** (T... args)
- T **tie** (T... args)
- T **back_inserter** (T... args)
- T **upper_bound** (T... args)
- T **adjacent_find** (T... args)
- T **use_facet** (T... args)
- T **vfwprintf** (T... args)
- T **atomic_fetch_add** (T... args)
- T **fsetpos** (T... args)
- T **malloc** (T... args)

- T **localtime** (T... args)
- T **wcscmp** (T... args)
- T **c32rtomb** (T... args)
- T **isupper** (T... args)
- T **wcstod** (T... args)
- T **tolower** (T... args)
- T **sort_heap** (T... args)
- T **isdigit** (T... args)
- T **wcslen** (T... args)
- T **wmemcmp** (T... args)
- T **move_if_noexcept** (T... args)
- T **declval** (T... args)
- T **fpclassify** (T... args)
- T **iswupper** (T... args)
- T **rand** (T... args)
- T **atomic_compare_exchange_weak_explicit** (T... args)
- T **partial_sort** (T... args)
- T **llrint** (T... args)
- T **fclose** (T... args)
- T **reverse** (T... args)
- T **partial_sum** (T... args)
- T **showbase** (T... args)
- T **vswscanf** (T... args)
- T **atan** (T... args)
- T **atanh** (T... args)
- T **iter_swap** (T... args)
- T **scalbln** (T... args)
- T **reverse_copy** (T... args)
- T **forward** (T... args)
- T **getc** (T... args)
- T **equal_range** (T... args)
- T **atomic_fetch_sub** (T... args)
- T **is_partitioned** (T... args)
- T **next_permutation** (T... args)
- T **isblank** (T... args)
- T **noshowpoint** (T... args)
- T **atan2** (T... args)
- T **nanf** (T... args)
- T **towctrans** (T... args)
- T **right** (T... args)
- T **fputwc** (T... args)
- T **strtoul** (T... args)
- T **is_heap** (T... args)
- T **fflush** (T... args)
- T **strtoumax** (T... args)
- T **nexttoward** (T... args)
- T **nounitbuf** (T... args)
- T **ispunct** (T... args)
- T **noboolalpha** (T... args)
- T **make_pair** (T... args)
- T **iswctype** (T... args)
- T **srand** (T... args)
- T **replace_copy** (T... args)
- T **future_category** (T... args)
- T **resetiosflags** (T... args)

- T **vprintf** (T... args)
- T **gmtime** (T... args)
- T **align** (T... args)
- T **tuple_cat** (T... args)
- T **ends** (T... args)
- T **set_terminate** (T... args)
- T **lrint** (T... args)
- T **none_of** (T... args)
- T **wscanf** (T... args)
- T **fputc** (T... args)
- T **dec** (T... args)
- T **strcat** (T... args)
- T **raise** (T... args)
- T **wcsspn** (T... args)
- T **fabs** (T... args)
- T **wmemcpy** (T... args)
- T **copy_n** (T... args)
- T **rethrow_if_nested** (T... args)
- T **setlocale** (T... args)
- T **addressof** (T... args)
- T **calloc** (T... args)
- T **strerror** (T... args)
- T **strcpy** (T... args)
- T **wcstoull** (T... args)
- T **c16rtomb** (T... args)
- T **generate_canonical** (T... args)
- T **vfprintf** (T... args)
- T **notify_all_at_thread_exit** (T... args)
- T **rotate** (T... args)
- T **current_exception** (T... args)
- T **strtok** (T... args)
- T **wcscat** (T... args)
- T **strncpy** (T... args)
- T **towlower** (T... args)
- T **floor** (T... args)
- T **left** (T... args)
- T **ferror** (T... args)
- T **atomic_load_explicit** (T... args)
- T **swap** (T... args)
- T **acos** (T... args)
- T **wcscoll** (T... args)
- T **sqrt** (T... args)
- T **mbsinit** (T... args)
- T **qsort** (T... args)
- T **stoll** (T... args)
- T **put_money** (T... args)
- T **wcstoul** (T... args)
- T **wcstol** (T... args)
- T **atexit** (T... args)
- T **atomic_fetch_or** (T... args)
- T **rewind** (T... args)
- T **wcsxfrm** (T... args)
- T **round** (T... args)
- T **vwprintf** (T... args)
- T **all_of** (T... args)

- T **replace** (T... args)
- T **remquo** (T... args)
- T **setbuf** (T... args)
- T **strncmp** (T... args)
- T **localeconv** (T... args)
- T **wctrans** (T... args)
- T **any_of** (T... args)
- T **equal** (T... args)
- T **max** (T... args)
- T **strxfrm** (T... args)
- T **iswxdigit** (T... args)
- T **labs** (T... args)
- T **regex_match** (T... args)
- T **fputws** (T... args)
- T **wcrtomb** (T... args)
- T **setprecision** (T... args)
- T **setvbuf** (T... args)
- T **regex_replace** (T... args)
- T **freopen** (T... args)
- T **logb** (T... args)
- T **wctob** (T... args)
- T **atomic_load** (T... args)
- T **search_n** (T... args)
- T **toupper** (T... args)
- T **move_backward** (T... args)
- T **is_sorted** (T... args)
- T **strtoull** (T... args)
- T **iswblank** (T... args)
- T **get_pointer_safety** (T... args)
- T **get_unexpected** (T... args)
- T **sscanf** (T... args)
- T **fesetenv** (T... args)
- T **atomic_store_explicit** (T... args)
- T **strtold** (T... args)
- T **fread** (T... args)
- T **memchr** (T... args)
- T **btowc** (T... args)
- T **replace_if** (T... args)
- T **strcoll** (T... args)
- T **vsprintf** (T... args)
- T **mismatch** (T... args)
- T **getchar** (T... args)
- T **islower** (T... args)
- T **tmpnam** (T... args)
- T **nanl** (T... args)
- T **fopen** (T... args)
- T **for_each** (T... args)
- T **fegetround** (T... args)
- T **ungetc** (T... args)
- T **internal** (T... args)
- T **vfwscanf** (T... args)
- T **fgetc** (T... args)
- T **wcstof** (T... args)
- T **bind** (T... args)
- T **skipws** (T... args)

- T **iswprint** (T... args)
- T **wcstombs** (T... args)
- T **inplace_merge** (T... args)
- T **copysign** (T... args)
- T **putwchar** (T... args)
- T **wcsstr** (T... args)
- T **fegetenv** (T... args)
- T **longjmp** (T... args)
- T **iswcntrl** (T... args)
- T **declare_no_pointers** (T... args)
- T **isnormal** (T... args)
- T **swap_ranges** (T... args)
- T **minmax** (T... args)
- T **defaultfloat** (T... args)
- T **rename** (T... args)
- T **snprintf** (T... args)
- T **try_lock** (T... args)
- T **stoul** (T... args)
- T **fgetpos** (T... args)
- T **partition_copy** (T... args)
- T **vscanf** (T... args)
- T **front_inserter** (T... args)
- T **get_terminate** (T... args)
- T **cosh** (T... args)
- T **prev** (T... args)
- T **strchr** (T... args)
- T **strstr** (T... args)
- T **printf** (T... args)
- T **setfill** (T... args)
- T **inner_product** (T... args)
- template<typename T , typename... CONSTRUCTOR_ARGS>
  std::unique_ptr< T > make_unique (CONSTRUCTOR_ARGS &&... constructor_args)

9.2.1 Function Documentation

9.2.1.1 make_unique()

```
template<typename T , typename...  CONSTRUCTOR_ARGS>
std::unique_ptr<T> std::make_unique (
            CONSTRUCTOR_ARGS &&...  constructor_args )
```

Definition at line 192 of file utils.h.

9.3 vxg Namespace Reference

Namespaces

- cloud
- media

Data Structures

- class logger

  Logger class, current implementation based on spdlog.

9.4 vxg::cloud Namespace Reference

Namespaces

- agent

  VXG Cloud Agent namespace.
- time_spec

  time point
- utils

Typedefs

- using time = **std::chrono::time_point**< **std::chrono::system_clock**, time_spec::precision >
- using duration = time_spec::duration< time_spec::precision >

9.4.1 Typedef Documentation

9.4.1.1 duration

```
typedef time_spec::duration< time_spec::precision > vxg::cloud::duration
```

Definition at line 43 of file config.h.

9.4.1.2 time

```
typedef std::chrono::time_point< std::chrono::system_clock, time_spec::precision > vxg::cloud::time
```

Definition at line 42 of file config.h.

9.5 vxg::cloud::agent Namespace Reference

VXG Cloud Agent namespace.

Namespaces

- media
- proto

Data Structures

- struct access_token

  VXG Cloud access token.
- struct audio_config

  Audio config.
- class callback

  VXG Cloud manager common callbacks class.
- struct event_config

  Event config.
- class event_stream

  Event stream, abstract class for event generation.
- struct events_config

  Events config, list of event_config objects.
- class manager

  VXG Cloud agent manager class.
- struct osd_config

  OSD config.
- struct ptz_command

  PTZ command.
- struct ptz_config

  PTZ config.
- struct ptz_preset

  PTZ preset.
- struct supported_stream_config

  Supported stream config.
- struct supported_streams_config

  Supported streams config, list of supported_stream_config.

Functions

- **std::string** version ()

  VXG Cloud Agent library version.

9.5.1 Detailed Description

VXG Cloud Agent namespace.

9.5.2 Function Documentation

9.5.2.1 version()

```
std::string vxg::cloud::agent::version ( )
```

VXG Cloud Agent library version.

Returns

std::string version string

9.6 vxg::cloud::agent::media Namespace Reference

Data Structures

- class rtsp_stream

  Implementation of the media::stream with RTSP source and NIY stubs.
- class stream

  Cloud agent media stream abstract class.

9.7 vxg::cloud::agent::proto Namespace Reference

Data Structures

- struct audio_caps

  Audio capabilities.
- struct audio_stream_config

  Audio media stream config.
- struct event_caps

  Events capabilies.
- struct motion_detection_caps

  Motion detection capabilities camera capabilities that limit possible motion detection configuration.
- struct motion_detection_config

  Motion detection config.
- struct motion_region

  Motion detection related structs.
- struct osd_caps

  OSD capabilities.
- struct stream_caps

  Media stream capabilites.
- struct stream_config

  Media stream config.
- struct video_caps

  Video image capabilities.
- struct video_clip_info

  Video recoding(mp4 file) clip description,.
- struct video_config

  Video image config.
- struct video_stream_config

  Video stream config.
- struct wifi_config

  WiFi config.
- struct wifi_network

  WiFi network object.

Typedefs

- typedef wifi_config wifi_list

  wifi_config

Enumerations

- enum mode { M_OFF, M_ON, M_AUTO, M_INVALID }

  Mode on/off.
- enum video_format { VF_H264, VF_H265, VF_MJPEG, VF_INVALID }

  Video codec format.
- enum audio_format {
  AF_G711A, AF_G711U, AF_RAW, AF_ADPCM,
  AF_MP3, AF_NELLY8, AF_NELLY16, AF_NELLY,
  AF_OPUS, AF_AAC, AF_SPEEX, AF_INVALID }

  Audio codec format.
- enum audio_file_format { AFF_AU_G711U, AFF_MP3, AFF_WAV_PCM, AFF_INVALID }

  Audio file format.
- enum motion_sensitivity { MS_REGION, MS_FRAME, MS_INVALID }

  Motion sensitivity.
- enum motion_region_shape { MR_RECTANGLE, MR_ANY, MR_INVALID }

  Motion region shape.
- enum ptz_action {
  A_LEFT, A_RIGHT, A_TOP, A_BOTTOM,
  A_ZOOM_IN, A_ZOOM_OUT, A_STOP, A_INVALID }

  PTZ actions.
- enum ptz_preset_action {
  PA_CREATE, PA_DELETE, PA_GOTO, PA_UPDATE,
  PA_INVALID }

  PTZ preset action.
- enum time_format_n { TF_12H, TF_24H, TF_INVALID }

  3.34 get_osd_conf (SRV) 3.35 osd_conf (CM) 3.36 set_osd_conf (SRV)
- enum event_status { ES_OK, ES_ERROR, ES_INVALID }

  Event status.
- enum event_type {
  ET_MOTION, ET_SOUND, ET_NET, ET_RECORD,
  ET_MEMORYCARD, ET_WIFI, ET_CUSTOM, ET_INVALID }

  Types of events.
- enum memorycard_status {
  MCS_NONE, MCS_NORMAL, MCS_NEED_FORMAT, MCS_FORMATTING,
  MCS_INITIALIZATION, MCS_INVALID }

  Memory card status.
- enum wifi_encryption {
  WFE_OPEN, WFE_WEP, WFE_WPA, WFE_WPA2,
  WFE_WPA_ENTERPRISE, WFE_WPA2_ENTERPRISE, WFE_INVALID }

  WiFi encryption type.
- enum wifi_network_state {
  WNS_UNKNOWN, WNS_INITIALIZE_0, WNS_INITIALIZE_1, WNS_TRY_CONNECT,
  WNS_RECEIVING_IP, WNS_CONNECTED, WNS_INVALID }

  WiFi connection state.

Functions

- **std::string** name ()

9.7.1 Typedef Documentation

9.7.1.1 wifi_list

```
typedef wifi_config vxg::cloud::agent::proto::wifi_list
```

wifi_config

Definition at line 597 of file config.h.

9.7.2 Enumeration Type Documentation

9.7.2.1 audio_file_format

```
enum vxg::cloud::agent::proto::audio_file_format
```

Audio file format.

Enumerator

AFF_AU_G711U	AU file format, encoded in mu-law and sampled with 8 or 16 kHz;.
AFF_MP3	MP3 file format, in mono or stereo with bitrate of 64 kbps to 320 kbps and sample rate of 8 to 48 kHz.
AFF_WAV_PCM	WAV file format, encoded in PCM audio that depends on what the product supports. It may support encoded as 8 or 16-bit mono or stereo and sample rate of 8 to 48 kHz;
AFF_INVALID	Invalid value.

Definition at line 147 of file caps.h.

9.7.2.2 audio_format

```
enum vxg::cloud::agent::proto::audio_format
```

Audio codec format.

Enumerator

AF_G711A	G711A - PCMA, A-Law.
AF_G711U	G711U - PCMU, U-Law.
AF_RAW	PCM.
AF_ADPCM	G726LE.
AF_MP3	
AF_NELLY8	
AF_NELLY16	
AF_NELLY	
AF_OPUS	
AF_AAC	AAC.
AF_SPEEX	
AF_INVALID	Invalid value.

Definition at line 106 of file caps.h.

9.7.2.3 event_status

`enum vxg::cloud::agent::proto::event_status`

Event status.

Enumerator

ES_OK	Ok.
ES_ERROR	Error.
ES_INVALID	Default status, invalid.

Definition at line 381 of file config.h.

9.7.2.4 event_type

`enum vxg::cloud::agent::proto::event_type`

Types of events.

Enumerator

ET_MOTION	"motion" for motion detection events
ET_SOUND	"sound" for audio detection
ET_NET	"net" for the camera network status change
ET_RECORD	"record" CM informs server about necessity of changing of recording state
ET_MEMORYCARD	"memorycard" camera's memory-card status change
ET_WIFI	"wifi" status of camera's currently used Wi-Fi
ET_CUSTOM	Custom event.
ET_INVALID	Invalid event type.

Definition at line 404 of file config.h.

9.7.2.5 memorycard_status

enum vxg::cloud::agent::proto::memorycard_status

Memory card status.

Enumerator

MCS_NONE	No memorycard.
MCS_NORMAL	Memorycard is OK.
MCS_NEED_FORMAT	Need formatting.
MCS_FORMATTING	Formatting ongoing.
MCS_INITIALIZATION	Initialization, not mounted yet for example.
MCS_INVALID	Invalid value.

Definition at line 484 of file config.h.

9.7.2.6 mode

enum vxg::cloud::agent::proto::mode

Mode on/off.

Enumerator

M_OFF	
M_ON	
M_AUTO	
M_INVALID	

Definition at line 30 of file caps.h.

9.7.2.7 motion_region_shape

enum vxg::cloud::agent::proto::motion_region_shape

Motion region shape.

Enumerator

MR_RECTANGLE	Rectangle.
MR_ANY	Any shape.
MR_INVALID	Invalid.

Definition at line 313 of file caps.h.

9.7.2.8 motion_sensitivity

enum vxg::cloud::agent::proto::motion_sensitivity

Motion sensitivity.

Enumerator

MS_REGION	Indicates if sensitivity can be set for region.
MS_FRAME	Indicates if sensitivity can be only for the full frame.
MS_INVALID	Invalid value.

Definition at line 291 of file caps.h.

9.7.2.9 ptz_action

enum vxg::cloud::agent::proto::ptz_action

PTZ actions.

Enumerator

A_LEFT	Go left.
A_RIGHT	Go right.
A_TOP	Go tip.
A_BOTTOM	Go bottom.
A_ZOOM_IN	Zoom in.
A_ZOOM_OUT	Zoom out.
A_STOP	Stop current action.
A_INVALID	Invalid value.

Definition at line 527 of file caps.h.

9.7.2.10 ptz_preset_action

enum vxg::cloud::agent::proto::ptz_preset_action

PTZ preset action.

Enumerator

PA_CREATE	
PA_DELETE	
PA_GOTO	
PA_UPDATE	
PA_INVALID	

Definition at line 563 of file caps.h.

9.7.2.11 time_format_n

`enum vxg::cloud::agent::proto::time_format_n`

3.34 get_osd_conf (SRV) 3.35 osd_conf (CM) 3.36 set_osd_conf (SRV)

Time format

Enumerator

TF_12H	12 hours
TF_24H	24 hours
TF_INVALID	Invalid value.

Definition at line 592 of file caps.h.

9.7.2.12 video_format

`enum vxg::cloud::agent::proto::video_format`

Video codec format.

Enumerator

VF_H264	H264 (AVC)
VF_H265	H265 (HEVC)
VF_MJPEG	Motion JPEG.
VF_INVALID	Invalid value.

Definition at line 81 of file caps.h.

9.7.2.13 wifi_encryption

`enum vxg::cloud::agent::proto::wifi_encryption`

WiFi encryption type.

Enumerator

WFE_OPEN	No encryption.
WFE_WEP	WEP.
WFE_WPA	WPA-PSK.
WFE_WPA2	WPA2-PSK.
WFE_WPA_ENTERPRISE	WPA-Enterprise.
WFE_WPA2_ENTERPRISE	WPA2-Enterprise.
WFE_INVALID	Default, invalid value.

Definition at line 520 of file config.h.

9.7.2.14 wifi_network_state

```
enum vxg::cloud::agent::proto::wifi_network_state
```

WiFi connection state.

Enumerator

WNS_UNKNOWN	
WNS_INITIALIZE_0	
WNS_INITIALIZE_1	
WNS_TRY_CONNECT	
WNS_RECEIVING_IP	
WNS_CONNECTED	
WNS_INVALID	Invalid value.

Definition at line 600 of file config.h.

9.7.3 Function Documentation

9.7.3.1 name()

```
std::string vxg::cloud::agent::proto::name ( )
```

Definition at line 887 of file config.h.

9.8 vxg::cloud::time_spec Namespace Reference

time point

Typedefs

- using precision = **std::chrono::nanoseconds**
- template<typename T >
  using duration = typename **std::conditional**< **std::is_same**< T, precision >::value, precision, **std**↩
  ::chrono::duration< T > >::type

9.8.1 Detailed Description

time point

9.8.2 Typedef Documentation

9.8.2.1 duration

```
template<typename T >
using vxg::cloud::time_spec::duration = typedef typename  std::conditional< std::is_same<T,
precision>::value, precision,  std::chrono::duration<T> >::type
```

Definition at line 39 of file config.h.

9.8.2.2 precision

```
typedef  std::chrono::nanoseconds vxg::cloud::time_spec::precision
```

Definition at line 35 of file config.h.

9.9 vxg::cloud::utils Namespace Reference

Namespaces

- gcc_abi
- motion
- time

Data Structures

- struct uri

Functions

- void set_thread_name (**std::string** name)
- template<typename... Args>
  std::string string_format (const **std::string** &format, Args... args)
- **std::string** string_trim (const **std::string** &name, **std::regex** regx)
- **std::string** string_trim (const **std::string** &name)
- **std::vector**< **std::string** > string_split (const **std::string** &s, char delimiter)
- bool string_startswith (**std::string** const &fullString, **std::string** const &start)
- bool string_endswith (**std::string** const &fullString, **std::string** const &ending)
- bool string_replace (**std::string** &str, const **std::string** &from, const **std::string** &to)
- **std::string** string_urlencode (const **std::string** &value)
- **std::string** string_urldecode (const **std::string** &text)
- **std::string** string_tolower (const **std::string** &s)
- **std::string** string_toupper (const **std::string** &s)
- bool string_contains (**std::string** s, char c)
- **std::string** dirname (const **std::string** &filepath)

9.9.1 Function Documentation

9.9.1.1 dirname()

```
std::string vxg::cloud::utils::dirname (
            const  std::string & filepath )
```

9.9.1.2 set_thread_name()

```
void vxg::cloud::utils::set_thread_name (
             std::string name )
```

9.9.1.3 string_contains()

```
bool vxg::cloud::utils::string_contains (
             std::string s,
            char c )  [inline]
```

Definition at line 172 of file utils.h.

9.9.1.4 string_endswith()

```
bool vxg::cloud::utils::string_endswith (
              std::string const & fullString,
              std::string const & ending )
```

9.9.1.5 string_format()

```
template<typename...  Args>
std::string vxg::cloud::utils::string_format (
             const  std::string & format,
             Args...  args )
```

Definition at line 149 of file utils.h.

9.9.1.6 string_replace()

```
bool vxg::cloud::utils::string_replace (
              std::string & str,
             const  std::string & from,
             const  std::string & to )
```

9.9.1.7 string_split()

```
 std::vector< std::string> vxg::cloud::utils::string_split (
             const  std::string & s,
             char delimiter )
```

9.9.1.8 string_startswith()

```
bool vxg::cloud::utils::string_startswith (
              std::string const & fullString,
              std::string const & start )
```

9.9.1.9 string_tolower()

```
 std::string vxg::cloud::utils::string_tolower (
             const  std::string & s )
```

9.9.1.10 string_toupper()

```
std::string vxg::cloud::utils::string_toupper (
            const  std::string & s )
```

9.9.1.11 string_trim() [1/2]

```
std::string vxg::cloud::utils::string_trim (
            const  std::string & name )
```

9.9.1.12 string_trim() [2/2]

```
std::string vxg::cloud::utils::string_trim (
            const  std::string & name,
             std::regex regx )
```

9.9.1.13 string_urldecode()

```
std::string vxg::cloud::utils::string_urldecode (
            const  std::string & text )
```

9.9.1.14 string_urlencode()

```
std::string vxg::cloud::utils::string_urlencode (
            const  std::string & value )
```

9.10 vxg::cloud::utils::gcc_abi Namespace Reference

Functions

- **std::string** demangle (**std::string** name)

9.10.1 Function Documentation

9.10.1.1 demangle()

```
std::string vxg::cloud::utils::gcc_abi::demangle (
            std::string name )
```

9.11 vxg::cloud::utils::motion Namespace Reference

Data Structures

- struct map

9.12 vxg::cloud::utils::time Namespace Reference

Functions

- cloud::time now ()
- **std::string** time_to_ISO8601 (**std::time_t**)
- **std::string** time_to_ISO8601_packed (**std::time_t**)
- **std::string** now_ISO8601_UTC ()
- **std::string** now_ISO8601_UTC_packed ()
- **std::time_t** now_time_UTC ()
- **std::time_t** ISO8601_to_time (const **std::string** &input)
- **std::string** to_iso_8601 (cloud::time t)
- **std::string** to_iso (cloud::time t)
- **std::string** to_iso2 (cloud::time t)
- **std::string** to_iso_packed (cloud::time t)
- **std::string** to_iso_local (cloud::time t)
- cloud::time from_double (double t)
- double to_double (cloud::time t)
- cloud::time from_iso (**std::string** st)
- cloud::time from_iso2 (**std::string** st)
- cloud::time from_iso_packed (**std::string** st)
- bool iso_time_valid (const **std::string** &s)
- cloud::time null ()
- cloud::time max ()
- bool is_iso_packed (const **std::string** &s)
- bool is_iso (const **std::string** &s)

9.12.1 Function Documentation

9.12.1.1 from_double()

```
cloud::time vxg::cloud::utils::time::from_double (
            double t )
```

9.12.1.2 from_iso()

```
cloud::time vxg::cloud::utils::time::from_iso (
              std::string st )
```

9.12.1.3 from_iso2()

```
cloud::time vxg::cloud::utils::time::from_iso2 (
              std::string st )
```

9.12.1.4 from_iso_packed()

```
cloud::time vxg::cloud::utils::time::from_iso_packed (
              std::string st )
```

9.12.1.5 is_iso()

```
bool vxg::cloud::utils::time::is_iso (
             const  std::string & s )
```

9.12.1.6 is_iso_packed()

```
bool vxg::cloud::utils::time::is_iso_packed (
             const  std::string & s )
```

9.12.1.7 ISO8601_to_time()

```
 std::time_t vxg::cloud::utils::time::ISO8601_to_time (
             const  std::string & input )
```

9.12.1.8 iso_time_valid()

```
bool vxg::cloud::utils::time::iso_time_valid (
             const  std::string & s )
```

9.12.1.9 max()

```
cloud::time vxg::cloud::utils::time::max ( )  [inline]
```

Definition at line 57 of file utils.h.

9.12.1.10 now()

```
cloud::time vxg::cloud::utils::time::now ( )  [inline]
```

Definition at line 30 of file utils.h.

9.12.1.11 now_ISO8601_UTC()

```
std::string vxg::cloud::utils::time::now_ISO8601_UTC ( )
```

9.12.1.12 now_ISO8601_UTC_packed()

```
std::string vxg::cloud::utils::time::now_ISO8601_UTC_packed ( )
```

9.12.1.13 now_time_UTC()

```
std::time_t vxg::cloud::utils::time::now_time_UTC ( )
```

9.12.1.14 null()

```
cloud::time vxg::cloud::utils::time::null ( )  [inline]
```

Definition at line 53 of file utils.h.

9.12.1.15 time_to_ISO8601()

```
std::string vxg::cloud::utils::time::time_to_ISO8601 (
            std::time_t  )
```

9.12.1.16 time_to_ISO8601_packed()

```
std::string vxg::cloud::utils::time::time_to_ISO8601_packed (
            std::time_t  )
```

9.12.1.17 to_double()

```
double vxg::cloud::utils::time::to_double (
            cloud::time t )
```

9.12.1.18 to_iso()

```
std::string vxg::cloud::utils::time::to_iso (
            cloud::time t )
```

9.12.1.19 to_iso2()

```
std::string vxg::cloud::utils::time::to_iso2 (
            cloud::time t )
```

9.12.1.20 to_iso_8601()

```
std::string vxg::cloud::utils::time::to_iso_8601 (
            cloud::time t )
```

9.12.1.21 to_iso_local()

```
std::string vxg::cloud::utils::time::to_iso_local (
            cloud::time t )
```

9.12.1.22 to_iso_packed()

```
std::string vxg::cloud::utils::time::to_iso_packed (
            cloud::time t )
```

9.13 vxg::media Namespace Reference

Namespaces

- ffmpeg
- Streamer

Data Structures

- class rtmp_sink

  RTMP sink class.
- class rtmp_source

  RTMP source class.
- class rtsp_source

  RTSP source class.
- class stream

  base media stream abstract class

9.14 vxg::media::ffmpeg Namespace Reference

Data Structures

- class Sink

  Base ffmpeg sink class.
- class Source

  Base ffmpeg source class.

9.15 vxg::media::Streamer Namespace Reference

Data Structures

- class ISink
- class ISource

  ISource interface class.
- struct MediaFrame

  Media frame container.
- struct StreamInfo

  Stream info descriotion.

Enumerations

- enum DropDirection { DROP_FRONT, DROP_BACK }
- enum StreamError { E_NONE, E_FATAL, E_EOS }

  Stream error.
- enum MediaType {
  UKNOWN, VIDEO, VIDEO_AVC_SPS, VIDEO_AVC_PPS,
  VIDEO_SEQ_HDR, AUDIO, AUDIO_SEQ_HDR, FLV,
  DATA, MAX }

  Media frame type.

Variables

- constexpr int SINK_THREAD_PRIO
- constexpr int SRC_THREAD_PRIO

9.15.1 Enumeration Type Documentation

9.15.1.1 DropDirection

enum vxg::media::Streamer::DropDirection

Enumerator

DROP_FRONT	
DROP_BACK	

Definition at line 27 of file base_streamer.h.

9.15.1.2 MediaType

enum vxg::media::Streamer::MediaType

Media frame type.

Used to indicate when type of frame was passed from source to sink.

Enumerator

UKNOWN	
VIDEO	
VIDEO_AVC_SPS	
VIDEO_AVC_PPS	
VIDEO_SEQ_HDR	
AUDIO	
AUDIO_SEQ_HDR	
FLV	
DATA	
MAX	

Definition at line 389 of file base_streamer.h.

9.15.1.3 StreamError

```
enum vxg::media::Streamer::StreamError
```

Stream error.

Enumerator

E_NONE	
E_FATAL	
E_EOS	

Definition at line 33 of file base_streamer.h.

9.15.2 Variable Documentation

9.15.2.1 SINK_THREAD_PRIO

```
constexpr int vxg::media::Streamer::SINK_THREAD_PRIO  [constexpr]
```

Definition at line 25 of file base_streamer.h.

9.15.2.2 SRC_THREAD_PRIO

```
constexpr int vxg::media::Streamer::SRC_THREAD_PRIO  [constexpr]
```

Definition at line 26 of file base_streamer.h.

Chapter 10

Data Structure Documentation

10.1 vxg::cloud::agent::access_token Struct Reference

VXG Cloud access token.

`#include <agent-proto/objects/config.h>`

Data Structures

- struct proxy_config
  Socks proxy settings.

Public Types

- typedef **std::shared_ptr**< access_token > ptr

Public Member Functions

- **std::string** api_uri (bool secure=true)
- **std::string** pack ()

Static Public Member Functions

- static access_token::ptr parse (**std::string** packed_token)

10.1.1 Detailed Description

VXG Cloud access token.

Definition at line 1192 of file config.h.

10.1.2 Member Typedef Documentation

10.1.2.1 ptr

```
typedef  std::shared_ptr<access_token> vxg::cloud::agent::access_token::ptr
```

Definition at line 1193 of file config.h.

10.1.3 Member Function Documentation

10.1.3.1 api_uri()

```
std::string vxg::cloud::agent::access_token::api_uri (
            bool secure = true )  [inline]
```

Definition at line 1242 of file config.h.

10.1.3.2 pack()

```
std::string vxg::cloud::agent::access_token::pack ( )  [inline]
```

Definition at line 1250 of file config.h.

10.1.3.3 parse()

```
static access_token::ptr vxg::cloud::agent::access_token::parse (
              std::string packed_token )  [inline], [static]
```

Definition at line 1252 of file config.h.

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

- config.h

10.2 alter_bool Struct Reference

alternative bool class Standard bool type has two states, this class adds 3rd state - undefined.

```
#include <agent-proto/command/unset-helper.h>
```

Public Types

- enum n_alter_bool { B_FALSE, B_TRUE, B_INVALID }

  Internal boolean values.

Public Member Functions

- alter_bool (const n_alter_bool &v)
- alter_bool (const bool &v)
- alter_bool operator= (const bool &b)
- operator bool () const

Data Fields

- n_alter_bool val

Friends

- void from_json (const nlohmann::json &j, alter_bool &c)
- void to_json (nlohmann::json &j, const alter_bool &c)

10.2.1 Detailed Description

alternative bool class Standard bool type has two states, this class adds 3rd state - undefined.

This class used for json boolean => C++ bool type reflection. The B_INVALID value of the C++ data indicates that source json has no such field.

Definition at line 168 of file unset-helper.h.

10.2.2 Member Enumeration Documentation

10.2.2.1 n_alter_bool

```
enum alter_bool::n_alter_bool
```

Internal boolean values.

Enumerator

B_FALSE	false
B_TRUE	true
B_INVALID	Undefined, i.e. if the object was constructed from the json object this value means that original json had no such field.

Definition at line 170 of file unset-helper.h.

10.2.3 Constructor & Destructor Documentation

10.2.3.1 alter_bool() [1/2]

```
alter_bool::alter_bool (
            const n_alter_bool & v )  [inline]
```

Definition at line 180 of file unset-helper.h.

10.2.3.2 alter_bool() [2/2]

```
alter_bool::alter_bool (
            const bool & v )  [inline]
```

Definition at line 182 of file unset-helper.h.

10.2.4 Member Function Documentation

10.2.4.1 operator bool()

```
alter_bool::operator bool ( ) const  [inline]
```

Definition at line 196 of file unset-helper.h.

10.2.4.2 operator=()

```
alter_bool alter_bool::operator= (
            const bool & b )  [inline]
```

Definition at line 189 of file unset-helper.h.

10.2.5 Friends And Related Function Documentation

10.2.5.1 from_json

```
void from_json (
            const nlohmann::json & j,
            alter_bool & c )  [friend]
```

Definition at line 202 of file unset-helper.h.

10.2.5.2 to_json

```
void to_json (
            nlohmann::json & j,
            const alter_bool & c )  [friend]
```

Definition at line 209 of file unset-helper.h.

10.2.6 Field Documentation

10.2.6.1 val

```
n_alter_bool alter_bool::val
```

Definition at line 216 of file unset-helper.h.

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

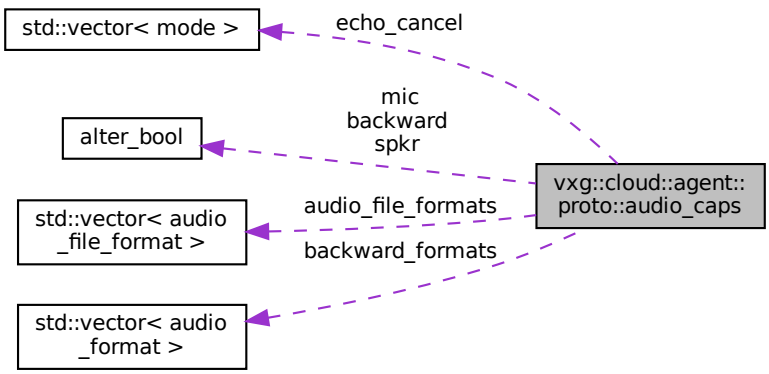
- unset-helper.h

10.3 vxg::cloud::agent::proto::audio_caps Struct Reference

Audio capabilities.

`#include <agent-proto/objects/caps.h>`

Collaboration diagram for vxg::cloud::agent::proto::audio_caps:



Data Fields

- alter_bool mic

  mic: bool, microphone is supported
- alter_bool spkr

  spkr: bool, speaker is supported
- **std::vector**< mode > echo_cancel

  echo_cancel: list of string, echo cancellation modes, empty or absent means not supported
- alter_bool backward

  backward: bool, backward audio supported.
- **std::vector**< audio_format > backward_formats

  backward_formats: list of audio_format, list of supported backward formats.
- **std::vector**< audio_file_format > audio_file_formats

  audio_file_formats: list of string, list of supported formats of audio files.

10.3.1 Detailed Description

Audio capabilities.

Definition at line 484 of file caps.h.

10.3.2 Field Documentation

10.3.2.1 audio_file_formats

```
std::vector<audio_file_format> vxg::cloud::agent::proto::audio_caps::audio_file_formats
```

audio_file_formats: list of string, list of supported formats of audio files.

Definition at line 507 of file caps.h.

10.3.2.2 backward

```
alter_bool vxg::cloud::agent::proto::audio_caps::backward
```

backward: bool, backward audio supported.

Obsolete. Server will ignore it when backward_formats exists. If true and backward_formats is missed, server will interpret supported formats list as [“UNKNOWN”]

Definition at line 497 of file caps.h.

10.3.2.3 backward_formats

```
std::vector<audio_format> vxg::cloud::agent::proto::audio_caps::backward_formats
```

backward_formats: list of audio_format, list of supported backward formats.

Supported values: [“RAW”, “ADPCM”, “MP3”, “NELLY8”, “NELLY16”, “NELLY”, “G711A”, “G711U”, “AAC”, “SPEEX”, “UNKNOWN”]. Empty list or missing parameter – camera doesn't support back audio channel.

Definition at line 503 of file caps.h.

10.3.2.4 echo_cancel

```
std::vector<mode> vxg::cloud::agent::proto::audio_caps::echo_cancel
```

echo_cancel: list of string, echo cancellation modes, empty or absent means not supported

Definition at line 492 of file caps.h.

10.3.2.5 mic

```
alter_bool vxg::cloud::agent::proto::audio_caps::mic
```

mic: bool, microphone is supported

Definition at line 486 of file caps.h.

10.3.2.6 spkr

```
alter_bool vxg::cloud::agent::proto::audio_caps::spkr
```

spkr: bool, speaker is supported

Definition at line 489 of file caps.h.

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

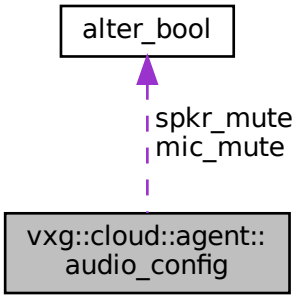
- caps.h

10.4 vxg::cloud::agent::audio_config Struct Reference

Audio config.

`#include <agent-proto/objects/config.h>`

Collaboration diagram for vxg::cloud::agent::audio_config:



Data Fields

- int mic_gain

  mic_gain: optional int range 0-100, microphone gain
- alter_bool mic_mute

  mic_mute: optional bool, microphone mute
- int spkr_vol

  spkr_vol: optional int range 0-100, speaker volume
- alter_bool spkr_mute

  spkr_mute: optional bool, speaker mute
- mode echo_cancel

  echo_cancel: optional string, echo cancellation mode, "" means off
- audio_caps caps

  caps

10.4.1 Detailed Description

Audio config.

Definition at line 1036 of file config.h.

10.4.2 Field Documentation

10.4.2.1 caps

```
audio_caps vxg::cloud::agent::audio_config::caps
```

caps

Definition at line 1049 of file config.h.

10.4.2.2 echo_cancel

```
mode vxg::cloud::agent::audio_config::echo_cancel
```

echo_cancel: optional string, echo cancellation mode, “” means off

Definition at line 1046 of file config.h.

10.4.2.3 mic_gain

```
int vxg::cloud::agent::audio_config::mic_gain
```

mic_gain: optional int range 0-100, microphone gain

Definition at line 1038 of file config.h.

10.4.2.4 mic_mute

```
alter_bool vxg::cloud::agent::audio_config::mic_mute
```

mic_mute: optional bool, microphone mute

Definition at line 1040 of file config.h.

10.4.2.5 spkr_mute

```
alter_bool vxg::cloud::agent::audio_config::spkr_mute
```

spkr_mute: optional bool, speaker mute

Definition at line 1044 of file config.h.

10.4.2.6 spkr_vol

```
int vxg::cloud::agent::audio_config::spkr_vol
```

spkr_vol: optional int range 0-100, speaker volume

Definition at line 1042 of file config.h.

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

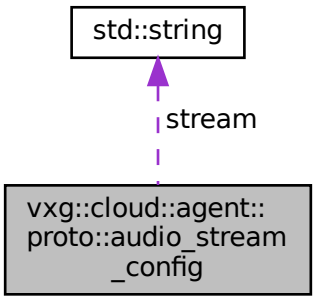
- config.h

10.5 vxg::cloud::agent::proto::audio_stream_config Struct Reference

Audio media stream config.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::proto::audio_stream_config:



Data Fields

- **std::string** stream

  Mandatory: audio ES to use.
- audio_format format

  Mandatory: audio encoding format.
- int brt

  Mandatory: bitrate, kbps.
- double srt

  Mandatory: samplerate, KHz.

10.5.1 Detailed Description

Audio media stream config.

Definition at line 182 of file config.h.

10.5.2 Field Documentation

10.5.2.1 brt

```
int vxg::cloud::agent::proto::audio_stream_config::brt
```

Mandatory: bitrate, kbps.

Definition at line 193 of file config.h.

10.5.2.2 format

```
audio_format vxg::cloud::agent::proto::audio_stream_config::format
```

Mandatory: audio encoding format.

Definition at line 189 of file config.h.

10.5.2.3 srt

```
double vxg::cloud::agent::proto::audio_stream_config::srt
```

Mandatory: samplerate, KHz.

Definition at line 197 of file config.h.

10.5.2.4 stream

```
std::string vxg::cloud::agent::proto::audio_stream_config::stream
```

Mandatory: audio ES to use.

Definition at line 185 of file config.h.

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

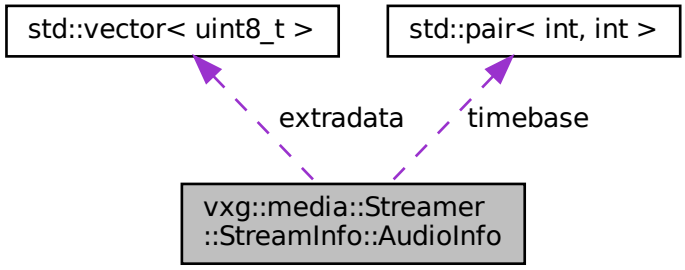
- config.h

10.6 vxg::media::Streamer::StreamInfo::AudioInfo Struct Reference

Audio stream info.

`#include <streamer/base_streamer.h>`

Collaboration diagram for vxg::media::Streamer::StreamInfo::AudioInfo:



Data Fields

- AudioCodec codec
  Audio codec.
- int channels
  Audio channels.
- int samplerate
  Audio samplerate.
- int bitrate
  Audio bitrate.
- **std::pair**< int, int > timebase
  Audio timestamps timescale.
- **std::vector**< uint8_t > extradata
  Audio extradata. AAC requires one.

10.6.1 Detailed Description

Audio stream info.

Definition at line 349 of file base_streamer.h.

10.6.2 Field Documentation

10.6.2.1 bitrate

```
int vxg::media::Streamer::StreamInfo::AudioInfo::bitrate
```

Audio bitrate.

Definition at line 357 of file base_streamer.h.

10.6.2.2 channels

```
int vxg::media::Streamer::StreamInfo::AudioInfo::channels
```

Audio channels.

Definition at line 353 of file base_streamer.h.

10.6.2.3 codec

```
AudioCodec vxg::media::Streamer::StreamInfo::AudioInfo::codec
```

Audio codec.

Definition at line 351 of file base_streamer.h.

10.6.2.4 extradata

```
std::vector<uint8_t> vxg::media::Streamer::StreamInfo::AudioInfo::extradata
```

Audio extradata. AAC requires one.

Definition at line 361 of file base_streamer.h.

10.6.2.5 samplerate

```
int vxg::media::Streamer::StreamInfo::AudioInfo::samplerate
```

Audio samplerate.

Definition at line 355 of file base_streamer.h.

10.6.2.6 timebase

```
std::pair<int, int> vxg::media::Streamer::StreamInfo::AudioInfo::timebase
```

Audio timestamps timescale.

Definition at line 359 of file base_streamer.h.

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

- base_streamer.h

10.7 vxg::cloud::agent::callback Class Reference

VXG Cloud manager common callbacks class.

```
#include <agent/callback.h>
```

Public Types

- typedef **std::unique_ptr**< callback > ptr
  std::unique_ptr *to callback*

Public Member Functions

- virtual void on_bye (proto::command::bye_reason reason)=0
  VXG Cloud Bye command callback.
- virtual void on_registered (const **std::string** &sid)
  Registration on the Cloud has passed callback.
- virtual bool on_raw_msg (**std::string** client_id, **std::string** &data)
  raw message callback
- virtual bool on_get_log (**std::string** &log_data)
  Get logging data callback.
- virtual bool on_start_backward_audio (**std::string** url)
  Start backward audio stream.
- virtual bool on_stop_backward_audio (**std::string** url)
  Stop backward audio.
- virtual bool on_get_cam_video_config (proto::video_config &config)
  Get video image config.
- virtual bool on_set_cam_video_config (const proto::video_config &config)
  Set video input config.
- virtual bool on_get_cam_audio_config (proto::audio_config &config)
  Get audio input configuration.
- virtual bool on_set_cam_audio_config (const proto::audio_config &config)
  Set audio input/output config.
- virtual bool on_get_ptz_config (proto::ptz_config &config)
  Get PTZ config.
- virtual bool on_cam_ptz (proto::ptz_command &command)

PTZ command.

- virtual bool on_cam_ptz_preset (proto::ptz_preset &preset_op)

  PTZ preset command.
- virtual bool on_get_osd_config (proto::osd_config &config)

  Get OSD config.
- virtual bool on_set_osd_config (const proto::osd_config &config)

  Set OSD config.
- virtual bool on_get_wifi_config (proto::wifi_config &config)

  Get WiFi config.
- virtual bool on_set_wifi_config (const proto::wifi_network &config)

  Set WiFi config.
- virtual bool on_get_motion_detection_config (proto::motion_detection_config &config)

  Get motion detection configuration.
- virtual bool on_set_motion_detection_config (const proto::motion_detection_config &config)

  Set motion detection config.
- virtual bool on_get_cam_events_config (proto::events_config &config)

  Get events configuration.
- virtual bool on_set_cam_events_config (const proto::events_config &config)

  Set motion detection config.
- virtual bool on_get_timezone (**std::string** &timezone)

  Get device timezone in IANA format.
- virtual bool on_set_timezone (**std::string** timezone)

  Set device timezone in IANA format.
- virtual bool on_get_memorycard_info (proto::event_object::memorycard_info_object &info)

  Get memory card information, If this callback returned false or if `info` status not equal to proto::MCS_NORMAL, the recording will not be started, i.e.
- virtual bool on_cam_upgrade_firmware (const **std::string** &firmware)

  Firmware upgrade.
- virtual bool on_audio_file_play (const **std::string** audio_file_data, const **std::string** filename)

  Audio file play.
- virtual bool on_trigger_event (proto::event_object &event)

10.7.1 Detailed Description

VXG Cloud manager common callbacks class.

Definition at line 17 of file callback.h.

10.7.2 Member Typedef Documentation

10.7.2.1 ptr

```
typedef std::unique_ptr<callback> vxg::cloud::agent::callback::ptr
```

std::unique_ptr to callback

Definition at line 20 of file callback.h.

10.7.3 Member Function Documentation

10.7.3.1 on_audio_file_play()

```
virtual bool vxg::cloud::agent::callback::on_audio_file_play (
            const  std::string audio_file_data,
            const  std::string filename )  [inline], [virtual]
```

Audio file play.

Parameters

in	*audio_file*	Audio file binary data.
in	*audio_file_format*	Audio file data format.

Returns

true if firware upgrade was successfull.

false if firware upgrade failed.

Definition at line 332 of file callback.h.

10.7.3.2 on_bye()

```
virtual void vxg::cloud::agent::callback::on_bye (
            proto::command::bye_reason reason )  [pure virtual]
```

VXG Cloud Bye command callback.

Parameters

reason	bye reason

10.7.3.3 on_cam_ptz()

```
virtual bool vxg::cloud::agent::callback::on_cam_ptz (
            proto::ptz_command & command )  [inline], [virtual]
```

PTZ command.

Parameters

in	*command*	ptz command

Returns

true success

false PTZ command failure

Definition at line 162 of file callback.h.

10.7.3.4 on_cam_ptz_preset()

```
virtual bool vxg::cloud::agent::callback::on_cam_ptz_preset (
            proto::ptz_preset & preset_op ) [inline], [virtual]
```

PTZ preset command.

Parameters

in,out	*preset_op*	ptz preset operation, if operation is proto::PA_CREATE the callee should fill the token.

Returns

true PTZ preset operation success

false PTZ preset operation failure

Definition at line 174 of file callback.h.

10.7.3.5 on_cam_upgrade_firmware()

```
virtual bool vxg::cloud::agent::callback::on_cam_upgrade_firmware (
            const std::string & firmware ) [inline], [virtual]
```

Firmware upgrade.

Parameters

in	*firmware*	Firmware binary data.

Returns

true if firware upgrade was successfull.

false if firware upgrade failed.

Definition at line 322 of file callback.h.

10.7.3.6 on_get_cam_audio_config()

```
virtual bool vxg::cloud::agent::callback::on_get_cam_audio_config (
            proto::audio_config & config )  [inline], [virtual]
```

Get audio input configuration.

Parameters

out	*config*	audio input config

Returns

true get audio input configuration success

false get audio input configuration failed

Definition at line 126 of file callback.h.

10.7.3.7 on_get_cam_events_config()

```
virtual bool vxg::cloud::agent::callback::on_get_cam_events_config (
            proto::events_config & config )  [inline], [virtual]
```

Get events configuration.

Parameters

out	*config*	events config

Returns

true if `config` is valid

false if `config` is invalid

Definition at line 261 of file callback.h.

10.7.3.8 on_get_cam_video_config()

```
virtual bool vxg::cloud::agent::callback::on_get_cam_video_config (
            proto::video_config & config )  [inline], [virtual]
```

Get video image config.

Parameters

out	*config*	video image config

Returns

true if get image config success

false get image config failed

Definition at line 102 of file callback.h.

10.7.3.9 on_get_log()

```
virtual bool vxg::cloud::agent::callback::on_get_log (
              std::string & log_data )  [inline], [virtual]
```

Get logging data callback.

Cloud API provides the way to request log data using Cloud API

Parameters

log_data	log data

Returns

true on success

false on failure

Definition at line 64 of file callback.h.

10.7.3.10 on_get_memorycard_info()

```
virtual bool vxg::cloud::agent::callback::on_get_memorycard_info (
             proto::event_object::memorycard_info_object & info )  [inline], [virtual]
```

Get memory card information, If this callback returned false or if `info` status not equal to proto::MCS_NORMAL, the recording will not be started, i.e.

no agent::media::stream::record_start() will be called.

Parameters

out	*info*	memorycard info

Returns

true if `info` is valid

false if `info` is not valid

Definition at line 312 of file callback.h.

10.7.3.11 on_get_motion_detection_config()

```
virtual bool vxg::cloud::agent::callback::on_get_motion_detection_config (
            proto::motion_detection_config & config )  [inline], [virtual]
```

Get motion detection configuration.

Parameters

out	*config*	Motion detection config if return value is true

Returns

true if `config` is valid

false if failed to get motion detection config

Definition at line 235 of file callback.h.

10.7.3.12 on_get_osd_config()

```
virtual bool vxg::cloud::agent::callback::on_get_osd_config (
            proto::osd_config & config )  [inline], [virtual]
```

Get OSD config.

Parameters

out	*config*	OSD config

Returns

true OSD config get success, `config` is valid

false OSD config get failure, `config` should not be used

Definition at line 186 of file callback.h.

10.7.3.13 on_get_ptz_config()

```
virtual bool vxg::cloud::agent::callback::on_get_ptz_config (
            proto::ptz_config & config ) [inline], [virtual]
```

Get PTZ config.

Parameters

out	*config*	ptz config

Returns

true success

false Get PTZ config failed

Definition at line 150 of file callback.h.

10.7.3.14 on_get_timezone()

```
virtual bool vxg::cloud::agent::callback::on_get_timezone (
             std::string & timezone ) [inline], [virtual]
```

Get device timezone in IANA format.

Parameters

out	*timezone*	name in IANA format

Returns

true if `timezone` is valid

false if `timezone` is not valid

Definition at line 285 of file callback.h.

10.7.3.15 on_get_wifi_config()

```
virtual bool vxg::cloud::agent::callback::on_get_wifi_config (
            proto::wifi_config & config ) [inline], [virtual]
```

Get WiFi config.

Parameters

out	*config*	WiFi config

Returns

true success

false failed

Definition at line 210 of file callback.h.

10.7.3.16 on_raw_msg()

```
virtual bool vxg::cloud::agent::callback::on_raw_msg (
              std::string client_id,
              std::string & data )  [inline], [virtual]
```

raw message callback

Parameters

in	*client_id*	unique id of the client, every raw messages session uses the same unique client_id
in,out	*data*	raw message payload from client, output value will be sent to the client if return value is true

Returns

true raw message handled and reply in the output `data` argument should be sent to the client as reply

false raw message handling failure, `data` output argument should not be sent to client

Definition at line 52 of file callback.h.

10.7.3.17 on_registered()

```
virtual void vxg::cloud::agent::callback::on_registered (
             const  std::string & sid )  [inline], [virtual]
```

Registration on the Cloud has passed callback.

Parameters

sid	Cloud connection session id. Must be saved and provided via the profile::global::instance().cm_register_sid before the next vxg::cloud::agent::manager::start()

Definition at line 36 of file callback.h.

10.7.3.18 on_set_cam_audio_config()

```
virtual bool vxg::cloud::agent::callback::on_set_cam_audio_config (
            const proto::audio_config & config )  [inline], [virtual]
```

Set audio input/output config.

Parameters

config	audio input/output config

Returns

true applied

false failed to set config

Definition at line 138 of file callback.h.

10.7.3.19 on_set_cam_events_config()

```
virtual bool vxg::cloud::agent::callback::on_set_cam_events_config (
            const proto::events_config & config )  [inline], [virtual]
```

Set motion detection config.

Parameters

in	*config*	Motion detection config

Returns

true if `config` was successfully set

false if failed to set `config`

Definition at line 273 of file callback.h.

10.7.3.20 on_set_cam_video_config()

```
virtual bool vxg::cloud::agent::callback::on_set_cam_video_config (
            const proto::video_config & config )  [inline], [virtual]
```

Set video input config.

Parameters

config	video input config

Returns

true Video image input config was successfully set

false Failed to set video input image config

Definition at line 114 of file callback.h.

10.7.3.21 on_set_motion_detection_config()

```
virtual bool vxg::cloud::agent::callback::on_set_motion_detection_config (
            const proto::motion_detection_config & config )  [inline], [virtual]
```

Set motion detection config.

Parameters

in	*config*	motion detection config

Returns

true if `config` was successfully set

false if failed to set `config`

Definition at line 248 of file callback.h.

10.7.3.22 on_set_osd_config()

```
virtual bool vxg::cloud::agent::callback::on_set_osd_config (
            const proto::osd_config & config )  [inline], [virtual]
```

Set OSD config.

Parameters

in	*config*	OSD config

Returns

true OSD config was successfully set

false failed to set OSD config

Definition at line 198 of file callback.h.

10.7.3.23 on_set_timezone()

```
virtual bool vxg::cloud::agent::callback::on_set_timezone (
              std::string timezone )  [inline], [virtual]
```

Set device timezone in IANA format.

Parameters

in	*timezone*	timezone in IANA format

Returns

true if timezone was successfully set

false if timezone was not set

Definition at line 297 of file callback.h.

10.7.3.24 on_set_wifi_config()

```
virtual bool vxg::cloud::agent::callback::on_set_wifi_config (
             const proto::wifi_network & config )  [inline], [virtual]
```

Set WiFi config.

Parameters

in	*config*	WiFi configuration

Returns

true if `config` is valid

false if `config` is invalid

Definition at line 222 of file callback.h.

10.7.3.25 on_start_backward_audio()

```
virtual bool vxg::cloud::agent::callback::on_start_backward_audio (
              std::string url )  [inline], [virtual]
```

Start backward audio stream.

Parameters

url	rtmp url for backward channel, device supports backward audio if on_get_cam_audio_config() set proto::audio_config.caps spkr to true

Implementation should start rtmp client by its own, final implementation is also responsible for the demuxing, decoding and rendering of the audio stream.

Returns

true on success

false on failure

Definition at line 80 of file callback.h.

10.7.3.26 on_stop_backward_audio()

```
virtual bool vxg::cloud::agent::callback::on_stop_backward_audio (
              std::string url )  [inline], [virtual]
```

Stop backward audio.

Parameters

url	backward audio url which was used to start the backward channel

Definition at line 91 of file callback.h.

10.7.3.27 on_trigger_event()

```
virtual bool vxg::cloud::agent::callback::on_trigger_event (
             proto::event_object & event )  [inline], [virtual]
```

Definition at line 338 of file callback.h.

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

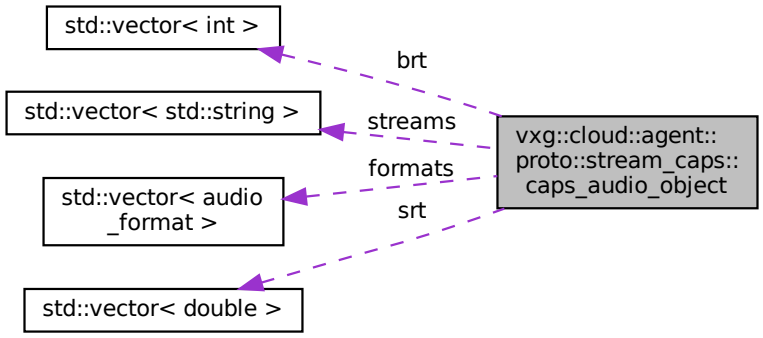
- callback.h

10.8 vxg::cloud::agent::proto::stream_caps::caps_audio_object Struct Reference

Audio streams capabilities.

`#include <agent-proto/objects/caps.h>`

Collaboration diagram for vxg::cloud::agent::proto::stream_caps::caps_audio_object:



Data Fields

- **std::vector**< **std::string** > streams

  Mandatory: list of strings, audio ES that are covered by this capability config.
- **std::vector**< audio_format > formats

  Mandatory: list of string, supported audio formats; currently only “AAC” and "G711U" is supported.
- **std::vector**< int > brt

  Mandatory: [min:int, max:int, step:int], range of bitrates, kbps.
- **std::vector**< double > srt

  Mandatory: list of float, supported samplerates.

10.8.1 Detailed Description

Audio streams capabilities.

Definition at line 247 of file caps.h.

10.8.2 Field Documentation

10.8.2.1 brt

```
std::vector<int> vxg::cloud::agent::proto::stream_caps::caps_audio_object::brt
```

Mandatory: [min:int, max:int, step:int], range of bitrates, kbps.

Definition at line 259 of file caps.h.

10.8.2.2 formats

```
std::vector<audio_format> vxg::cloud::agent::proto::stream_caps::caps_audio_object::formats
```

Mandatory: list of string, supported audio formats; currently only “AAC” and "G711U" is supported.

Definition at line 255 of file caps.h.

10.8.2.3 srt

```
std::vector<double> vxg::cloud::agent::proto::stream_caps::caps_audio_object::srt
```

Mandatory: list of float, supported samplerates.

Definition at line 263 of file caps.h.

10.8.2.4 streams

```
std::vector< std::string> vxg::cloud::agent::proto::stream_caps::caps_audio_object::streams
```

Mandatory: list of strings, audio ES that are covered by this capability config.

Definition at line 250 of file caps.h.

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

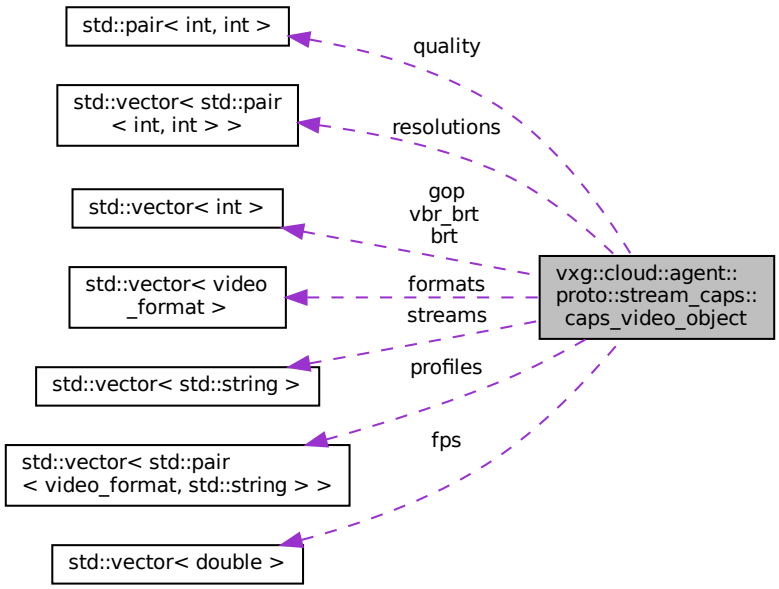
- caps.h

10.9 vxg::cloud::agent::proto::stream_caps::caps_video_object Struct Reference

Video streams capabilities.

`#include <agent-proto/objects/caps.h>`

Collaboration diagram for vxg::cloud::agent::proto::stream_caps::caps_video_object:



Data Fields

- **std::vector**< **std::string** > streams

  Mandatory: list of strings, video ES that are covered by this capability config.
- **std::vector**< video_format > formats

  Mandatory: list of string, supported video formats; currently only "H.264" is supported.
- **std::vector**< **std::pair**< video_format, **std::string** > > profiles

  Optional: list of pairs [string (format), string (profile)], list of profiles for formats (when they have).
- **std::vector**< **std::pair**< int, int > > resolutions

  Mandatory: list of pairs [int (horz), int (vert)], - supported video resolutions.
- **std::vector**< double > fps

  Mandatory: list of float, supported framerates.
- bool vbr

  Mandatory: VBR is supported.
- **std::pair**< int, int > quality

  Optional: [min:int, max:int], range of quality for VBR.
- **std::vector**< int > gop

  Mandatory: gop: [min:int, max:int, step:int], range of gop sizes.
- **std::vector**< int > brt

Mandatory: [min:int, max:int, step:int], range of bitrates, kbps.

- **std::vector**< int > vbr_brt

  Optional: [min:int, max:int, step:int], range of bitrates, kbps.

- bool smoothing

  Optional: True when stream smoothing can be controlled.

10.9.1 Detailed Description

Video streams capabilities.

Definition at line 177 of file caps.h.

10.9.2 Field Documentation

10.9.2.1 brt

std::vector<int> vxg::cloud::agent::proto::stream_caps::caps_video_object::brt

Mandatory: [min:int, max:int, step:int], range of bitrates, kbps.

Definition at line 219 of file caps.h.

10.9.2.2 formats

std::vector<video_format> vxg::cloud::agent::proto::stream_caps::caps_video_object::formats

Mandatory: list of string, supported video formats; currently only “H.264” is supported.

Definition at line 185 of file caps.h.

10.9.2.3 fps

std::vector<double> vxg::cloud::agent::proto::stream_caps::caps_video_object::fps

Mandatory: list of float, supported framerates.

Definition at line 203 of file caps.h.

10.9.2.4 gop

```
std::vector<int> vxg::cloud::agent::proto::stream_caps::caps_video_object::gop
```

Mandatory: gop: [min:int, max:int, step:int], range of gop sizes.

Definition at line 215 of file caps.h.

10.9.2.5 profiles

```
std::vector< std::pair<video_format, std::string> > vxg::cloud::agent::proto::stream_caps↩
::caps_video_object::profiles
```

Optional: list of pairs [string (format), string (profile)], list of profiles for formats (when they have).

Empty list means – color selection is not supported. “format” - one of listed in "formats" names. “profile”

- name of profile. Example: [["H.264", "Baseline"], ["H.264", "Main"], ["H.264", "High"]]

Definition at line 194 of file caps.h.

10.9.2.6 quality

```
std::pair<int, int> vxg::cloud::agent::proto::stream_caps::caps_video_object::quality
```

Optional: [min:int, max:int], range of quality for VBR.

Definition at line 211 of file caps.h.

10.9.2.7 resolutions

```
std::vector< std::pair<int, int> > vxg::cloud::agent::proto::stream_caps::caps_video_↩
object::resolutions
```

Mandatory: list of pairs [int (horz), int (vert)], - supported video resolutions.

Definition at line 199 of file caps.h.

10.9.2.8 smoothing

```
bool vxg::cloud::agent::proto::stream_caps::caps_video_object::smoothing
```

Optional: True when stream smoothing can be controlled.

Definition at line 227 of file caps.h.

10.9.2.9 streams

```
std::vector< std::string> vxg::cloud::agent::proto::stream_caps::caps_video_object::streams
```

Mandatory: list of strings, video ES that are covered by this capability config.

Definition at line 180 of file caps.h.

10.9.2.10 vbr

```
bool vxg::cloud::agent::proto::stream_caps::caps_video_object::vbr
```

Mandatory: VBR is supported.

Definition at line 207 of file caps.h.

10.9.2.11 vbr_brt

```
std::vector<int> vxg::cloud::agent::proto::stream_caps::caps_video_object::vbr_brt
```

Optional: [min:int, max:int, step:int], range of bitrates, kbps.

Definition at line 223 of file caps.h.

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

- caps.h

10.10 vxg::cloud::agent::proto::event_caps Struct Reference

Events capabilies.

```
#include <agent-proto/objects/caps.h>
```

Data Fields

- bool stream

  stream: bool, event can generate stream start
- bool snapshot

  snapshot: bool, event is sent with snapshot
- bool periodic

  periodic: optional bool, the event is a periodic event (camera generates and processes it using specified time interval)
- bool trigger

  trigger: optional bool, the event can be triggered externally, using 6.7
- bool statefull

10.10.1 Detailed Description

Events capabilies.

Definition at line 438 of file caps.h.

10.10.2 Field Documentation

10.10.2.1 periodic

```
bool vxg::cloud::agent::proto::event_caps::periodic
```

periodic: optional bool, the event is a periodic event (camera generates and processes it using specified time interval)

Definition at line 447 of file caps.h.

10.10.2.2 snapshot

```
bool vxg::cloud::agent::proto::event_caps::snapshot
```

snapshot: bool, event is sent with snapshot

Definition at line 443 of file caps.h.

10.10.2.3 statefull

```
bool vxg::cloud::agent::proto::event_caps::statefull
```

Definition at line 469 of file caps.h.

10.10.2.4 stream

```
bool vxg::cloud::agent::proto::event_caps::stream
```

stream: bool, event can generate stream start

Definition at line 440 of file caps.h.

10.10.2.5 trigger

```
bool vxg::cloud::agent::proto::event_caps::trigger
```

trigger: optional bool, the event can be triggered externally, using 6.7

Definition at line 450 of file caps.h.

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

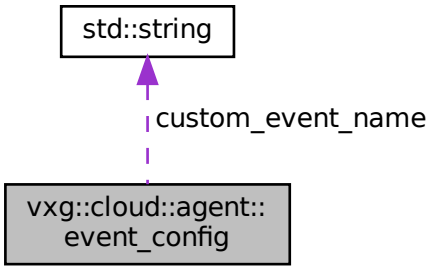
- caps.h

10.11 vxg::cloud::agent::event_config Struct Reference

Event config.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::event_config:



Public Member Functions

- bool name_eq (const event_config &r) const

  Is-equal predicate based on event's name only.
- bool caps_eq (const event_config &r) const

  Is-equal predicate based on event's caps.
- **std::string** name () const

Data Fields

- event_type event

  event: string, event name, see 6.1 Events naming for details
- **std::string** custom_event_name

  Custom event name, used if event set to event_type::ET_CUSTOM.
- bool active

  active: bool, event is active; if not set, corresponding events will not be sent
- bool stream

  stream: bool, start stream when event happens
- bool snapshot

  snapshot: bool, generate snapshot when event happens
- int period

  period: optional int, an interval between periodic events, seconds
- event_caps caps

  Event capabilities.

10.11.1 Detailed Description

Event config.

Definition at line 897 of file config.h.

10.11.2 Member Function Documentation

10.11.2.1 caps_eq()

```
bool vxg::cloud::agent::event_config::caps_eq (
            const event_config & r ) const  [inline]
```

Is-equal predicate based on event's caps.

Parameters

r	

Returns

true Compared configs have equal caps.

false Compared configs have non-equal caps.

Definition at line 937 of file config.h.

10.11.2.2 name()

```
std::string vxg::cloud::agent::event_config::name ( ) const  [inline]
```

Definition at line 941 of file config.h.

10.11.2.3 name_eq()

```
bool vxg::cloud::agent::event_config::name_eq (
            const event_config & r ) const  [inline]
```

Is-equal predicate based on event's name only.

Parameters

r	

Returns

true Compared configs are for the event with equal names.

false Compared configs are for events with non-equal names.

Definition at line 928 of file config.h.

10.11.3 Field Documentation

10.11.3.1 active

```
bool vxg::cloud::agent::event_config::active
```

active: bool, event is active; if not set, corresponding events will not be sent

Definition at line 906 of file config.h.

10.11.3.2 caps

```
event_caps vxg::cloud::agent::event_config::caps
```

Event capabilities.

Definition at line 921 of file config.h.

10.11.3.3 custom_event_name

```
std::string vxg::cloud::agent::event_config::custom_event_name
```

Custom event name, used if event set to event_type::ET_CUSTOM.

Definition at line 902 of file config.h.

10.11.3.4 event

```
event_type vxg::cloud::agent::event_config::event
```

event: string, event name, see 6.1 Events naming for details

Definition at line 899 of file config.h.

10.11.3.5 period

```
int vxg::cloud::agent::event_config::period
```

period: optional int, an interval between periodic events, seconds

Definition at line 915 of file config.h.

10.11.3.6 snapshot

```
bool vxg::cloud::agent::event_config::snapshot
```

snapshot: bool, generate snapshot when event happens

Definition at line 912 of file config.h.

10.11.3.7 stream

```
bool vxg::cloud::agent::event_config::stream
```

stream: bool, start stream when event happens

Definition at line 909 of file config.h.

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

- config.h

10.12 vxg::cloud::agent::manager::event_state::event_state_caps Struct Reference

`#include <agent/manager.h>`

Data Fields

- bool stateful
- bool need_clip
- bool need_snapshot

10.12.1 Detailed Description

Definition at line 71 of file manager.h.

10.12.2 Field Documentation

10.12.2.1 need_clip

`bool vxg::cloud::agent::manager::event_state::event_state_caps::need_clip`

Definition at line 73 of file manager.h.

10.12.2.2 need_snapshot

`bool vxg::cloud::agent::manager::event_state::event_state_caps::need_snapshot`

Definition at line 74 of file manager.h.

10.12.2.3 stateful

`bool vxg::cloud::agent::manager::event_state::event_state_caps::stateful`

Definition at line 72 of file manager.h.

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

- manager.h

10.13 vxg::cloud::agent::event_stream Class Reference

Event stream, abstract class for event generation.

`#include <agent/event-stream.h>`

Public Types

- typedef **std::shared_ptr**< event_stream > ptr

  std::shared_ptr *to event_stream*

Public Member Functions

- event_stream (**std::string** name)

  Construct a new event stream object.
- virtual ~event_stream ()
- bool notify (proto::event_object event)

  Callback should be called to notify event.
- virtual bool start ()=0

  Start events generation, called by internal code when the events generation requested by the VXG Cloud.
- virtual void stop ()=0

  Stop events generation.
- virtual bool get_events (**std::vector**< proto::event_config > &configs)=0

  Get the events configs list This method should update `config` object and add all configurations for the events provided by this event stream.
- virtual bool set_events (const **std::vector**< proto::event_config > &config)=0

  Set the events configuration.
- virtual bool trigger_event (proto::event_object &event)

  Trigger event provided by event_stream If get_events() returned event config with proto::event_config.caps.trigger == true and this event was triggered via the Cloud API this method will be called.
- virtual bool set_trigger_recording (bool enabled, int pre, int post)=0

  Turn on/off the event_stream triggered recording and pre/post recording time.
- virtual bool init ()=0
- virtual void finit ()=0

10.13.1 Detailed Description

Event stream, abstract class for event generation.

Definition at line 14 of file event-stream.h.

10.13.2 Member Typedef Documentation

10.13.2.1 ptr

```
typedef  std::shared_ptr<event_stream> vxg::cloud::agent::event_stream::ptr
```

std::shared_ptr to event_stream

Definition at line 25 of file event-stream.h.

10.13.3 Constructor & Destructor Documentation

10.13.3.1 event_stream()

```
vxg::cloud::agent::event_stream::event_stream (
              std::string name )  [inline]
```

Construct a new event stream object.

Parameters

in	*name*	Event stream name, unique name for event stream

Definition at line 31 of file event-stream.h.

10.13.3.2 ~event_stream()

```
virtual vxg::cloud::agent::event_stream::~event_stream ( )  [inline], [virtual]
```

Definition at line 33 of file event-stream.h.

10.13.4 Member Function Documentation

10.13.4.1 finit()

```
virtual void vxg::cloud::agent::event_stream::finit ( )  [pure virtual]
```

10.13.4.2 get_events()

```
virtual bool vxg::cloud::agent::event_stream::get_events (
              std::vector< proto::event_config > & configs )  [pure virtual]
```

Get the events configs list This method should update `config` object and add all configurations for the events provided by this event stream.

`config` may already include event configs reported by this get_event(), hence the implementation should consider this and do not include its event configs more than one time.

Parameters

out	*configs*	Events configurations.

Returns

true `configs` is valid.

false `configs` is invalid, should not be applied.

Note

This method MUST always return the configs with the same caps, otherwise the new config will not be applied by the library.

10.13.4.3 init()

```
virtual bool vxg::cloud::agent::event_stream::init ( )  [pure virtual]
```

10.13.4.4 notify()

```
bool vxg::cloud::agent::event_stream::notify (
            proto::event_object event )  [inline]
```

Callback should be called to notify event.

Parameters

in	*event*	Event object

Returns

true Event successfully notified

false Notification failed

Definition at line 46 of file event-stream.h.

10.13.4.5 set_events()

```
virtual bool vxg::cloud::agent::event_stream::set_events (
            const  std::vector< proto::event_config > & config )  [pure virtual]
```

Set the events configuration.

Parameters

config	Events configurations list which includes all events reported by the system and other event streams, implementation should find own event configurations and apply them.

Returns

true `config` applied.

false `config` not applied.

10.13.4.6 set_trigger_recording()

```
virtual bool vxg::cloud::agent::event_stream::set_trigger_recording (
            bool enabled,
            int pre,
            int post )  [pure virtual]
```

Turn on/off the event_stream triggered recording and pre/post recording time.

Triggered recording means that event generated by this event_stream should start recording. Final recorded file should have duration of pre time + duration of the even + post time.

Note

Trigger driven recording can be used if platform supports such type of recording, implementation of such type of recording should include specific agent::media::stream records exporting mechanism which handles two consecutive events pre/post time intersections.

Parameters

in	*enabled*	true if event stream should trigger the recording. Implementation may ignore this if not trigger driven record method is used.
in	*pre*	Pre recording time in milliseconds.
in	*post*	Post recording time in milliseconds.

Returns

true

false

10.13.4.7 start()

```
virtual bool vxg::cloud::agent::event_stream::start ( )  [pure virtual]
```

Start events generation, called by internal code when the events generation requested by the VXG Cloud.

Event stream MUST immediately notify states of all stateful events after the start() was invoked.

Returns

true Events generation started

false Failed to start events generation

10.13.4.8 stop()

```
virtual void vxg::cloud::agent::event_stream::stop ( )  [pure virtual]
```

Stop events generation.

10.13.4.9 trigger_event()

```
virtual bool vxg::cloud::agent::event_stream::trigger_event (
            proto::event_object & event )  [inline], [virtual]
```

Trigger event provided by event_stream If get_events() returned event config with proto::event_config.caps.trigger == true and this event was triggered via the Cloud API this method will be called.

The logic of this method should be the same as for vxg::cloud::agent::callback::on_trigger_event().

See also

vxg::cloud::agent::callback::on_trigger_event()

Parameters

event	

Returns

true

false

Definition at line 103 of file event-stream.h.

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

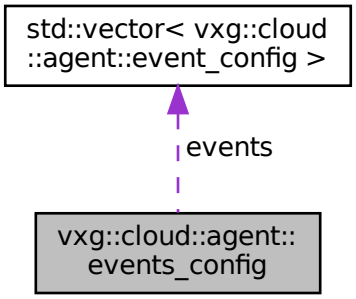
- event-stream.h

10.14 vxg::cloud::agent::events_config Struct Reference

Events config, list of event_config objects.

`#include <agent-proto/objects/config.h>`

Collaboration diagram for vxg::cloud::agent::events_config:



Public Member Functions

- bool get_event_config (const event_object &event, event_config &result)

  Finds event which corresponds to event_config arg in the events_config structure.

Data Fields

- bool enabled

  enabled: bool, indicates global events and event-driven streaming enabling flag
- **std::vector**< event_config > events

  events: list of event_config struct

10.14.1 Detailed Description

Events config, list of event_config objects.

Definition at line 986 of file config.h.

10.14.2 Member Function Documentation

10.14.2.1 get_event_config()

```
bool vxg::cloud::agent::events_config::get_event_config (
            const event_object & event,
            event_config & result )  [inline]
```

Finds event which corresponds to event_config arg in the events_config structure.

Parameters

in	*event*	- event_object, event_object.event used to find the event_config
out	*result*	- if event_config found it will be storred here

Returns

true event found

false event not found

Definition at line 1003 of file config.h.

10.14.3 Field Documentation

10.14.3.1 enabled

```
bool vxg::cloud::agent::events_config::enabled
```

enabled: bool, indicates global events and event-driven streaming enabling flag

Definition at line 989 of file config.h.

10.14.3.2 events

```
std::vector<event_config> vxg::cloud::agent::events_config::events
```

events: list of event_config struct

Definition at line 992 of file config.h.

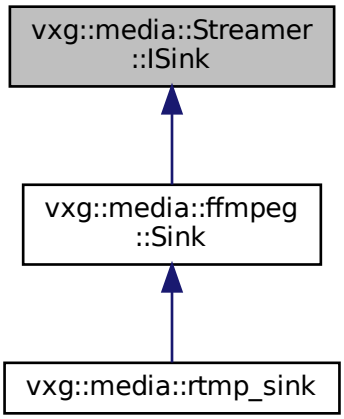
The documentation for this struct was generated from the following file:

- config.h

10.15 vxg::media::Streamer::ISink Class Reference

`#include <streamer/base_streamer.h>`

Inheritance diagram for vxg::media::Streamer::ISink:



Public Types

- typedef **std::shared_ptr**< ISink > ptr

  std::shared_ptr *alias*
- typedef **std::unique_ptr**< ISink > PtrU

  std::unique_ptr *alias*

Public Member Functions

- ISink (uint8_t prio=SINK_THREAD_PRIO)

  Construct a new ISink object.
- virtual ~ISink ()
- virtual bool init (**std::string** url="")=0

  Init sink.
- virtual bool finit ()=0

  Deinit sink.
- virtual bool process (**std::shared_ptr**< MediaFrame > frame)=0

  Process next media frame.
- virtual bool droppable ()=0

  If sink of with dropping its media frames.
- virtual bool negotiate (**std::vector**< Streamer::StreamInfo > info)

  Negotiation callback, this method called with collected from the ISource::negotiate media stream description.
- virtual void error (StreamError error)=0

  Media processing error callback, called when ISink::process returned false or linked source's ISource::pullFrame returned false, or when ISource::error was called.

- virtual **std::string** name ()=0

  Sink name.
- virtual cloud::duration duration ()

  Processed stream duration.
- void set_eos_cb (**std::function**< void(cloud::duration)> eos_cb)
- void set_eos (bool eos)

10.15.1 Detailed Description

Definition at line 492 of file base_streamer.h.

10.15.2 Member Typedef Documentation

10.15.2.1 ptr

```
typedef  std::shared_ptr<ISink> vxg::media::Streamer::ISink::ptr
```

std::shared_ptr alias

Definition at line 497 of file base_streamer.h.

10.15.2.2 PtrU

```
typedef  std::unique_ptr<ISink> vxg::media::Streamer::ISink::PtrU
```

std::unique_ptr alias

Definition at line 499 of file base_streamer.h.

10.15.3 Constructor & Destructor Documentation

10.15.3.1 ISink()

```
vxg::media::Streamer::ISink::ISink (
            uint8_t prio = SINK_THREAD_PRIO )  [inline]
```

Construct a new ISink object.

Parameters

prio	internall thread priority, used on RTOS.

Definition at line 504 of file base_streamer.h.

10.15.3.2 ~ISink()

```
virtual vxg::media::Streamer::ISink::~ISink ( )  [inline], [virtual]
```

Definition at line 510 of file base_streamer.h.

10.15.4 Member Function Documentation

10.15.4.1 droppable()

```
virtual bool vxg::media::Streamer::ISink::droppable ( )  [pure virtual]
```

If sink of with dropping its media frames.

Returns

true Internal media thread allowed to drop frames if internal media queue is full.

false No media frames dropping allowed.

Implemented in vxg::media::rtmp_sink, and vxg::media::ffmpeg::Sink.

10.15.4.2 duration()

```
virtual cloud::duration vxg::media::Streamer::ISink::duration ( )  [inline], [virtual]
```

Processed stream duration.

Returns

duration

Reimplemented in vxg::media::ffmpeg::Sink.

Definition at line 605 of file base_streamer.h.

10.15.4.3 error()

```
virtual void vxg::media::Streamer::ISink::error (
            StreamError error )  [pure virtual]
```

Media processing error callback, called when ISink::process returned false or linked source's ISource::pullFrame returned false, or when ISource::error was called.

Parameters

error	Error type.

Implemented in vxg::media::rtmp_sink, and vxg::media::ffmpeg::Sink.

10.15.4.4 finit()

```
virtual bool vxg::media::Streamer::ISink::finit ( )  [pure virtual]
```

Deinit sink.

Returns

true finit success.

false finit failed.

Implemented in vxg::media::ffmpeg::Sink.

10.15.4.5 init()

```
virtual bool vxg::media::Streamer::ISink::init (
            std::string url = "" )  [pure virtual]
```

Init sink.

Parameters

in	*url*	Url if needed.

Returns

true init success.

false init failed.

Implemented in vxg::media::ffmpeg::Sink, and vxg::media::rtmp_sink.

10.15.4.6 name()

```
virtual  std::string vxg::media::Streamer::ISink::name ( )  [pure virtual]
```

Sink name.

Returns

std::string

Implemented in vxg::media::rtmp_sink, and vxg::media::ffmpeg::Sink.

10.15.4.7 negotiate()

```
virtual bool vxg::media::Streamer::ISink::negotiate (
              std::vector< Streamer::StreamInfo > info )  [inline], [virtual]
```

Negotiation callback, this method called with collected from the ISource::negotiate media stream description.

Parameters

info	List of elementary streams descriptions.

Returns

true If streams descriptions accepted.

false Streams not accepted, will cause media thread stopping.

Reimplemented in vxg::media::ffmpeg::Sink, and vxg::media::rtmp_sink.

Definition at line 557 of file base_streamer.h.

10.15.4.8 process()

```
virtual bool vxg::media::Streamer::ISink::process (
              std::shared_ptr< MediaFrame > frame )  [pure virtual]
```

Process next media frame.

Internal function called by media thread, the last function of media frame travel. Final class process frame in this function: sends to server, writes on disk etc.

Parameters

in	*frame*	Media frame.

Returns

true Media frame successfully processed.

false Media frame processing failed.

10.15.4.9 set_eos()

```
void vxg::media::Streamer::ISink::set_eos (
            bool eos )  [inline]
```

Definition at line 668 of file base_streamer.h.

10.15.4.10 set_eos_cb()

```
void vxg::media::Streamer::ISink::set_eos_cb (
              std::function< void(cloud::duration)> eos_cb )  [inline]
```

Definition at line 664 of file base_streamer.h.

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

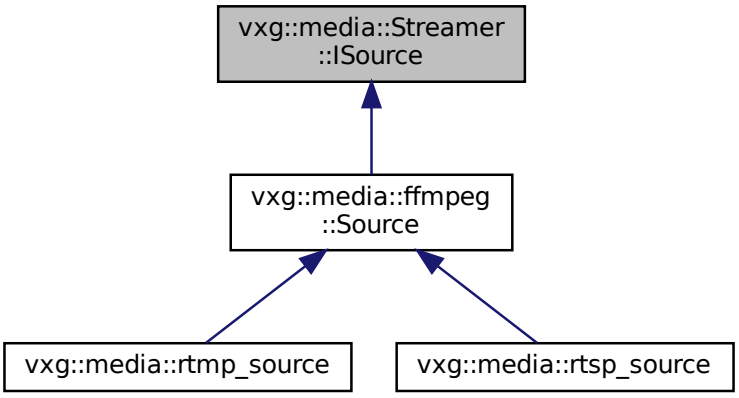
- base_streamer.h

10.16 vxg::media::Streamer::ISource Class Reference

ISource interface class.

`#include <streamer/base_streamer.h>`

Inheritance diagram for vxg::media::Streamer::ISource:



Public Types

- enum Mode { PULL, PUSH }

  Source operation mode.
- typedef **std::shared_ptr**< ISource > ptr

Public Member Functions

- ISource (uint8_t _prio=SRC_THREAD_PRIO, Mode _mode=PULL, bool drop=true)
  Construct a new ISource object.
- virtual bool init (**std::string** url="")=0
  Init source.
- virtual void finit ()=0
  Finit souce.
- virtual void error (StreamError stream_error)
  Error notification.
- virtual **std::vector**< Streamer::StreamInfo > negotiate ()=0
  Negotiation callback.
- virtual **std::shared_ptr**< MediaFrame > pullFrame ()=0
  Main method of the Mode::PULL mode data producing.
- virtual **std::string** name ()=0
  Source class name.
- void pushFrame (**std::shared_ptr**< MediaFrame > frame)
  Implementation should call this method to provide media frames in the Mode::PUSH source operation mode.

Protected Attributes

- Mode mode_

10.16.1 Detailed Description

ISource interface class.

Definition at line 691 of file base_streamer.h.

10.16.2 Member Typedef Documentation

10.16.2.1 ptr

```
typedef  std::shared_ptr<ISource> vxg::media::Streamer::ISource::ptr
```

Definition at line 696 of file base_streamer.h.

10.16.3 Member Enumeration Documentation

10.16.3.1 Mode

```
enum vxg::media::Streamer::ISource::Mode
```

Source operation mode.

Enumerator

PULL	Pull mode. The ISource::pullFrame() will be called from the separate thread. User should implement it and return std::shared_ptr<MediaFrame>.
PUSH	Push mode. Inherited class should feed media data on its own by calling the ISource::pushFrame() method with MediaFrame object passed as argument.

Definition at line 698 of file base_streamer.h.

10.16.4 Constructor & Destructor Documentation

10.16.4.1 ISource()

```
vxg::media::Streamer::ISource::ISource (
            uint8_t _prio = SRC_THREAD_PRIO,
            Mode _mode = PULL,
            bool drop = true )  [inline]
```

Construct a new ISource object.

Parameters

in	*_prio*	Push thread priority. Used if _mode is Mode::PUSH.
in	*_mode*	Source operating mode.
in	*drop*	If true he media frames may be dropped if queue is full.

Definition at line 714 of file base_streamer.h.

10.16.5 Member Function Documentation

10.16.5.1 error()

```
virtual void vxg::media::Streamer::ISource::error (
            StreamError stream_error )  [inline], [virtual]
```

Error notification.

Calling this method will inform media thread and all sinks about error happened in the source.

Parameters

in	*stream_error*	

Definition at line 749 of file base_streamer.h.

10.16.5.2 finit()

```
virtual void vxg::media::Streamer::ISource::finit ( )  [pure virtual]
```

Finit souce.

Implemented in vxg::media::ffmpeg::Source.

10.16.5.3 init()

```
virtual bool vxg::media::Streamer::ISource::init (
              std::string url = "" )  [pure virtual]
```

Init source.

Parameters

url	Url if needed.

Returns

true Init success.

false Init failed.

Implemented in vxg::media::ffmpeg::Source, vxg::media::rtsp_source, and vxg::media::rtmp_source.

10.16.5.4 name()

```
virtual  std::string vxg::media::Streamer::ISource::name ( )  [pure virtual]
```

Source class name.

Returns

std::string

Implemented in vxg::media::rtsp_source, and vxg::media::ffmpeg::Source.

10.16.5.5 negotiate()

```
virtual std::vector<Streamer::StreamInfo> vxg::media::Streamer::ISource::negotiate ( ) [pure virtual]
```

Negotiation callback.

Called by internals. Class implementation should return the list of the streams info source will be producing for the sinks, this list will be then passed to the ISink::negotiate method.

Returns

std::vector<Streamer::StreamInfo>

Implemented in vxg::media::ffmpeg::Source.

10.16.5.6 pullFrame()

```
virtual std::shared_ptr<MediaFrame> vxg::media::Streamer::ISource::pullFrame ( ) [pure virtual]
```

Main method of the Mode::PULL mode data producing.

Called by internals if the source operation mode is Mode::PULL. Implementation should return media frame object with correctly filled fields.

Returns

std::shared_ptr<MediaFrame>

Implemented in vxg::media::ffmpeg::Source.

10.16.5.7 pushFrame()

```
void vxg::media::Streamer::ISource::pushFrame (
            std::shared_ptr< MediaFrame > frame ) [inline]
```

Implementation should call this method to provide media frames in the Mode::PUSH source operation mode.

Parameters

frame	smart pointer to MediaFrame.

Definition at line 849 of file base_streamer.h.

10.16.6 Field Documentation

10.16.6.1 mode_

`Mode vxg::media::Streamer::ISource::mode_ [protected]`

Definition at line 986 of file base_streamer.h.

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

- base_streamer.h

10.17 vxg::logger Class Reference

Logger class, current implementation based on spdlog.

`#include <utils/logging.h>`

Data Structures

- struct options

Public Types

- enum loglevel {
  lvl_crit, lvl_off, lvl_error, lvl_warn,
  lvl_info, lvl_debug, lvl_trace }
- typedef **std::shared_ptr**< spdlog::logger > logger_ptr

Static Public Member Functions

- static **std::shared_ptr**< spdlog::logger > instance (**std::string** name)

  Get pointer to the instance of the named spdlog::logger object.
- static void reset (int argc, char **argv, loglevel l, **std::string** syslog_ident="VXGCloudAgentDefault", std::string crash_logfile_path="", **std::string** logfile_path="", size_t logfile_max_size=(1024 *1024), size_t logfile_max_files=3)

  Reset default logger parameters.
- static void reset (const options &opts)
- static void set_level (logger_ptr log_ptr, loglevel lvl)

  Change the logger object loglevel.
- template<typename FormatString , typename... Args>
  static void info (const FormatString &fmt, const Args &... args)

  Static info log.
- template<typename FormatString , typename... Args>
  static void error (const FormatString &fmt, const Args &... args)

- template<typename FormatString , typename... Args>
  static void warn (const FormatString &fmt, const Args &... args)
- template<typename FormatString , typename... Args>
  static void debug (const FormatString &fmt, const Args &... args)
- template<typename FormatString , typename... Args>
  static void trace (const FormatString &fmt, const Args &... args)
- template<typename T >
  static void trace (const T &msg)
- template<typename T >
  static void debug (const T &msg)
- template<typename T >
  static void info (const T &msg)
- template<typename T >
  static void warn (const T &msg)
- template<typename T >
  static void error (const T &msg)

10.17.1 Detailed Description

Logger class, current implementation based on spdlog.

Definition at line 22 of file logging.h.

10.17.2 Member Typedef Documentation

10.17.2.1 logger_ptr

```
typedef std::shared_ptr<spdlog::logger> vxg::logger::logger_ptr
```

Definition at line 24 of file logging.h.

10.17.3 Member Enumeration Documentation

10.17.3.1 loglevel

```
enum vxg::logger::loglevel
```

Enumerator

lvl_crit	
lvl_off	
lvl_error	
lvl_warn	
lvl_info	
lvl_debug	
lvl_trace	

Definition at line 25 of file logging.h.

10.17.4 Member Function Documentation

10.17.4.1 debug() [1/2]

```
template<typename FormatString , typename...  Args>
static void vxg::logger::debug (
            const FormatString & fmt,
            const Args &...  args )  [inline], [static]
```

Definition at line 282 of file logging.h.

10.17.4.2 debug() [2/2]

```
template<typename T >
static void vxg::logger::debug (
            const T & msg )  [inline], [static]
```

Definition at line 295 of file logging.h.

10.17.4.3 error() [1/2]

```
template<typename FormatString , typename...  Args>
static void vxg::logger::error (
            const FormatString & fmt,
            const Args &...  args )  [inline], [static]
```

Definition at line 274 of file logging.h.

10.17.4.4 error() [2/2]

```
template<typename T >
static void vxg::logger::error (
            const T & msg )  [inline], [static]
```

Definition at line 310 of file logging.h.

10.17.4.5 info() [1/2]

```
template<typename FormatString , typename...  Args>
static void vxg::logger::info (
            const FormatString & fmt,
            const Args &...  args )  [inline], [static]
```

Static info log.

Template Parameters

FormatString	
Args	

Parameters

fmt	
args	

Definition at line 270 of file logging.h.

10.17.4.6 info() [2/2]

```
template<typename T >
static void vxg::logger::info (
             const T & msg )  [inline], [static]
```

Definition at line 300 of file logging.h.

10.17.4.7 instance()

```
static  std::shared_ptr<spdlog::logger> vxg::logger::instance (
              std::string name )  [inline], [static]
```

Get pointer to the instance of the named spdlog::logger object.

On the very first call creates default logger named 'default'. Contructs new logger if logger with such name was never requested

Parameters

in	*name*	Logger name. If logger with such name was already created, then it will be reused, otherwise a new one will be constructed.

Returns

std::shared_ptr<spdlog::logger>

Definition at line 192 of file logging.h.

10.17.4.8 reset() [1/2]

```
static void vxg::logger::reset (
            const options & opts )  [inline], [static]
```

Definition at line 239 of file logging.h.

10.17.4.9 reset() [2/2]

```
static void vxg::logger::reset (
            int argc,
            char ** argv,
            loglevel l,
             std::string syslog_ident = "VXGCloudAgentDefault",
             std::string crash_logfile_path = "",
             std::string logfile_path = "",
            size_t logfile_max_size = (1024 * 1024),
            size_t logfile_max_files = 3 )  [inline], [static]
```

Reset default logger parameters.

Used to change all loggers parameters such as syslog/file sinks usage. Should be called before very first logger::instance() call to take effect. If wasn't called the default console logging sink only will be used for all loggers.

Deprecated Use reset(const options& opts)

Parameters

argc	Process argc
argv	Process argv
l	default loglevel, all loggers will be created with this loglevel, can be overriden with SPDLOG_LEVEL env variable
syslog_ident	Syslog identification string, if empty syslog logging will be disabled.
logfile_path	Rotating plain log file path, if empty no plain log file will be used.
logfile_max_size	Max log file size before invoking logrotate.
logfile_max_files	Max number if rotating logfiles.

Definition at line 220 of file logging.h.

10.17.4.10 set_level()

```
static void vxg::logger::set_level (
            logger_ptr log_ptr,
            loglevel lvl )  [inline], [static]
```

Change the logger object loglevel.

Parameters

log_ptr	Logger object pointer.
lvl	New loglevel.

Definition at line 259 of file logging.h.

10.17.4.11 trace() [1/2]

```
template<typename FormatString , typename...  Args>
static void vxg::logger::trace (
            const FormatString & fmt,
            const Args &...  args )  [inline], [static]
```

Definition at line 286 of file logging.h.

10.17.4.12 trace() [2/2]

```
template<typename T >
static void vxg::logger::trace (
            const T & msg )  [inline], [static]
```

Definition at line 290 of file logging.h.

10.17.4.13 warn() [1/2]

```
template<typename FormatString , typename...  Args>
static void vxg::logger::warn (
            const FormatString & fmt,
            const Args &...  args )  [inline], [static]
```

Definition at line 278 of file logging.h.

10.17.4.14 warn() [2/2]

```
template<typename T >
static void vxg::logger::warn (
            const T & msg )  [inline], [static]
```

Definition at line 305 of file logging.h.

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

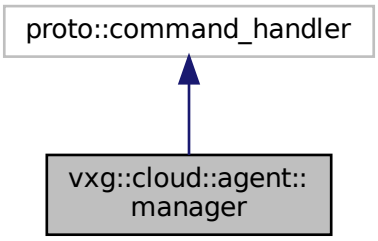
- logging.h

10.18 vxg::cloud::agent::manager Class Reference

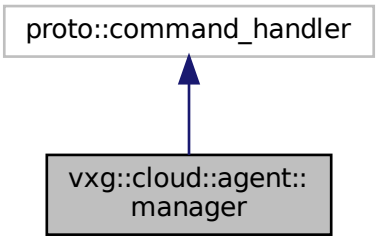
VXG Cloud agent manager class.

`#include <agent/manager.h>`

Inheritance diagram for vxg::cloud::agent::manager:



Collaboration diagram for vxg::cloud::agent::manager:



Public Types

- typedef **std::shared_ptr**< manager > ptr

  shared_ptr to manager object

Public Member Functions

- bool start ()

  Start internal workflow, this is the main function which starts all internal threads and connections.
- void stop ()

  Stop manager, disconnect from the VXG Cloud.

Static Public Member Functions

- static manager::ptr create (callback::ptr callback, proto::access_token::ptr access_token, **std::vector**< agent::media::stream::ptr > media_streams, **std::vector**< event_stream::ptr > event_streams= **std**↩ **::vector**< event_stream::ptr >(0))

  Create manager object.

Protected Member Functions

- bool notify_event (proto::event_object event)
- bool _update_storage_status ()
- void _stop_all_streams ()

  Streams helpers.
- void _stop_stream (agent::media::stream::ptr s)
- void _stop_all_event_streams ()
- void _schedule_periodic_events (proto::events_config &events_conf)
- void _schedule_periodic_event (const proto::event_config &event_conf)
- void _cancel_periodic_event (const proto::event_config &event_conf)
- void _cancel_periodic_events (const proto::events_config &events_conf)
- void _append_internal_custom_events (proto::events_config &config)
- void __trigger_periodic_event (const proto::event_config &event_conf)
- void _init_events_states (const proto::events_config &config)
- void _load_events_configs (proto::events_config &config)
- bool _update_events_configs (const **std::vector**< proto::event_config > &new_configs, **std::vector**< proto::event_config > &dest_configs)
- bool _update_event_stream_configs (const **std::string** &stream_name, const **std::vector**< proto::event↩ _config > &new_configs)
- event_stream::ptr _lookup_event_stream_by_event (const proto::event_object &event)
- event_stream::ptr _lookup_event_stream (const **std::string** &name)
- bool handle_stream_event (proto::event_object &event)
- bool _handle_stream_stateful_event (proto::event_object &event, event_state::stream_delivery_mode delivery_mode)
- bool _handle_stream_stateless_event (proto::event_object &event, event_state::stream_delivery_mode delivery_mode)
- bool handle_event_snapshot (proto::event_object &event)
- bool handle_event_meta_file (proto::event_object &event)
- bool __notify_record_event (**std::string** stream_id, bool on)
- event_state::stream_delivery_mode _current_delivery_mode ()
- bool _schedule_direct_upload (proto::get_direct_upload_url get_upload_url)
- bool _cancel_direct_uploads_by_ticket (**std::string** ticket)
- bool _request_direct_upload_video (proto::get_direct_upload_url direct_upload)
- bool _request_direct_upload_snapshot (proto::get_direct_upload_url direct_upload)
- void _update_direct_upload_queue_latency ()
- bool direct_upload_sync_cb ()
- agent::media::stream::ptr lookup_stream (**std::string** name)
- virtual bool on_get_stream_config (proto::stream_config &config)
- virtual bool on_set_stream_config (const proto::stream_config &config)
- virtual bool on_get_motion_detection_config (proto::motion_detection_config &config)
- virtual bool on_set_motion_detection_config (const proto::motion_detection_config &config)
- virtual bool on_get_cam_video_config (proto::video_config &config)
- virtual bool on_set_cam_video_config (const proto::video_config &config)
- virtual bool on_get_cam_events_config (proto::events_config &config)
- virtual bool on_set_cam_events_config (const proto::events_config &config)
- virtual bool on_get_cam_audio_config (proto::audio_config &config)

- virtual bool on_set_cam_audio_config (const proto::audio_config &config)
- virtual bool on_get_ptz_config (proto::ptz_config &config)
- virtual bool on_cam_ptz (proto::ptz_command command)
- virtual bool on_cam_ptz_preset (proto::ptz_preset &preset_op)
- virtual bool on_get_osd_config (proto::osd_config &config)
- virtual bool on_set_osd_config (const proto::osd_config &config)
- virtual bool on_get_wifi_config (proto::wifi_config &config)
- virtual bool on_set_wifi_config (const proto::wifi_network &config)
- virtual bool on_stream_start (const **std::string** &streamId, int publishSessionID, proto::stream_reason reason)
- virtual bool on_stream_stop (const **std::string** &streamId, proto::stream_reason reason)
- virtual bool on_get_stream_caps (proto::stream_caps &caps)
- virtual bool on_get_supported_streams (proto::supported_streams_config &supportedStreamsConfig)
- virtual bool on_cam_upgrade_firmware (**std::string** url)
- virtual bool on_raw_message (**std::string** client_id, **std::string** &data)
- virtual bool on_set_stream_by_event (proto::stream_by_event_config conf)
- virtual bool on_get_stream_by_event (proto::stream_by_event_config &conf)
- virtual bool on_update_preview (**std::string** url)
- virtual bool on_direct_upload_url (const proto::command::direct_upload_url_base &direct_upload, int event_id, int ref_id)
- virtual bool on_get_log ()
- virtual void on_prepared ()
- virtual void on_closed (int error, proto::command::bye_reason reason)
- virtual bool on_get_timezone (**std::string** &timezone)
- virtual bool on_set_timezone (**std::string** timezone)
- void on_set_periodic_events (const char *name, int period, bool active)
- virtual bool on_audio_file_play (**std::string** url)
- virtual bool on_start_backward (**std::string** &url)
- virtual bool on_stop_backward (**std::string** &url)
- virtual bool on_get_cam_memorycard_timeline (proto::command::cam_memorycard_timeline &timeline)
- virtual bool on_cam_memorycard_synchronize (proto::command::cam_memorycard_synchronize_status &synchronize_status, vxg::cloud::time start, vxg::cloud::time end)
- virtual bool on_cam_memorycard_synchronize_cancel (const **std::string** &request_id)
- virtual bool on_cam_memorycard_recording (const **std::string** &stream_id, bool enabled)
- virtual bool on_trigger_event (**std::string** event, json meta, cloud::time time)
- virtual bool on_set_log_enable (bool bEnable)
- virtual bool on_set_activity (bool bEnable)
- virtual void on_registered (const **std::string** &sid)

10.18.1 Detailed Description

VXG Cloud agent manager class.

Definition at line 28 of file manager.h.

10.18.2 Member Typedef Documentation

10.18.2.1 ptr

```
typedef  std::shared_ptr<manager> vxg::cloud::agent::manager::ptr
```

shared_ptr to manager object

Definition at line 478 of file manager.h.

10.18.3 Member Function Documentation

10.18.3.1 __notify_record_event()

```
bool vxg::cloud::agent::manager::__notify_record_event (
              std::string stream_id,
             bool on )  [protected]
```

10.18.3.2 __trigger_periodic_event()

```
void vxg::cloud::agent::manager::__trigger_periodic_event (
             const proto::event_config & event_conf )  [protected]
```

10.18.3.3 _append_internal_custom_events()

```
void vxg::cloud::agent::manager::_append_internal_custom_events (
             proto::events_config & config )  [protected]
```

10.18.3.4 _cancel_direct_uploads_by_ticket()

```
bool vxg::cloud::agent::manager::_cancel_direct_uploads_by_ticket (
              std::string ticket )  [protected]
```

10.18.3.5 _cancel_periodic_event()

```
void vxg::cloud::agent::manager::_cancel_periodic_event (
             const proto::event_config & event_conf )  [protected]
```

10.18.3.6 _cancel_periodic_events()

```
void vxg::cloud::agent::manager::_cancel_periodic_events (
            const proto::events_config & events_conf )  [protected]
```

10.18.3.7 _current_delivery_mode()

```
event_state::stream_delivery_mode vxg::cloud::agent::manager::_current_delivery_mode ( )  [protected]
```

10.18.3.8 _handle_stream_stateful_event()

```
bool vxg::cloud::agent::manager::_handle_stream_stateful_event (
            proto::event_object & event,
            event_state::stream_delivery_mode delivery_mode )  [protected]
```

10.18.3.9 _handle_stream_stateless_event()

```
bool vxg::cloud::agent::manager::_handle_stream_stateless_event (
            proto::event_object & event,
            event_state::stream_delivery_mode delivery_mode )  [protected]
```

10.18.3.10 _init_events_states()

```
void vxg::cloud::agent::manager::_init_events_states (
            const proto::events_config & config )  [protected]
```

10.18.3.11 _load_events_configs()

```
void vxg::cloud::agent::manager::_load_events_configs (
            proto::events_config & config )  [protected]
```

10.18.3.12 _lookup_event_stream()

```
event_stream::ptr vxg::cloud::agent::manager::_lookup_event_stream (
            const  std::string & name )  [protected]
```

10.18.3.13 _lookup_event_stream_by_event()

```
event_stream::ptr vxg::cloud::agent::manager::_lookup_event_stream_by_event (
            const proto::event_object & event ) [protected]
```

10.18.3.14 _request_direct_upload_snapshot()

```
bool vxg::cloud::agent::manager::_request_direct_upload_snapshot (
            proto::get_direct_upload_url direct_upload ) [protected]
```

10.18.3.15 _request_direct_upload_video()

```
bool vxg::cloud::agent::manager::_request_direct_upload_video (
            proto::get_direct_upload_url direct_upload ) [protected]
```

10.18.3.16 _schedule_direct_upload()

```
bool vxg::cloud::agent::manager::_schedule_direct_upload (
            proto::get_direct_upload_url get_upload_url ) [protected]
```

10.18.3.17 _schedule_periodic_event()

```
void vxg::cloud::agent::manager::_schedule_periodic_event (
            const proto::event_config & event_conf ) [protected]
```

10.18.3.18 _schedule_periodic_events()

```
void vxg::cloud::agent::manager::_schedule_periodic_events (
            proto::events_config & events_conf ) [protected]
```

10.18.3.19 _stop_all_event_streams()

```
void vxg::cloud::agent::manager::_stop_all_event_streams ( ) [protected]
```

10.18.3.20 _stop_all_streams()

```
void vxg::cloud::agent::manager::_stop_all_streams ( )  [protected]
```

Streams helpers.

10.18.3.21 _stop_stream()

```
void vxg::cloud::agent::manager::_stop_stream (
            agent::media::stream::ptr s )  [protected]
```

10.18.3.22 _update_direct_upload_queue_latency()

```
void vxg::cloud::agent::manager::_update_direct_upload_queue_latency ( )  [protected]
```

10.18.3.23 _update_event_stream_configs()

```
bool vxg::cloud::agent::manager::_update_event_stream_configs (
            const  std::string & stream_name,
            const  std::vector< proto::event_config > & new_configs )  [protected]
```

10.18.3.24 _update_events_configs()

```
bool vxg::cloud::agent::manager::_update_events_configs (
            const  std::vector< proto::event_config > & new_configs,
             std::vector< proto::event_config > & dest_configs )  [protected]
```

10.18.3.25 _update_storage_status()

```
bool vxg::cloud::agent::manager::_update_storage_status ( )  [protected]
```

10.18.3.26 create()

```
static manager::ptr vxg::cloud::agent::manager::create (
            callback::ptr callback,
            proto::access_token::ptr access_token,
             std::vector< agent::media::stream::ptr > media_streams,
             std::vector< event_stream::ptr > event_streams =  std::vector< event_stream::ptr >(0)
)  [static]
```

Create manager object.

Parameters

in	*callback*	cm::callback object, should not be null
in	*access_token*	VXG Cloud access token
in	*media_streams*	List of **std::shared_ptr** to base_stream derived objects. Should have at least one element. base_stream is abstract class so you need to declare you own class derived from the base_stream or use one of the provided classes (rtsp_stream,...), basically each stream is for example one rtsp stream provided by the device. Each media stream device has should be represented as a separate base_stream derived object, currently only two streams per device are supported by the VXG Cloud.
in	*event_streams*	List of event_stream::ptr, can be empty. event_stream is abstract class so final implementation should use own class derived from the event_stream.

Returns

manager::ptr

10.18.3.27 direct_upload_sync_cb()

```
bool vxg::cloud::agent::manager::direct_upload_sync_cb ( )  [protected]
```

10.18.3.28 handle_event_meta_file()

```
bool vxg::cloud::agent::manager::handle_event_meta_file (
            proto::event_object & event )  [protected]
```

10.18.3.29 handle_event_snapshot()

```
bool vxg::cloud::agent::manager::handle_event_snapshot (
            proto::event_object & event )  [protected]
```

10.18.3.30 handle_stream_event()

```
bool vxg::cloud::agent::manager::handle_stream_event (
            proto::event_object & event )  [protected]
```

10.18.3.31 lookup_stream()

```
agent::media::stream::ptr vxg::cloud::agent::manager::lookup_stream (
             std::string name )  [protected]
```

10.18.3.32 notify_event()

```
bool vxg::cloud::agent::manager::notify_event (
            proto::event_object event )  [protected]
```

10.18.3.33 on_audio_file_play()

```
virtual bool vxg::cloud::agent::manager::on_audio_file_play (
             std::string url )  [protected], [virtual]
```

10.18.3.34 on_cam_memorycard_recording()

```
virtual bool vxg::cloud::agent::manager::on_cam_memorycard_recording (
            const  std::string & stream_id,
            bool enabled )  [protected], [virtual]
```

10.18.3.35 on_cam_memorycard_synchronize()

```
virtual bool vxg::cloud::agent::manager::on_cam_memorycard_synchronize (
            proto::command::cam_memorycard_synchronize_status & synchronize_status,
            vxg::cloud::time start,
            vxg::cloud::time end )  [protected], [virtual]
```

10.18.3.36 on_cam_memorycard_synchronize_cancel()

```
virtual bool vxg::cloud::agent::manager::on_cam_memorycard_synchronize_cancel (
            const  std::string & request_id )  [protected], [virtual]
```

10.18.3.37 on_cam_ptz()

```
virtual bool vxg::cloud::agent::manager::on_cam_ptz (
            proto::ptz_command command )  [protected], [virtual]
```

10.18.3.38 on_cam_ptz_preset()

```
virtual bool vxg::cloud::agent::manager::on_cam_ptz_preset (
            proto::ptz_preset & preset_op )  [protected], [virtual]
```

10.18.3.39 on_cam_upgrade_firmware()

```
virtual bool vxg::cloud::agent::manager::on_cam_upgrade_firmware (
             std::string url )  [protected], [virtual]
```

10.18.3.40 on_closed()

```
virtual void vxg::cloud::agent::manager::on_closed (
            int error,
            proto::command::bye_reason reason )  [protected], [virtual]
```

10.18.3.41 on_direct_upload_url()

```
virtual bool vxg::cloud::agent::manager::on_direct_upload_url (
            const proto::command::direct_upload_url_base & direct_upload,
            int event_id,
            int ref_id )  [protected], [virtual]
```

10.18.3.42 on_get_cam_audio_config()

```
virtual bool vxg::cloud::agent::manager::on_get_cam_audio_config (
            proto::audio_config & config )  [protected], [virtual]
```

10.18.3.43 on_get_cam_events_config()

```
virtual bool vxg::cloud::agent::manager::on_get_cam_events_config (
            proto::events_config & config )  [protected], [virtual]
```

10.18.3.44 on_get_cam_memorycard_timeline()

```
virtual bool vxg::cloud::agent::manager::on_get_cam_memorycard_timeline (
            proto::command::cam_memorycard_timeline & timeline )  [protected], [virtual]
```

10.18.3.45 on_get_cam_video_config()

```
virtual bool vxg::cloud::agent::manager::on_get_cam_video_config (
            proto::video_config & config )  [protected], [virtual]
```

10.18.3.46 on_get_log()

```
virtual bool vxg::cloud::agent::manager::on_get_log ( )  [protected], [virtual]
```

10.18.3.47 on_get_motion_detection_config()

```
virtual bool vxg::cloud::agent::manager::on_get_motion_detection_config (
            proto::motion_detection_config & config )  [protected], [virtual]
```

10.18.3.48 on_get_osd_config()

```
virtual bool vxg::cloud::agent::manager::on_get_osd_config (
            proto::osd_config & config )  [protected], [virtual]
```

10.18.3.49 on_get_ptz_config()

```
virtual bool vxg::cloud::agent::manager::on_get_ptz_config (
            proto::ptz_config & config )  [protected], [virtual]
```

10.18.3.50 on_get_stream_by_event()

```
virtual bool vxg::cloud::agent::manager::on_get_stream_by_event (
            proto::stream_by_event_config & conf )  [protected], [virtual]
```

10.18.3.51 on_get_stream_caps()

```
virtual bool vxg::cloud::agent::manager::on_get_stream_caps (
            proto::stream_caps & caps )  [protected], [virtual]
```

10.18.3.52 on_get_stream_config()

```
virtual bool vxg::cloud::agent::manager::on_get_stream_config (
            proto::stream_config & config )  [protected], [virtual]
```

10.18.3.53 on_get_supported_streams()

```
virtual bool vxg::cloud::agent::manager::on_get_supported_streams (
            proto::supported_streams_config & supportedStreamsConfig )  [protected], [virtual]
```

10.18.3.54 on_get_timezone()

```
virtual bool vxg::cloud::agent::manager::on_get_timezone (
             std::string & timezone )  [protected], [virtual]
```

10.18.3.55 on_get_wifi_config()

```
virtual bool vxg::cloud::agent::manager::on_get_wifi_config (
            proto::wifi_config & config )  [protected], [virtual]
```

10.18.3.56 on_prepared()

```
virtual void vxg::cloud::agent::manager::on_prepared ( )  [protected], [virtual]
```

10.18.3.57 on_raw_message()

```
virtual bool vxg::cloud::agent::manager::on_raw_message (
             std::string client_id,
             std::string & data )  [protected], [virtual]
```

10.18.3.58 on_registered()

```
virtual void vxg::cloud::agent::manager::on_registered (
            const  std::string & sid )  [protected], [virtual]
```

10.18.3.59 on_set_activity()

```
virtual bool vxg::cloud::agent::manager::on_set_activity (
            bool bEnable )  [protected], [virtual]
```

10.18.3.60 on_set_cam_audio_config()

```
virtual bool vxg::cloud::agent::manager::on_set_cam_audio_config (
            const proto::audio_config & config )  [protected], [virtual]
```

10.18.3.61 on_set_cam_events_config()

```
virtual bool vxg::cloud::agent::manager::on_set_cam_events_config (
            const proto::events_config & config )  [protected], [virtual]
```

10.18.3.62 on_set_cam_video_config()

```
virtual bool vxg::cloud::agent::manager::on_set_cam_video_config (
            const proto::video_config & config )  [protected], [virtual]
```

10.18.3.63 on_set_log_enable()

```
virtual bool vxg::cloud::agent::manager::on_set_log_enable (
            bool bEnable )  [protected], [virtual]
```

10.18.3.64 on_set_motion_detection_config()

```
virtual bool vxg::cloud::agent::manager::on_set_motion_detection_config (
            const proto::motion_detection_config & config )  [protected], [virtual]
```

10.18.3.65 on_set_osd_config()

```
virtual bool vxg::cloud::agent::manager::on_set_osd_config (
            const proto::osd_config & config )  [protected], [virtual]
```

10.18.3.66 on_set_periodic_events()

```
void vxg::cloud::agent::manager::on_set_periodic_events (
            const char * name,
            int period,
            bool active )  [protected]
```

10.18.3.67 on_set_stream_by_event()

```
virtual bool vxg::cloud::agent::manager::on_set_stream_by_event (
            proto::stream_by_event_config conf )  [protected], [virtual]
```

10.18.3.68 on_set_stream_config()

```
virtual bool vxg::cloud::agent::manager::on_set_stream_config (
            const proto::stream_config & config )  [protected], [virtual]
```

10.18.3.69 on_set_timezone()

```
virtual bool vxg::cloud::agent::manager::on_set_timezone (
             std::string timezone )  [protected], [virtual]
```

10.18.3.70 on_set_wifi_config()

```
virtual bool vxg::cloud::agent::manager::on_set_wifi_config (
            const proto::wifi_network & config )  [protected], [virtual]
```

10.18.3.71 on_start_backward()

```
virtual bool vxg::cloud::agent::manager::on_start_backward (
             std::string & url )  [protected], [virtual]
```

10.18.3.72 on_stop_backward()

```
virtual bool vxg::cloud::agent::manager::on_stop_backward (
             std::string & url )  [protected], [virtual]
```

10.18.3.73 on_stream_start()

```
virtual bool vxg::cloud::agent::manager::on_stream_start (
            const  std::string & streamId,
            int publishSessionID,
            proto::stream_reason reason )  [protected], [virtual]
```

10.18.3.74 on_stream_stop()

```
virtual bool vxg::cloud::agent::manager::on_stream_stop (
            const  std::string & streamId,
            proto::stream_reason reason )  [protected], [virtual]
```

10.18.3.75 on_trigger_event()

```
virtual bool vxg::cloud::agent::manager::on_trigger_event (
             std::string event,
            json meta,
            cloud::time time )  [protected], [virtual]
```

10.18.3.76 on_update_preview()

```
virtual bool vxg::cloud::agent::manager::on_update_preview (
             std::string url )  [protected], [virtual]
```

10.18.3.77 start()

```
bool vxg::cloud::agent::manager::start ( )
```

Start internal workflow, this is the main function which starts all internal threads and connections.

Returns

true started

false start failed

10.18.3.78 stop()

```
void vxg::cloud::agent::manager::stop ( )
```

Stop manager, disconnect from the VXG Cloud.

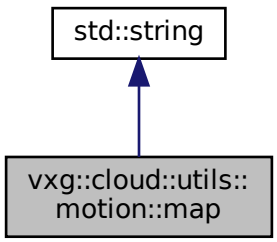
The documentation for this class was generated from the following file:

- manager.h

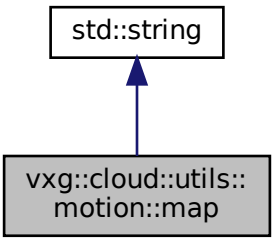
10.19 vxg::cloud::utils::motion::map Struct Reference

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

Inheritance diagram for vxg::cloud::utils::motion::map:



Collaboration diagram for vxg::cloud::utils::motion::map:



Public Member Functions

- map ()
- map (const map &motionMap)
- map & operator= (const **std::string** &motionMap)

Static Public Member Functions

- static **std::string** pack (const **std::string** &unpackedGrid)
- static **std::string** unpack (const **std::string** &packedMap, size_t outputLen)

10.19.1 Detailed Description

Definition at line 126 of file utils.h.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 map() [1/2]

```
vxg::cloud::utils::motion::map::map ( )  [inline], [explicit]
```

Definition at line 127 of file utils.h.

10.19.2.2 map() [2/2]

```
vxg::cloud::utils::motion::map::map (
            const map & motionMap )  [inline]
```

Definition at line 129 of file utils.h.

10.19.3 Member Function Documentation

10.19.3.1 operator=()

```
map& vxg::cloud::utils::motion::map::operator= (
            const  std::string & motionMap )  [inline]
```

Definition at line 131 of file utils.h.

10.19.3.2 pack()

```
static  std::string vxg::cloud::utils::motion::map::pack (
            const  std::string & unpackedGrid )  [static]
```

10.19.3.3 unpack()

```
static  std::string vxg::cloud::utils::motion::map::unpack (
            const  std::string & packedMap,
            size_t outputLen )  [static]
```

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

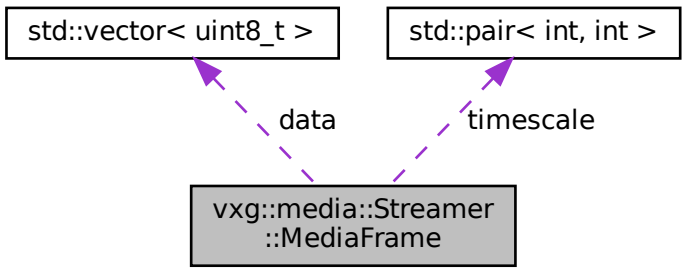
- utils.h

10.20 vxg::media::Streamer::MediaFrame Struct Reference

Media frame container.

`#include <streamer/base_streamer.h>`

Collaboration diagram for vxg::media::Streamer::MediaFrame:



Public Member Functions

- bool operator< (const MediaFrame &rv)

  Two frames comparation using timestamps.

Data Fields

- **std::vector**< uint8_t > data

  Media frame data.
- size_t len

  Media frame data length.
- int64_t pts

  Media frame timestamp in timescale that corresponds to timescale.
- int64_t dts

  Media frame decoding timestamp in timescale that corresponds to timescale.
- int64_t duration

  Media frame duration if needed.
- bool is_key

  Is key frame flag.
- MediaType type

  Media frame type.
- **std::pair**< int, int > timescale

  Timescale of pts and duration. ex. : 1/90000, 1/1000 etc.
- int64_t time_realtime

  Real time if available from source, for ex.

Static Public Attributes

- static constexpr int64_t NO_PTS

10.20.1 Detailed Description

Media frame container.

Definition at line 403 of file base_streamer.h.

10.20.2 Member Function Documentation

10.20.2.1 operator<()

```
bool vxg::media::Streamer::MediaFrame::operator< (
            const MediaFrame & rv )  [inline]
```

Two frames comparation using timestamps.

Parameters

rv	Right value

Returns

true

false

Definition at line 421 of file base_streamer.h.

10.20.3 Field Documentation

10.20.3.1 data

```
std::vector<uint8_t> vxg::media::Streamer::MediaFrame::data
```

Media frame data.

Definition at line 426 of file base_streamer.h.

10.20.3.2 dts

```
int64_t vxg::media::Streamer::MediaFrame::dts
```

Media frame decoding timestamp in timescale that corresponds to timescale.

Definition at line 433 of file base_streamer.h.

10.20.3.3 duration

```
int64_t vxg::media::Streamer::MediaFrame::duration
```

Media frame duration if needed.

Definition at line 435 of file base_streamer.h.

10.20.3.4 is_key

```
bool vxg::media::Streamer::MediaFrame::is_key
```

Is key frame flag.

Definition at line 437 of file base_streamer.h.

10.20.3.5 len

```
size_t vxg::media::Streamer::MediaFrame::len
```

Media frame data length.

Definition at line 428 of file base_streamer.h.

10.20.3.6 NO_PTS

```
constexpr int64_t vxg::media::Streamer::MediaFrame::NO_PTS  [static], [constexpr]
```

Definition at line 423 of file base_streamer.h.

10.20.3.7 pts

```
int64_t vxg::media::Streamer::MediaFrame::pts
```

Media frame timestamp in timescale that corresponds to timescale.

Definition at line 430 of file base_streamer.h.

10.20.3.8 time_realtime

```
int64_t vxg::media::Streamer::MediaFrame::time_realtime
```

Real time if available from source, for ex.

pts based on NTP time from RTCP SR

Definition at line 444 of file base_streamer.h.

10.20.3.9 timescale

```
std::pair<int, int> vxg::media::Streamer::MediaFrame::timescale
```

Timescale of pts and duration. ex. : 1/90000, 1/1000 etc.

Definition at line 441 of file base_streamer.h.

10.20.3.10 type

```
MediaType vxg::media::Streamer::MediaFrame::type
```

Media frame type.

Definition at line 439 of file base_streamer.h.

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

- base_streamer.h

10.21 vxg::cloud::agent::proto::motion_detection_caps Struct Reference

Motion detection capabilities camera capabilities that limit possible motion detection configuration.

```
#include <agent-proto/objects/caps.h>
```

Data Fields

- size_t max_regions
  Mandatory: supported number of motion regions.
- motion_sensitivity sensitivity
  Mandatory: (“region”, “frame”), default “region”; indicates if sensitivity can be set for region or for whole frame only.
- motion_region_shape region_shape
  Mandatory: (“rect”, “any”), default “any”; specifies limitation of region shape.

10.21.1 Detailed Description

Motion detection capabilities camera capabilities that limit possible motion detection configuration.

Definition at line 336 of file caps.h.

10.21.2 Field Documentation

10.21.2.1 max_regions

```
size_t vxg::cloud::agent::proto::motion_detection_caps::max_regions
```

Mandatory: supported number of motion regions.

Definition at line 339 of file caps.h.

10.21.2.2 region_shape

```
motion_region_shape vxg::cloud::agent::proto::motion_detection_caps::region_shape
```

Mandatory: (“rect”, “any”), default “any”; specifies limitation of region shape.

Definition at line 348 of file caps.h.

10.21.2.3 sensitivity

```
motion_sensitivity vxg::cloud::agent::proto::motion_detection_caps::sensitivity
```

Mandatory: (“region”, “frame”), default “region”; indicates if sensitivity can be set for region or for whole frame only.

Definition at line 344 of file caps.h.

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

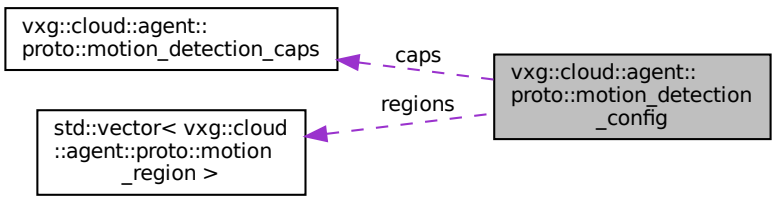
- caps.h

10.22 vxg::cloud::agent::proto::motion_detection_config Struct Reference

Motion detection config.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::proto::motion_detection_config:



Data Fields

- int columns

  Mandatory.
- int rows

  Mandatory.
- motion_detection_caps caps

  Mandatory for CM => SRV (reply to 'get_motion_detection') camera capabilities that limit possible motion detection configuration.
- **std::vector**< motion_region > regions

  Mandatory List of motion regions.

10.22.1 Detailed Description

Motion detection config.

Definition at line 280 of file config.h.

10.22.2 Field Documentation

10.22.2.1 caps

```
motion_detection_caps vxg::cloud::agent::proto::motion_detection_config::caps
```

Mandatory for CM => SRV (reply to 'get_motion_detection') camera capabilities that limit possible motion detection configuration.

Definition at line 289 of file config.h.

10.22.2.2 columns

```
int vxg::cloud::agent::proto::motion_detection_config::columns
```

Mandatory.

Definition at line 283 of file config.h.

10.22.2.3 regions

```
std::vector<motion_region> vxg::cloud::agent::proto::motion_detection_config::regions
```

Mandatory List of motion regions.

Definition at line 292 of file config.h.

10.22.2.4 rows

```
int vxg::cloud::agent::proto::motion_detection_config::rows
```

Mandatory.

Definition at line 286 of file config.h.

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

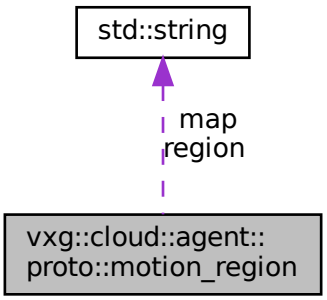
- config.h

10.23 vxg::cloud::agent::proto::motion_region Struct Reference

Motion detection related structs.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::proto::motion_region:



Data Fields

- **std::string** region

  Mandatory: name of region if supported by camera.
- **std::string** map

  Mandatory: String is packed with Apple Packbit algorithm and after that encoded with Base64.
- size_t sensitivity

  Mandatory: range 0-100; 0 - minimal sensitivity.
- bool enabled

  Mandatory: indicates that motion detection is enabled for the region.

10.23.1 Detailed Description

Motion detection related structs.

Motion region

Definition at line 243 of file config.h.

10.23.2 Field Documentation

10.23.2.1 enabled

```
bool vxg::cloud::agent::proto::motion_region::enabled
```

Mandatory: indicates that motion detection is enabled for the region.

Definition at line 265 of file config.h.

10.23.2.2 map

```
std::string vxg::cloud::agent::proto::motion_region::map
```

Mandatory: String is packed with Apple Packbit algorithm and after that encoded with Base64.

Bitstring where “1” denotes an active cell and a “0” an inactive cell. The first cell is in the upper left corner. Then the cell order goes first from left to right and then from up to down. If the number of cells is not a multiple of 8 the last byte is padded with zeros.

Definition at line 255 of file config.h.

10.23.2.3 region

```
std::string vxg::cloud::agent::proto::motion_region::region
```

Mandatory: name of region if supported by camera.

Definition at line 246 of file config.h.

10.23.2.4 sensitivity

```
size_t vxg::cloud::agent::proto::motion_region::sensitivity
```

Mandatory: range 0-100; 0 - minimal sensitivity.

If sensitivity is supported only for whole frame, the same value should be used for all regions.

Definition at line 261 of file config.h.

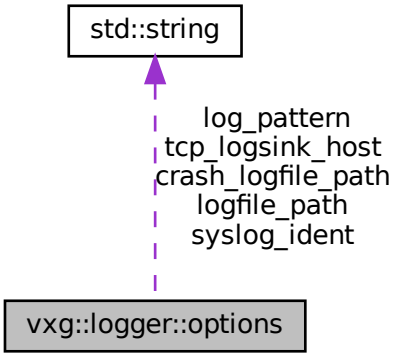
The documentation for this struct was generated from the following file:

- config.h

10.24 vxg::logger::options Struct Reference

```
#include <utils/logging.h>
```

Collaboration diagram for vxg::logger::options:



Data Fields

- **std::string** log_pattern
- **std::string** logfile_path
- size_t logfile_max_size
- size_t logfile_max_files
- **std::string** crash_logfile_path
- **std::string** syslog_ident
- loglevel default_loglevel
- bool tcp_logsink_enabled
- **std::string** tcp_logsink_host
- uint16_t tcp_logsink_port

10.24.1 Detailed Description

Definition at line 35 of file logging.h.

10.24.2 Field Documentation

10.24.2.1 crash_logfile_path

```
std::string vxg::logger::options::crash_logfile_path
```

Definition at line 41 of file logging.h.

10.24.2.2 default_loglevel

```
loglevel vxg::logger::options::default_loglevel
```

Definition at line 43 of file logging.h.

10.24.2.3 log_pattern

```
std::string vxg::logger::options::log_pattern
```

Definition at line 36 of file logging.h.

10.24.2.4 logfile_max_files

```
size_t vxg::logger::options::logfile_max_files
```

Definition at line 40 of file logging.h.

10.24.2.5 logfile_max_size

```
size_t vxg::logger::options::logfile_max_size
```

Definition at line 39 of file logging.h.

10.24.2.6 logfile_path

```
std::string vxg::logger::options::logfile_path
```

Definition at line 38 of file logging.h.

10.24.2.7 syslog_ident

```
std::string vxg::logger::options::syslog_ident
```

Definition at line 42 of file logging.h.

10.24.2.8 tcp_logsink_enabled

```
bool vxg::logger::options::tcp_logsink_enabled
```

Definition at line 44 of file logging.h.

10.24.2.9 tcp_logsink_host

```
std::string vxg::logger::options::tcp_logsink_host
```

Definition at line 45 of file logging.h.

10.24.2.10 tcp_logsink_port

```
uint16_t vxg::logger::options::tcp_logsink_port
```

Definition at line 46 of file logging.h.

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

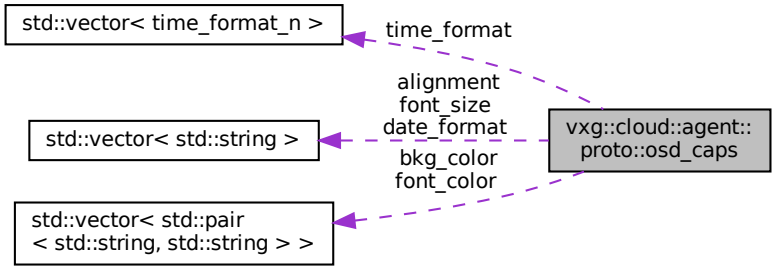
- logging.h

10.25 vxg::cloud::agent::proto::osd_caps Struct Reference

OSD capabilities.

`#include <agent-proto/objects/caps.h>`

Collaboration diagram for vxg::cloud::agent::proto::osd_caps:



Data Fields

- bool system_id

  system_id: bool, True when OSD supports separate system_id enabling/disabling
- bool system_id_text

  system_id_text: bool, True when OSD supports separate system_id customization
- bool time

  time: bool, True when OSD supports separate time enabling/disabling
- **std::vector**< time_format_n > time_format

  time_format: list of string, supported time formats.
- bool date

  date: bool, True when OSD supports separate date enabling/disabling
- **std::vector**< **std::string** > date_format

  date_format: list of string, supported date formats.
- **std::vector**< **std::string** > font_size

  font_size: list of string, describes supported font sizes.
- **std::vector**< **std::pair**< **std::string**, **std::string** > > font_color

  font_color: list of pairs [string (name), optional string (value)], predefined set of possible font colors.
- **std::vector**< **std::pair**< **std::string**, **std::string** > > bkg_color

  bkg_color: list of pairs [string (name), optional string (value)], predefined set of possible background colors.
- bool bkg_transp

  bkg_transp: bool, True when OSD supports background transparency
- **std::vector**< **std::string** > alignment

  alignment: list of strings, supported OSD positions.

10.25.1 Detailed Description

OSD capabilities.

Definition at line 615 of file caps.h.

10.25.2 Field Documentation

10.25.2.1 alignment

```
std::vector< std::string> vxg::cloud::agent::proto::osd_caps::alignment
```

alignment: list of strings, supported OSD positions.

Empty list means – position can't be changed. Example: ["UpperLeft", "UpperRight", "LowerLeft", "LowerRight"]

Definition at line 654 of file caps.h.

10.25.2.2 bkg_color

```
std::vector< std::pair< std::string, std::string> > vxg::cloud::agent::proto::osd_caps↩
::bkg_color
```

bkg_color: list of pairs [string (name), optional string (value)], predefined set of possible background colors.

Empty list means – color selection is not supported. Optioanal value is a RGB color code in HEX. Example: [["↩
Black", "000000"]]

Definition at line 648 of file caps.h.

10.25.2.3 bkg_transp

```
bool vxg::cloud::agent::proto::osd_caps::bkg_transp
```

bkg_transp: bool, True when OSD supports background transparency

Definition at line 650 of file caps.h.

10.25.2.4 date

```
bool vxg::cloud::agent::proto::osd_caps::date
```

date: bool, True when OSD supports separate date enabling/disabling

Definition at line 629 of file caps.h.

10.25.2.5 date_format

```
std::vector< std::string> vxg::cloud::agent::proto::osd_caps::date_format
```

date_format: list of string, supported date formats.

Empty list means – date format selection is not supported. Example: ["YYYY-MM-DD", "MM-DD-YYYY", "DD-MM-YYYY", "YYYY/MM/DD", "MM/DD/YYYY2, "DD/MM/YYYY"]

Definition at line 633 of file caps.h.

10.25.2.6 font_color

```
std::vector< std::pair< std::string, std::string> > vxg::cloud::agent::proto::osd_caps↩
::font_color
```

font_color: list of pairs [string (name), optional string (value)], predefined set of possible font colors.

Empty list means – color selection is not supported. Optioanal value is a RGB color code in HEX. Example: [["↩
Orange", "FF9C00"]]

Definition at line 642 of file caps.h.

10.25.2.7 font_size

```
std::vector< std::string> vxg::cloud::agent::proto::osd_caps::font_size
```

font_size: list of string, describes supported font sizes.

Empty list means – font size format selection is not supported. Examples: ["16", "32", "48", "64", "auto"] or ["Small", "Normal", "Big"]

Definition at line 637 of file caps.h.

10.25.2.8 system_id

```
bool vxg::cloud::agent::proto::osd_caps::system_id
```

system_id: bool, True when OSD supports separate system_id enabling/disabling

Definition at line 618 of file caps.h.

10.25.2.9 system_id_text

```
bool vxg::cloud::agent::proto::osd_caps::system_id_text
```

system_id_text: bool, True when OSD supports separate system_id customization

Definition at line 621 of file caps.h.

10.25.2.10 time

```
bool vxg::cloud::agent::proto::osd_caps::time
```

time: bool, True when OSD supports separate time enabling/disabling

Definition at line 623 of file caps.h.

10.25.2.11 time_format

```
std::vector<time_format_n> vxg::cloud::agent::proto::osd_caps::time_format
```

time_format: list of string, supported time formats.

Empty list means – time format selection is not supported. Example: ["12h", "24h"]

Definition at line 627 of file caps.h.

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

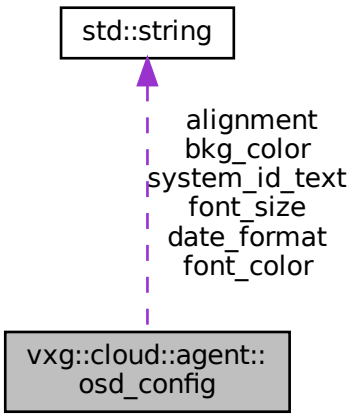
- caps.h

10.26 vxg::cloud::agent::osd_config Struct Reference

OSD config.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::osd_config:



Data Fields

- bool system_id

  system_id: optional bool, enable/disable static part of OSD
- **std::string** system_id_text

  system_id_text: optional string, a static content of OSD
- bool time

  time: optional bool, enable/disable time part of OSD
- time_format_n time_format

  time_format: optional string, one of predefined values from the time_format_n, should be included in caps.
- bool date

  date: optional bool, enable/disable date part of OSD
- **std::string** date_format

  date_format: optional string, one of predefined values from caps
- **std::string** font_size

  font_size: optional string, one of predefined font sizes from caps
- **std::string** font_color

  font_color: optional string, name of one of predefined font colors from caps
- **std::string** bkg_color

  bkg_color: optional string, name of one of predefined background colors from caps
- bool bkg_transp

  bkg_transp: optional bool, enable/disable OSD background transparency
- **std::string** alignment

  alignment: optional string, one of predefined positions from caps
- osd_caps caps

  OSD capabilities of the device.

10.26.1 Detailed Description

OSD config.

On Screen Display configuration object.

Definition at line 1137 of file config.h.

10.26.2 Field Documentation

10.26.2.1 alignment

```
std::string vxg::cloud::agent::osd_config::alignment
```

alignment: optional string, one of predefined positions from caps

Definition at line 1168 of file config.h.

10.26.2.2 bkg_color

```
std::string vxg::cloud::agent::osd_config::bkg_color
```

bkg_color: optional string, name of one of predefined background colors from caps

Definition at line 1164 of file config.h.

10.26.2.3 bkg_transp

```
bool vxg::cloud::agent::osd_config::bkg_transp
```

bkg_transp: optional bool, enable/disable OSD background transparency

Definition at line 1166 of file config.h.

10.26.2.4 caps

```
osd_caps vxg::cloud::agent::osd_config::caps
```

OSD capabilities of the device.

Definition at line 1171 of file config.h.

10.26.2.5 date

```
bool vxg::cloud::agent::osd_config::date
```

date: optional bool, enable/disable date part of OSD

Definition at line 1152 of file config.h.

10.26.2.6 date_format

```
std::string vxg::cloud::agent::osd_config::date_format
```

date_format: optional string, one of predefined values from caps

Definition at line 1155 of file config.h.

10.26.2.7 font_color

```
std::string vxg::cloud::agent::osd_config::font_color
```

font_color: optional string, name of one of predefined font colors from caps

Definition at line 1161 of file config.h.

10.26.2.8 font_size

```
std::string vxg::cloud::agent::osd_config::font_size
```

font_size: optional string, one of predefined font sizes from caps

Definition at line 1158 of file config.h.

10.26.2.9 system_id

```
bool vxg::cloud::agent::osd_config::system_id
```

system_id: optional bool, enable/disable static part of OSD

Definition at line 1140 of file config.h.

10.26.2.10 system_id_text

```
std::string vxg::cloud::agent::osd_config::system_id_text
```

system_id_text: optional string, a static content of OSD

Definition at line 1143 of file config.h.

10.26.2.11 time

```
bool vxg::cloud::agent::osd_config::time
```

time: optional bool, enable/disable time part of OSD

Definition at line 1146 of file config.h.

10.26.2.12 time_format

```
time_format_n vxg::cloud::agent::osd_config::time_format
```

time_format: optional string, one of predefined values from the time_format_n, should be included in caps.

Definition at line 1149 of file config.h.

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

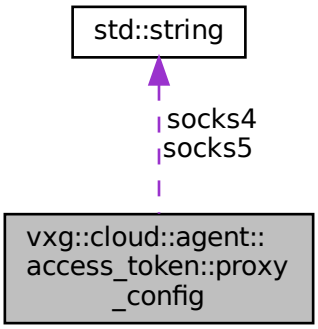
- config.h

10.27 vxg::cloud::agent::access_token::proxy_config Struct Reference

Socks proxy settings.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::access_token::proxy_config:



Data Fields

- **std::string** socks4
  SOCKS4 proxy uri.
- **std::string** socks5
  SOCKS5 proxy uri, ex. socks5://user:pwd@host:port.

10.27.1 Detailed Description

Socks proxy settings.

Definition at line 1197 of file config.h.

10.27.2 Field Documentation

10.27.2.1 socks4

```
std::string vxg::cloud::agent::access_token::proxy_config::socks4
```

SOCKS4 proxy uri.

Definition at line 1199 of file config.h.

10.27.2.2 socks5

```
std::string vxg::cloud::agent::access_token::proxy_config::socks5
```

SOCKS5 proxy uri, ex. socks5://user:pwd@host:port.

Definition at line 1201 of file config.h.

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

- config.h

10.28 vxg::cloud::agent::ptz_command Struct Reference

PTZ command.

```
#include <agent-proto/objects/config.h>
```

Data Fields

- ptz_action action

  action: string, Camera informs server about list of supported actions with 3.30 cam_ptz_conf (CM) command
- int tm

  tm: optional int, operation time that allows to make PTZ with specified steps, msec

10.28.1 Detailed Description

PTZ command.

Definition at line 1115 of file config.h.

10.28.2 Field Documentation

10.28.2.1 action

```
ptz_action vxg::cloud::agent::ptz_command::action
```

action: string, Camera informs server about list of supported actions with 3.30 cam_ptz_conf (CM) command

Definition at line 1119 of file config.h.

10.28.2.2 tm

```
int vxg::cloud::agent::ptz_command::tm
```

tm: optional int, operation time that allows to make PTZ with specified steps, msec

Definition at line 1123 of file config.h.

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

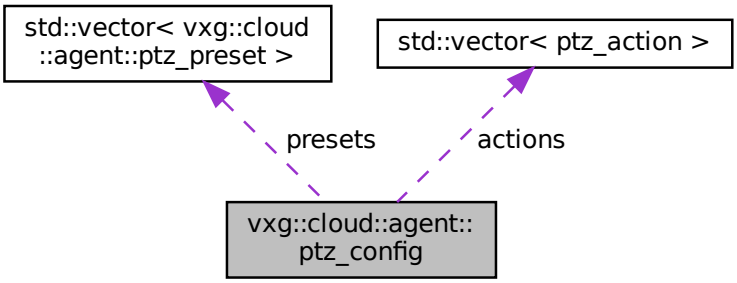
- config.h

10.29 vxg::cloud::agent::ptz_config Struct Reference

PTZ config.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::ptz_config:



Data Fields

- **std::vector**< ptz_action > actions

  actions: list of strings, list of supported PTZ actions.
- int maximum_number_of_presets

  maximum_number_of_presets: optional int, max number of supported presets when camera supports.
- **std::vector**< ptz_preset > presets

  presets: optional list of structures ptz_preset

10.29.1 Detailed Description

PTZ config.

Definition at line 1090 of file config.h.

10.29.2 Field Documentation

10.29.2.1 actions

```
std::vector<ptz_action> vxg::cloud::agent::ptz_config::actions
```

actions: list of strings, list of supported PTZ actions.

Possible values: “left”, “right”, “top”, “bottom”, “zoom_in”, “zoom_out”, “stop”. Server sends commands via 3.5 cam_ptz (SRV)

Definition at line 1094 of file config.h.

10.29.2.2 maximum_number_of_presets

```
int vxg::cloud::agent::ptz_config::maximum_number_of_presets
```

maximum_number_of_presets: optional int, max number of supported presets when camera supports.

Zero value, the missed parameter or missed or empty presets list are interpreted by server as “camera doesn't support PTZ”

Definition at line 1100 of file config.h.

10.29.2.3 presets

```
std::vector<ptz_preset> vxg::cloud::agent::ptz_config::presets
```

presets: optional list of structures ptz_preset

Definition at line 1103 of file config.h.

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

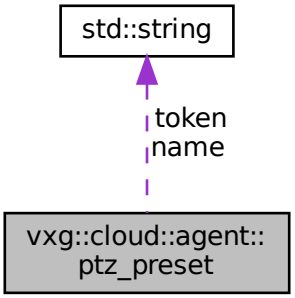
- config.h

10.30 vxg::cloud::agent::ptz_preset Struct Reference

PTZ preset.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::ptz_preset:



Data Fields

- **std::string** token

  token: string, an unique token of preset what is used for all operations with preset
- **std::string** name

  name: string, user friendly name of preset
- ptz_preset_action action

  actions: list of strings, required preset action.

10.30.1 Detailed Description

PTZ preset.

Definition at line 1072 of file config.h.

10.30.2 Field Documentation

10.30.2.1 action

```
ptz_preset_action vxg::cloud::agent::ptz_preset::action
```

actions: list of strings, required preset action.

Possible values: “create”, “delete”, “goto”, “update”

Definition at line 1081 of file config.h.

10.30.2.2 name

```
std::string vxg::cloud::agent::ptz_preset::name
```

name: string, user friendly name of preset

Definition at line 1077 of file config.h.

10.30.2.3 token

```
std::string vxg::cloud::agent::ptz_preset::token
```

token: string, an unique token of preset what is used for all operations with preset

Definition at line 1075 of file config.h.

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

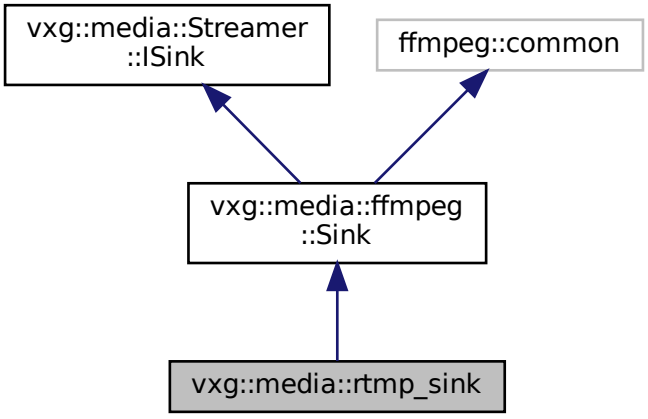
- config.h

10.31 vxg::media::rtmp_sink Class Reference

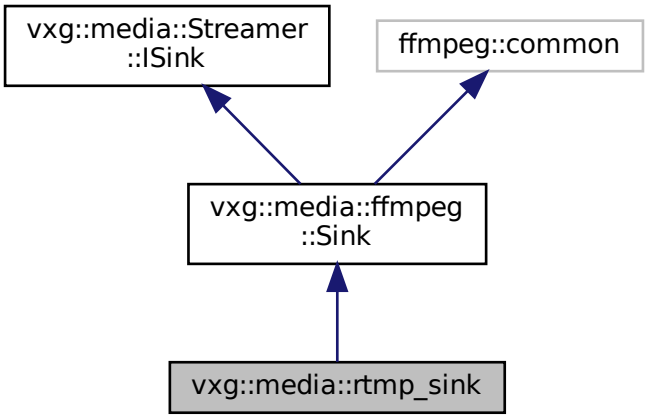
RTMP sink class.

`#include <streamer/rtmp_sink.h>`

Inheritance diagram for vxg::media::rtmp_sink:



Collaboration diagram for vxg::media::rtmp_sink:



Public Member Functions

- rtmp_sink (**std::function**< void(vxg::media::Streamer::StreamError)> cb)

  Construct a new rtmp sink object.
- virtual bool init (**std::string** url) override

  *Overriden vxg::media::ffmpeg::Sink::init(**std::string**, **std::string**) "init" method with hidden output ffmpeg format.*
- virtual void error (Streamer::StreamError stream_error) override

  Media processing error callback, called when ISink::process returned false or linked source's ISource::pullFrame returned false, or when ISource::error was called.
- virtual **std::string** name () override

  Sink name.
- virtual bool droppable () override

  If sink of with dropping its media frames.
- bool negotiate (**std::vector**< Streamer::StreamInfo > streams_info)

  Override negotiate() for removing all data streams.

Additional Inherited Members

10.31.1 Detailed Description

RTMP sink class.

Definition at line 13 of file rtmp_sink.h.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 rtmp_sink()

```
vxg::media::rtmp_sink::rtmp_sink (
            std::function< void(vxg::media::Streamer::StreamError)> cb )  [inline]
```

Construct a new rtmp sink object.

Parameters

in	*cb*	error callback

Definition at line 20 of file rtmp_sink.h.

10.31.3 Member Function Documentation

10.31.3.1 droppable()

```
virtual bool vxg::media::rtmp_sink::droppable ( )  [inline], [override], [virtual]
```

If sink of with dropping its media frames.

Returns

true Internal media thread allowed to drop frames if internal media queue is full.

false No media frames dropping allowed.

Reimplemented from vxg::media::ffmpeg::Sink.

Definition at line 47 of file rtmp_sink.h.

10.31.3.2 error()

```
virtual void vxg::media::rtmp_sink::error (
            Streamer::StreamError error )  [inline], [override], [virtual]
```

Media processing error callback, called when ISink::process returned false or linked source's ISource::pullFrame returned false, or when ISource::error was called.

Parameters

error	Error type.

Reimplemented from vxg::media::ffmpeg::Sink.

Definition at line 33 of file rtmp_sink.h.

10.31.3.3 init()

```
virtual bool vxg::media::rtmp_sink::init (
             std::string url )  [inline], [override], [virtual]
```

Overriden vxg::media::ffmpeg::Sink::init(**std::string**, **std::string**) "init" method with hidden output ffmpeg format.

Parameters

url	RTMP url

Returns

true On success

false On failure

Reimplemented from vxg::media::ffmpeg::Sink.

Definition at line 29 of file rtmp_sink.h.

10.31.3.4 name()

```
virtual std::string vxg::media::rtmp_sink::name ( ) [inline], [override], [virtual]
```

Sink name.

Returns

std::string

Reimplemented from vxg::media::ffmpeg::Sink.

Definition at line 45 of file rtmp_sink.h.

10.31.3.5 negotiate()

```
bool vxg::media::rtmp_sink::negotiate (
            std::vector< Streamer::StreamInfo > streams_info ) [inline], [virtual]
```

Override negotiate() for removing all data streams.

This is required for preventing buffering inside the ffmpeg muxer, ffmpeg waits for at least one packet for each stream or 10 seconds by default before output next chunk, this leads to 10 seconds delay if data track was added to output muxing context but no actual data packets were received hence sparse streams like onvif metadata may significantly increase delay.

Parameters

in	*streams_info*	- list of streams descrtiptions.

Returns

true

false

Reimplemented from vxg::media::ffmpeg::Sink.

Definition at line 60 of file rtmp_sink.h.

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

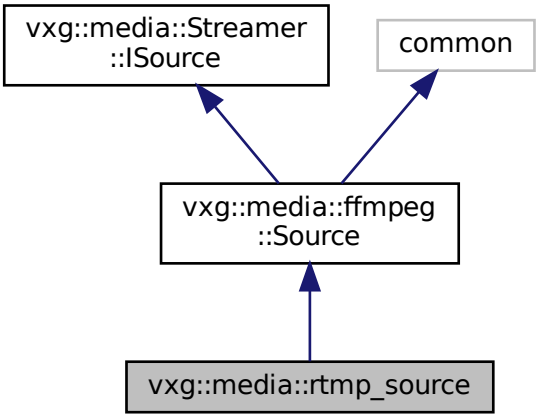
- rtmp_sink.h

10.32 vxg::media::rtmp_source Class Reference

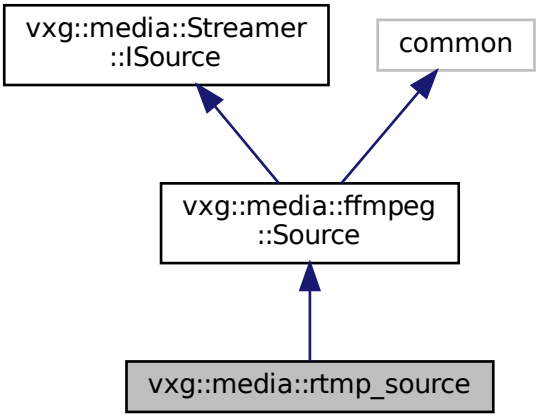
RTMP source class.

`#include <streamer/rtmp_source.h>`

Inheritance diagram for vxg::media::rtmp_source:



Collaboration diagram for vxg::media::rtmp_source:



Public Member Functions

- virtual bool init (**std::string** url)

  Init source with url.

Additional Inherited Members

10.32.1 Detailed Description

RTMP source class.

Definition at line 13 of file rtmp_source.h.

10.32.2 Member Function Documentation

10.32.2.1 init()

```
virtual bool vxg::media::rtmp_source::init (
              std::string url )  [inline], [virtual]
```

Init source with url.

Parameters

in	*url*	RTMP url

Returns

true Success

false Failed

Reimplemented from vxg::media::ffmpeg::Source.

Definition at line 24 of file rtmp_source.h.

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

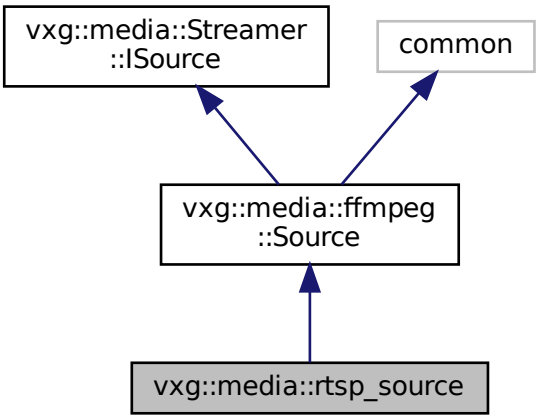
- rtmp_source.h

10.33 vxg::media::rtsp_source Class Reference

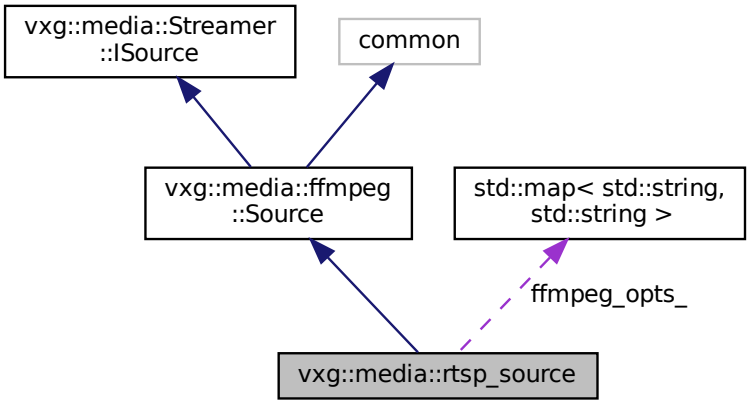
RTSP source class.

```
#include <streamer/rtsp_source.h>
```

Inheritance diagram for vxg::media::rtsp_source:



Collaboration diagram for vxg::media::rtsp_source:



Public Member Functions

- rtsp_source (bool rtp_over_tcp=true, **std::vector**< Streamer::MediaType > media_types= **std::vector**< Streamer::MediaType >(0))

  Construct a new rtsp source object.
- rtsp_source (**std::string** rtp_transport="tcp", std::vector< Streamer::MediaType > media_types= **std::vector**< Streamer::MediaType >(0), **std::map**< **std::string**, **std::string** > ffmpeg_opts={}, **std::chrono::seconds** timeout= **std::chrono::seconds**(0))

  Construct a new rtsp source object.

- virtual bool init (**std::string** url)

  Overloaded init method.
- virtual **std::string** name () override

  Source class name.

Protected Attributes

- **std::map**< **std::string**, **std::string** > ffmpeg_opts_

Additional Inherited Members

10.33.1 Detailed Description

RTSP source class.

Definition at line 13 of file rtsp_source.h.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 rtsp_source() [1/2]

```
vxg::media::rtsp_source::rtsp_source (
            bool rtp_over_tcp = true,
             std::vector< Streamer::MediaType > media_types =  std::vector<Streamer::MediaType>(0)
)  [inline]
```

Construct a new rtsp source object.

Parameters

in	*rtp_over_tcp*	Flag indicates if user wants RTP over TCP
in	*media_types*	List of media types to ask from RTSP server, can be used to filter out unnecessary tracks. If empty all types will be requested.

Definition at line 31 of file rtsp_source.h.

10.33.2.2 rtsp_source() [2/2]

```
vxg::media::rtsp_source::rtsp_source (
             std::string rtp_transport = "tcp",
             std::vector< Streamer::MediaType > media_types =  std::vector<Streamer::MediaType>(0),
```

```
std::map< std::string, std::string > ffmpeg_opts = {},
std::chrono::seconds timeout = std::chrono::seconds(0) )  [inline]
```

Construct a new rtsp source object.

Parameters

in	*rtp_transport*	RTP transport passed directly to ffmpeg.
in	*media_types*	List of media types to ask from RTSP server, can be used to filter out unnecessary tracks. If empty all types will be requested.
in	*ffmpeg_opts*	Map of ffmpeg options key values pairs.
in	*timeout*	RTSP client io timeout. Doesn't mean the connection will be closed after this timeout but specifies the amount of time ffmpeg spends in io loop spinning, infinite timeout causes spining forever if connection wasn't closed but no data was received.

Definition at line 51 of file rtsp_source.h.

10.33.3 Member Function Documentation

10.33.3.1 init()

```
virtual bool vxg::media::rtsp_source::init (
              std::string url )  [inline], [virtual]
```

Overloaded init method.

Parameters

in	*url*	RTSP URL link

Returns

true

false

Reimplemented from vxg::media::ffmpeg::Source.

Definition at line 69 of file rtsp_source.h.

10.33.3.2 name()

```
virtual  std::string vxg::media::rtsp_source::name ( )  [inline], [override], [virtual]
```

Source class name.

Returns

std::string

Reimplemented from vxg::media::ffmpeg::Source.

Definition at line 164 of file rtsp_source.h.

10.33.4 Field Documentation

10.33.4.1 ffmpeg_opts_

```
std::map< std::string, std::string> vxg::media::rtsp_source::ffmpeg_opts_ [protected]
```

Definition at line 22 of file rtsp_source.h.

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

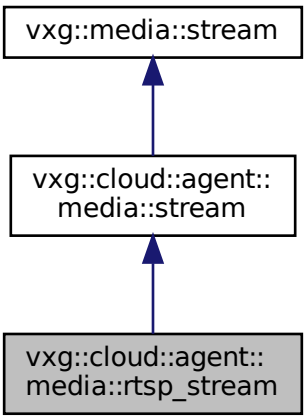
- rtsp_source.h

10.34 vxg::cloud::agent::media::rtsp_stream Class Reference

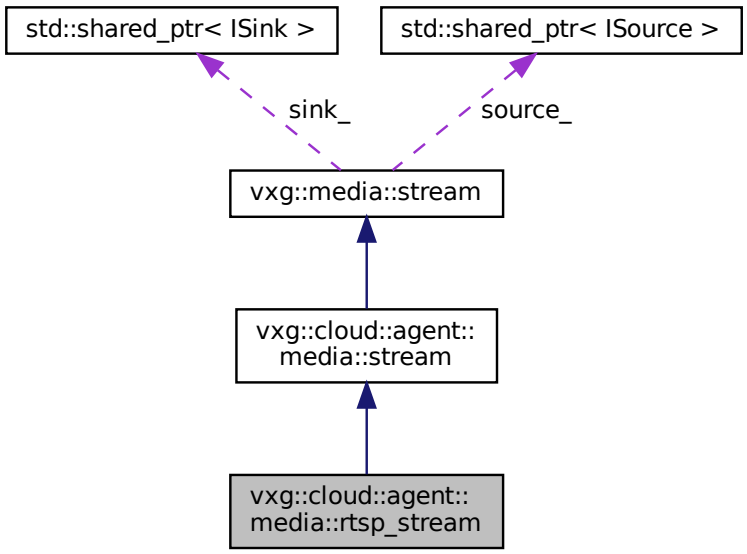
Implementation of the media::stream with RTSP source and NIY stubs.

```
#include <agent/rtsp-stream.h>
```

Inheritance diagram for vxg::cloud::agent::media::rtsp_stream:



Collaboration diagram for vxg::cloud::agent::media::rtsp_stream:



Public Types

- typedef **std::shared_ptr**< rtsp_stream > ptr

Public Member Functions

- rtsp_stream (**std::string** source_url, **std::string** name, bool rtp_transport_tcp=true, bool recorder_needs↩ _source=false)

  Construct a new rtsp stream object.
- virtual ~rtsp_stream ()
- virtual bool start (**std::string** not_used="")
- bool get_supported_stream (proto::supported_stream_config &config)
- virtual bool get_stream_caps (proto::stream_caps &caps) override

  Get the media stream caps.
- virtual bool get_stream_config (proto::stream_config &streamConfig)

  Get the stream config.
- virtual bool set_stream_config (const proto::stream_config &streamConfig)

  Set the streams config.
- virtual bool get_snapshot (proto::event_object::snapshot_info_object &snapshot)
- virtual **std::vector**< proto::video_clip_info > record_get_list (cloud::time begin, cloud::time end, bool align)

  Get list of the recorded clips for specific time period.
- virtual proto::video_clip_info record_export (cloud::time begin, cloud::time end)

  Export recorded clip for specified time.
- virtual bool start_record ()

  Start recording of this media stream.
- virtual bool stop_record ()

  Stop recording of this stream.

Additional Inherited Members

10.34.1 Detailed Description

Implementation of the media::stream with RTSP source and NIY stubs.

Definition at line 18 of file rtsp-stream.h.

10.34.2 Member Typedef Documentation

10.34.2.1 ptr

```
typedef  std::shared_ptr<rtsp_stream> vxg::cloud::agent::media::rtsp_stream::ptr
```

Definition at line 34 of file rtsp-stream.h.

10.34.3 Constructor & Destructor Documentation

10.34.3.1 rtsp_stream()

```
vxg::cloud::agent::media::rtsp_stream::rtsp_stream (
              std::string source_url,
              std::string name,
             bool rtp_transport_tcp = true,
             bool recorder_needs_source = false )  [inline]
```

Construct a new rtsp stream object.

Parameters

source_url	RTSP url
name	Unique stream name
rtp_transport_tcp	true - RTP over TCP; false - RTP over UDP
record_needs_source	Indicates if stream needs source start before calling start_record() virtual method.

Definition at line 43 of file rtsp-stream.h.

10.34.3.2 ~rtsp_stream()

```
virtual vxg::cloud::agent::media::rtsp_stream::~rtsp_stream ( )  [inline], [virtual]
```

Definition at line 60 of file rtsp-stream.h.

10.34.4 Member Function Documentation

10.34.4.1 get_snapshot()

```
virtual bool vxg::cloud::agent::media::rtsp_stream::get_snapshot (
            proto::event_object::snapshot_info_object & snapshot )  [inline], [virtual]
```

Definition at line 95 of file rtsp-stream.h.

10.34.4.2 get_stream_caps()

```
virtual bool vxg::cloud::agent::media::rtsp_stream::get_stream_caps (
            proto::stream_caps & caps )  [inline], [override], [virtual]
```

Get the media stream caps.

video/audio elementary streams caps request passes caps with names of the elementary streams for which caps are required to be filled inside this method

Parameters

out	*caps*	

Returns

true if `caps` valid

false if `caps` is invalid

Implements vxg::cloud::agent::media::stream.

Definition at line 77 of file rtsp-stream.h.

10.34.4.3 get_stream_config()

```
virtual bool vxg::cloud::agent::media::rtsp_stream::get_stream_config (
            proto::stream_config & config )  [inline], [virtual]
```

Get the stream config.

Parameters

in,out	*config*	input `config` contains list of streams for which configuration should be returned

Returns

true if `config` is valid

false if `config` is invalid

Implements vxg::cloud::agent::media::stream.

Definition at line 83 of file rtsp-stream.h.

10.34.4.4 get_supported_stream()

```
bool vxg::cloud::agent::media::rtsp_stream::get_supported_stream (
            proto::supported_stream_config & config )  [inline]
```

Definition at line 66 of file rtsp-stream.h.

10.34.4.5 record_export()

```
virtual proto::video_clip_info vxg::cloud::agent::media::rtsp_stream::record_export (
            cloud::time begin,
            cloud::time end )  [inline], [virtual]
```

Export recorded clip for specified time.

Parameters

begin	
end	

Returns

proto::video_clip_info

Implements vxg::cloud::agent::media::stream.

Definition at line 110 of file rtsp-stream.h.

10.34.4.6 record_get_list()

```
virtual  std::vector<proto::video_clip_info> vxg::cloud::agent::media::rtsp_stream::record_↩
get_list (
            cloud::time begin,
            cloud::time end,
            bool align )  [inline], [virtual]
```

Get list of the recorded clips for specific time period.

Parameters

in	*begin*	beginning of the time period
in	*end*	ending of the time period
in	*align*	Align returned records to key frames and begin/end. If true the implementation should align returned records to not include data with timestamps less than `begin` and greater than `end`. Also any returned record MUST start with key frame and the last frame of any not last record in the list MUST be the frame prior to key frame - first frame of the next record.
in	*limit*	Max records number that may be returned. Value 0 means no limitation.

Returns

std::vector<proto::video_clip_info>

Implements vxg::cloud::agent::media::stream.

Definition at line 103 of file rtsp-stream.h.

10.34.4.7 set_stream_config()

```
virtual bool vxg::cloud::agent::media::rtsp_stream::set_stream_config (
            const proto::stream_config & config )  [inline], [virtual]
```

Set the streams config.

Parameters

in	*config*	input `config` contains list of streams for which configuration should be set

Returns

true if `config` successfully set

false if `config` failed to set

Implements vxg::cloud::agent::media::stream.

Definition at line 89 of file rtsp-stream.h.

10.34.4.8 start()

```
virtual bool vxg::cloud::agent::media::rtsp_stream::start (
             std::string not_used = "" )  [inline], [virtual]
```

Reimplemented from vxg::media::stream.

Definition at line 62 of file rtsp-stream.h.

10.34.4.9 start_record()

```
virtual bool vxg::cloud::agent::media::rtsp_stream::start_record ( )  [inline], [virtual]
```

Start recording of this media stream.

Called only if memory card is presented and can be used.

Returns

true if recording started

false if recording start failed

See also

agent::event_stream::on_get_memorycard_info

Implements vxg::cloud::agent::media::stream.

Definition at line 118 of file rtsp-stream.h.

10.34.4.10 stop_record()

```
virtual bool vxg::cloud::agent::media::rtsp_stream::stop_record ( )  [inline], [virtual]
```

Stop recording of this stream.

Returns

true Stopped

false Failed to stop

Implements vxg::cloud::agent::media::stream.

Definition at line 124 of file rtsp-stream.h.

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

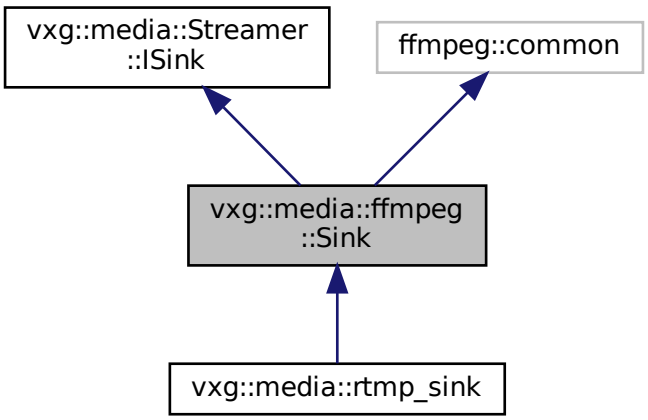
- rtsp-stream.h

10.35 vxg::media::ffmpeg::Sink Class Reference

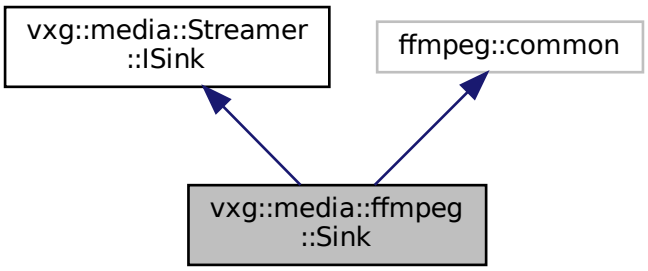
Base ffmpeg sink class.

`#include <streamer/ffmpeg_sink.h>`

Inheritance diagram for vxg::media::ffmpeg::Sink:



Collaboration diagram for vxg::media::ffmpeg::Sink:



Public Member Functions

- Sink ()
- virtual ~Sink ()
- bool init (**std::string** url, **std::string** fmt, **std::shared_ptr**< **std::vector**< uint8_t >> data_buffer=nullptr)

  Sink init.

- virtual bool init (**std::string** url="")

  Init sink.
- virtual bool finit ()

  Deinit sink.
- virtual void stop ()
- virtual void error (Streamer::StreamError stream_error)

  Media processing error callback, called when ISink::process returned false or linked source's ISource::pullFrame returned false, or when ISource::error was called.
- virtual **std::string** name ()

  Sink name.
- virtual bool droppable ()

  If sink of with dropping its media frames.
- virtual bool negotiate (**std::vector**< Streamer::StreamInfo >)

  Negotiation callback, this method called with collected from the ISource::negotiate media stream description.
- virtual cloud::duration duration ()

  Processed stream duration.

Additional Inherited Members

10.35.1 Detailed Description

Base ffmpeg sink class.

Definition at line 12 of file ffmpeg_sink.h.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 Sink()

```
vxg::media::ffmpeg::Sink::Sink ( )
```

10.35.2.2 ~Sink()

```
virtual vxg::media::ffmpeg::Sink::~Sink ( )  [virtual]
```

10.35.3 Member Function Documentation

10.35.3.1 droppable()

```
virtual bool vxg::media::ffmpeg::Sink::droppable ( )  [inline], [virtual]
```

If sink of with dropping its media frames.

Returns

true Internal media thread allowed to drop frames if internal media queue is full.

false No media frames dropping allowed.

Implements vxg::media::Streamer::ISink.

Reimplemented in vxg::media::rtmp_sink.

Definition at line 55 of file ffmpeg_sink.h.

10.35.3.2 duration()

```
virtual cloud::duration vxg::media::ffmpeg::Sink::duration ( )  [inline], [virtual]
```

Processed stream duration.

Returns

duration

Reimplemented from vxg::media::Streamer::ISink.

Definition at line 57 of file ffmpeg_sink.h.

10.35.3.3 error()

```
virtual void vxg::media::ffmpeg::Sink::error (
            Streamer::StreamError error )  [inline], [virtual]
```

Media processing error callback, called when ISink::process returned false or linked source's ISource::pullFrame returned false, or when ISource::error was called.

Parameters

error	Error type.

Implements vxg::media::Streamer::ISink.

Reimplemented in vxg::media::rtmp_sink.

Definition at line 33 of file ffmpeg_sink.h.

10.35.3.4 finit()

```
virtual bool vxg::media::ffmpeg::Sink::finit ( )  [virtual]
```

Deinit sink.

Returns

true finit success.

false finit failed.

Implements vxg::media::Streamer::ISink.

10.35.3.5 init() [1/2]

```
bool vxg::media::ffmpeg::Sink::init (
            std::string url,
            std::string fmt,
            std::shared_ptr<  std::vector< uint8_t >> data_buffer = nullptr )
```

Sink init.

Parameters

url	Output url
fmt	Output format
data_buffer	Output buffer for output to memory, if specified and not nullptr the `url` will be ignored.

Returns

true On success

false On failure

10.35.3.6 init() [2/2]

```
virtual bool vxg::media::ffmpeg::Sink::init (
            std::string url = "" )  [virtual]
```

Init sink.

Parameters

in	*url*	Url if needed.

Returns

true init success.

false init failed.

Implements vxg::media::Streamer::ISink.

Reimplemented in vxg::media::rtmp_sink.

10.35.3.7 name()

```
virtual  std::string vxg::media::ffmpeg::Sink::name ( )  [inline], [virtual]
```

Sink name.

Returns

std::string

Implements vxg::media::Streamer::ISink.

Reimplemented in vxg::media::rtmp_sink.

Definition at line 53 of file ffmpeg_sink.h.

10.35.3.8 negotiate()

```
virtual bool vxg::media::ffmpeg::Sink::negotiate (
              std::vector< Streamer::StreamInfo > info )  [virtual]
```

Negotiation callback, this method called with collected from the ISource::negotiate media stream description.

Parameters

info	List of elementary streams descriptions.

Returns

true If streams descriptions accepted.

false Streams not accepted, will cause media thread stopping.

Reimplemented from vxg::media::Streamer::ISink.

Reimplemented in vxg::media::rtmp_sink.

10.35.3.9 stop()

```
virtual void vxg::media::ffmpeg::Sink::stop ( )  [virtual]
```

Reimplemented from vxg::media::Streamer::ISink.

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

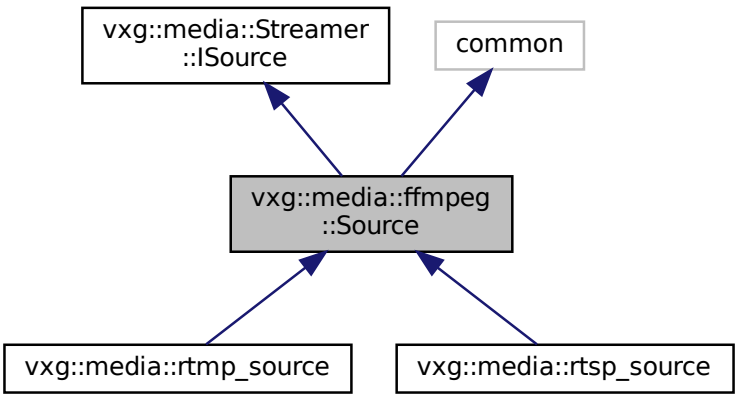
- ffmpeg_sink.h

10.36 vxg::media::ffmpeg::Source Class Reference

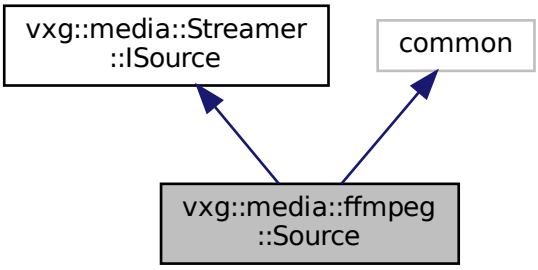
Base ffmpeg source class.

```
#include <streamer/ffmpeg_source.h>
```

Inheritance diagram for vxg::media::ffmpeg::Source:



Collaboration diagram for vxg::media::ffmpeg::Source:



Public Member Functions

- Source ()
- virtual ~Source ()
- bool init (**std::string** url, AVDictionary *opts, **std::string** fmt="")

  Init ffmpeg source with specific ffmpeg options.
- bool init (**std::shared_ptr**< **std::vector**< uint8_t >> input_buffer, AVDictionary *opts, **std::string** fmt)

  Init ffmpeg memory source with specific ffmpeg options.
- virtual bool init (**std::string** url="")

  Init source.
- virtual void finit ()

  Finit souce.
- virtual **std::shared_ptr**< Streamer::MediaFrame > pullFrame ()

  Main method of the Mode::PULL mode data producing.
- virtual **std::string** name ()

  Source class name.
- virtual **std::vector**< Streamer::StreamInfo > negotiate ()

  Negotiation callback.
- virtual void stop ()

Additional Inherited Members

10.36.1 Detailed Description

Base ffmpeg source class.

Definition at line 10 of file ffmpeg_source.h.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 Source()

```
vxg::media::ffmpeg::Source::Source ( )
```

Definition at line 9 of file ffmpeg_source.cc.

10.36.2.2 ~Source()

```
vxg::media::ffmpeg::Source::~Source ( )  [virtual]
```

Definition at line 12 of file ffmpeg_source.cc.

10.36.3 Member Function Documentation

10.36.3.1 finit()

```
void vxg::media::ffmpeg::Source::finit ( )  [virtual]
```

Finit souce.

Implements vxg::media::Streamer::ISource.

Definition at line 30 of file ffmpeg_source.cc.

10.36.3.2 init() [1/3]

```
bool vxg::media::ffmpeg::Source::init (
              std::shared_ptr<  std::vector< uint8_t >> input_buffer,
             AVDictionary * opts,
              std::string fmt )
```

Init ffmpeg memory source with specific ffmpeg options.

Parameters

in	*input_buffer*	Input memory buffer containing whole media.
in	*opts*	ffmpeg options
in	*fmt*	ffmpeg input format to prevent auto-detection. ex.: "flv", "mp4", "http" etc.

Returns

true

false

Definition at line 20 of file ffmpeg_source.cc.

10.36.3.3 init() [2/3]

```
bool vxg::media::ffmpeg::Source::init (
              std::string url,
             AVDictionary * opts,
              std::string fmt = "" )
```

Init ffmpeg source with specific ffmpeg options.

Parameters

in	*url*	Url
in	*opts*	ffmpeg options
in	*fmt*	ffmpeg input format to prevent auto-detection. ex.: "flv", "rtsp", "http" etc.

Returns

true

false

Definition at line 14 of file ffmpeg_source.cc.

10.36.3.4 init() [3/3]

```
bool vxg::media::ffmpeg::Source::init (
              std::string url = "" )  [virtual]
```

Init source.

Parameters

url	Url if needed.

Returns

true Init success.

false Init failed.

Implements vxg::media::Streamer::ISource.

Reimplemented in vxg::media::rtsp_source, and vxg::media::rtmp_source.

Definition at line 26 of file ffmpeg_source.cc.

10.36.3.5 name()

```
virtual std::string vxg::media::ffmpeg::Source::name ( ) [inline], [virtual]
```

Source class name.

Returns

std::string

Implements vxg::media::Streamer::ISource.

Reimplemented in vxg::media::rtsp_source.

Definition at line 42 of file ffmpeg_source.h.

10.36.3.6 negotiate()

```
std::vector< Streamer::StreamInfo > vxg::media::ffmpeg::Source::negotiate ( ) [virtual]
```

Negotiation callback.

Called by internals. Class implementation should return the list of the streams info source will be producing for the sinks, this list will be then passed to the ISink::negotiate method.

Returns

std::vector<Streamer::StreamInfo>

Implements vxg::media::Streamer::ISource.

Definition at line 34 of file ffmpeg_source.cc.

10.36.3.7 pullFrame()

```
std::shared_ptr< Streamer::MediaFrame > vxg::media::ffmpeg::Source::pullFrame ( ) [virtual]
```

Main method of the Mode::PULL mode data producing.

Called by internals if the source operation mode is Mode::PULL. Implementation should return media frame object with correctly filled fields.

Returns

std::shared_ptr<MediaFrame>

Implements vxg::media::Streamer::ISource.

Definition at line 93 of file ffmpeg_source.cc.

10.36.3.8 stop()

```
void vxg::media::ffmpeg::Source::stop ( )  [virtual]
```

Reimplemented from vxg::media::Streamer::ISource.

Definition at line 186 of file ffmpeg_source.cc.

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

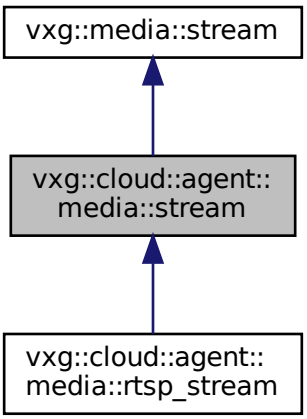
- ffmpeg_source.h
- ffmpeg_source.cc

10.37 vxg::cloud::agent::media::stream Class Reference

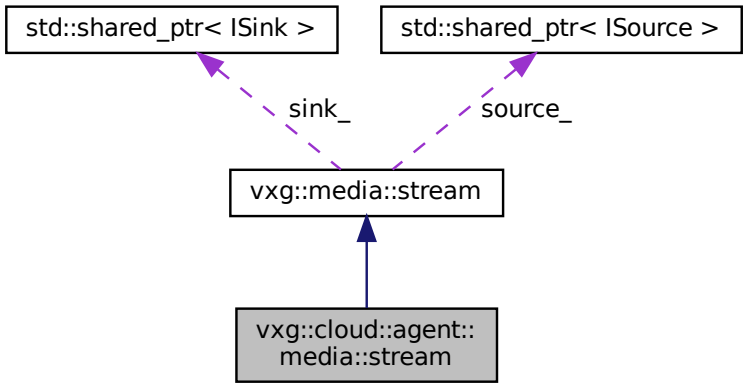
Cloud agent media stream abstract class.

```
#include <agent/stream.h>
```

Inheritance diagram for vxg::cloud::agent::media::stream:



Collaboration diagram for vxg::cloud::agent::media::stream:



Public Types

- typedef **std::shared_ptr**< stream > ptr

  std::shared_ptr *to the base_stream*

Public Member Functions

- stream (**std::string** name, vxg::media::Streamer::ISource::ptr source, **std::function**< void(vxg::media::Streamer::StreamErro
  sink_error_cb, bool recorder_needs_source=false)

  Construct a new agent media stream object.
- virtual ~stream ()
- virtual bool get_stream_caps (cloud::agent::proto::stream_caps &caps)=0

  Get the media stream caps.
- virtual bool get_supported_stream (cloud::agent::proto::supported_stream_config &supported_stream)=0

  Get the supported stream description.
- virtual bool get_stream_config (cloud::agent::proto::stream_config &config)=0

  Get the stream config.
- virtual bool set_stream_config (const cloud::agent::proto::stream_config &config)=0

  Set the streams config.
- virtual bool get_snapshot (cloud::agent::proto::event_object::snapshot_info_object &snapshot)=0

  Get the snapshot image of this media stream.
- virtual bool record_needs_source ()

  Should returns true if agent::manager should start stream source before calling start_record()
- virtual bool start_record ()=0

  Start recording of this media stream.
- virtual bool stop_record ()=0

  Stop recording of this stream.
- virtual **std::vector**< cloud::agent::proto::video_clip_info > record_get_list (cloud::time begin, cloud::time
  end, bool align=true)=0

  Get list of the recorded clips for specific time period.
- virtual cloud::agent::proto::video_clip_info record_export (cloud::time begin, cloud::time end)=0

  Export recorded clip for specified time.

Additional Inherited Members

10.37.1 Detailed Description

Cloud agent media stream abstract class.

vxg::media::stream derived class with VXG Cloud proto callbacks

Definition at line 21 of file agent/stream.h.

10.37.2 Member Typedef Documentation

10.37.2.1 ptr

```
typedef  std::shared_ptr<stream> vxg::cloud::agent::media::stream::ptr
```

std::shared_ptr to the base_stream

Definition at line 29 of file agent/stream.h.

10.37.3 Constructor & Destructor Documentation

10.37.3.1 stream()

```
vxg::cloud::agent::media::stream::stream (
              std::string name,
             vxg::media::Streamer::ISource::ptr source,
              std::function< void(vxg::media::Streamer::StreamError)> sink_error_cb,
             bool recorder_needs_source = false )  [inline]
```

Construct a new agent media stream object.

Parameters

in	*name*	Unique stream name which will be used by the VXG Cloud API
in	*source*	Source object pointer
in	*sink_error_cb*	Callback which will be called on sink error
in	*recorder_needs_source*	Indicates if stream needs source start before calling start_record() virtual method.

Definition at line 39 of file agent/stream.h.

10.37.3.2 ~stream()

```
virtual vxg::cloud::agent::media::stream::~stream ( )  [inline], [virtual]
```

Reimplemented from vxg::media::stream.

Definition at line 48 of file agent/stream.h.

10.37.4 Member Function Documentation

10.37.4.1 get_snapshot()

```
virtual bool vxg::cloud::agent::media::stream::get_snapshot (
            cloud::agent::proto::event_object::snapshot_info_object & snapshot )  [pure virtual]
```

Get the snapshot image of this media stream.

Parameters

out	*snapshot*	snapshot object

Returns

true if `snapshot` is valid

false if `snapshot` is invalid

10.37.4.2 get_stream_caps()

```
virtual bool vxg::cloud::agent::media::stream::get_stream_caps (
            cloud::agent::proto::stream_caps & caps )  [pure virtual]
```

Get the media stream caps.

video/audio elementary streams caps request passes caps with names of the elementary streams for which caps are required to be filled inside this method

Parameters

out	*caps*	

Returns

true if `caps` valid

false if `caps` is invalid

Implemented in vxg::cloud::agent::media::rtsp_stream.

10.37.4.3 get_stream_config()

```
virtual bool vxg::cloud::agent::media::stream::get_stream_config (
            cloud::agent::proto::stream_config & config )  [pure virtual]
```

Get the stream config.

Parameters

in,out	*config*	input config contains list of streams for which configuration should be returned

Returns

true if config is valid

false if config is invalid

Implemented in vxg::cloud::agent::media::rtsp_stream.

10.37.4.4 get_supported_stream()

```
virtual bool vxg::cloud::agent::media::stream::get_supported_stream (
            cloud::agent::proto::supported_stream_config & supported_stream )  [pure virtual]
```

Get the supported stream description.

Parameters

out	*supported_stream*	Stream supported by device

Returns

true if supported_stream is valid

false if supported_stream is not valid

10.37.4.5 record_export()

```
virtual cloud::agent::proto::video_clip_info vxg::cloud::agent::media::stream::record_export (
            cloud::time begin,
            cloud::time end )  [pure virtual]
```

Export recorded clip for specified time.

Parameters

begin	
end	

Returns

proto::video_clip_info

Implemented in vxg::cloud::agent::media::rtsp_stream.

10.37.4.6 record_get_list()

```
virtual std::vector<cloud::agent::proto::video_clip_info> vxg::cloud::agent::media::stream↩
::record_get_list (
            cloud::time begin,
            cloud::time end,
            bool align = true ) [pure virtual]
```

Get list of the recorded clips for specific time period.

Parameters

in	*begin*	beginning of the time period
in	*end*	ending of the time period
in	*align*	Align returned records to key frames and begin/end. If true the implementation should align returned records to not include data with timestamps less than `begin` and greater than `end`. Also any returned record MUST start with key frame and the last frame of any not last record in the list MUST be the frame prior to key frame - first frame of the next record.
in	*limit*	Max records number that may be returned. Value 0 means no limitation.

Returns

std::vector<proto::video_clip_info>

Implemented in vxg::cloud::agent::media::rtsp_stream.

10.37.4.7 record_needs_source()

```
virtual bool vxg::cloud::agent::media::stream::record_needs_source ( ) [inline], [virtual]
```

Should returns true if agent::manager should start stream source before calling start_record()

Returns

true agent::manager should start stream source

false agent::manager may not start stream source

Definition at line 104 of file agent/stream.h.

10.37.4.8 set_stream_config()

```
virtual bool vxg::cloud::agent::media::stream::set_stream_config (
            const cloud::agent::proto::stream_config & config )  [pure virtual]
```

Set the streams config.

Parameters

in	*config*	input `config` contains list of streams for which configuration should be set

Returns

true if `config` successfully set

false if `config` failed to set

Implemented in vxg::cloud::agent::media::rtsp_stream.

10.37.4.9 start_record()

```
virtual bool vxg::cloud::agent::media::stream::start_record ( )  [pure virtual]
```

Start recording of this media stream.

Called only if memory card is presented and can be used.

Returns

true if recording started

false if recording start failed

See also

agent::event_stream::on_get_memorycard_info

Implemented in vxg::cloud::agent::media::rtsp_stream.

10.37.4.10 stop_record()

```
virtual bool vxg::cloud::agent::media::stream::stop_record ( )  [pure virtual]
```

Stop recording of this stream.

Returns

true Stopped

false Failed to stop

Implemented in vxg::cloud::agent::media::rtsp_stream.

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

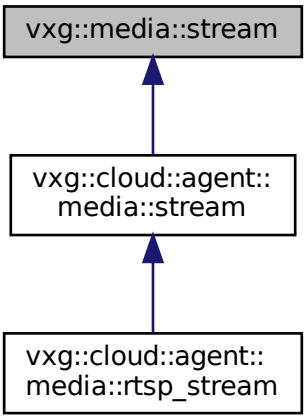
- agent/stream.h

10.38 vxg::media::stream Class Reference

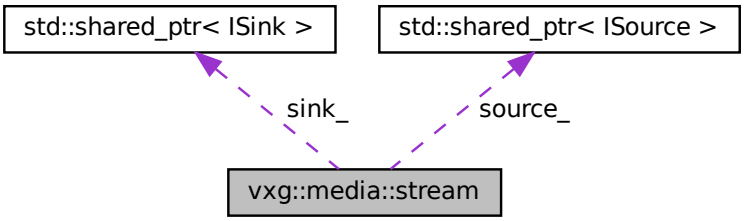
base media stream abstract class

`#include <streamer/stream.h>`

Inheritance diagram for vxg::media::stream:



Collaboration diagram for vxg::media::stream:



Public Types

- typedef **std::shared_ptr**< stream > ptr

  std::shared_ptr *to the base_stream*

Public Member Functions

- stream (**std::string** name, Streamer::ISource::ptr source, Streamer::ISink::ptr sink)

  Construct a new base stream object.
- virtual ~stream ()
- virtual bool init_source (**std::string** url)

  Initialize the source.
- virtual void finit_source ()

  Deinitialize source.
- virtual bool init_sink (**std::string** uri)

  Init media sink.
- virtual void finit_sink ()

  Deinitialize sink.

Protected Attributes

- Streamer::ISource::ptr source_

  media source
- Streamer::ISink::ptr sink_

  media sink

10.38.1 Detailed Description

base media stream abstract class

Media stream is the class representing media stream retranslation from the media source derived from the Streamer::ISource to the media sink derived from the Streamer::ISink. For instance, media stream could be a pair of `RTSP` source and `RTMP` sink, i.e. such media stream will be a retranslator of the `RTSP` stream to the `RTMP`

Definition at line 22 of file streamer/stream.h.

10.38.2 Member Typedef Documentation

10.38.2.1 ptr

```
typedef  std::shared_ptr<stream> vxg::media::stream::ptr
```

std::shared_ptr to the base_stream

Definition at line 27 of file streamer/stream.h.

10.38.3 Constructor & Destructor Documentation

10.38.3.1 stream()

```
vxg::media::stream::stream (
             std::string name,
            Streamer::ISource::ptr source,
            Streamer::ISink::ptr sink )  [inline]
```

Construct a new base stream object.

Parameters

name	Unique stream name which will be used by the VXG Cloud API
source	Source object pointer
sink	Sink object pointer

Definition at line 34 of file streamer/stream.h.

10.38.3.2 ~stream()

```
virtual vxg::media::stream::~stream ( )  [inline], [virtual]
```

Reimplemented in vxg::cloud::agent::media::stream.

Definition at line 44 of file streamer/stream.h.

10.38.4 Member Function Documentation

10.38.4.1 finit_sink()

```
virtual void vxg::media::stream::finit_sink ( )  [inline], [virtual]
```

Deinitialize sink.

Derived class deinitialize and deallocates base_stream::sink_

Definition at line 93 of file streamer/stream.h.

10.38.4.2 finit_source()

```
virtual void vxg::media::stream::finit_source ( )  [inline], [virtual]
```

Deinitialize source.

Definition at line 66 of file streamer/stream.h.

10.38.4.3 init_sink()

```
virtual bool vxg::media::stream::init_sink (
              std::string uri )  [inline], [virtual]
```

Init media sink.

Derived class should allocate and initialize base_stream::sink_ with RTMP sink publishing media stream to the RTMP server pointed by the uri

Parameters

in	*uri*	sink stream url if needed

Returns

true Sink started

false Sink start failed

Definition at line 80 of file streamer/stream.h.

10.38.4.4 init_source()

```
virtual bool vxg::media::stream::init_source (
              std::string url )  [inline], [virtual]
```

Initialize the source.

Called by the internal code, derived class should allocate and set base_stream::source_ with Streamer::ISink derived object pointer.

Parameters

url	source url

Returns

true if successfully initialized source

false if source initialization failed

Definition at line 56 of file streamer/stream.h.

10.38.5 Field Documentation

10.38.5.1 sink_

```
Streamer::ISink::ptr vxg::media::stream::sink_  [protected]
```

media sink

Definition at line 201 of file streamer/stream.h.

10.38.5.2 source_

```
Streamer::ISource::ptr vxg::media::stream::source_  [protected]
```

media source

Definition at line 199 of file streamer/stream.h.

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

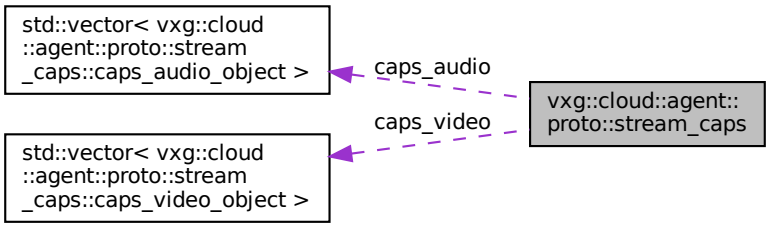
- streamer/stream.h

10.39 vxg::cloud::agent::proto::stream_caps Struct Reference

Media stream capabilites.

```
#include <agent-proto/objects/caps.h>
```

Collaboration diagram for vxg::cloud::agent::proto::stream_caps:



Data Structures

- struct caps_audio_object

  Audio streams capabilities.
- struct caps_video_object

  Video streams capabilities.

Data Fields

- **std::vector**< caps_video_object > caps_video

  List of video streams capabilities.
- **std::vector**< caps_audio_object > caps_audio

  List of audio streams capabilities.

10.39.1 Detailed Description

Media stream capabilites.

Definition at line 175 of file caps.h.

10.39.2 Field Documentation

10.39.2.1 caps_audio

```
std::vector<caps_audio_object> vxg::cloud::agent::proto::stream_caps::caps_audio
```

List of audio streams capabilities.

Definition at line 276 of file caps.h.

10.39.2.2 caps_video

```
std::vector<caps_video_object> vxg::cloud::agent::proto::stream_caps::caps_video
```

List of video streams capabilities.

Definition at line 274 of file caps.h.

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

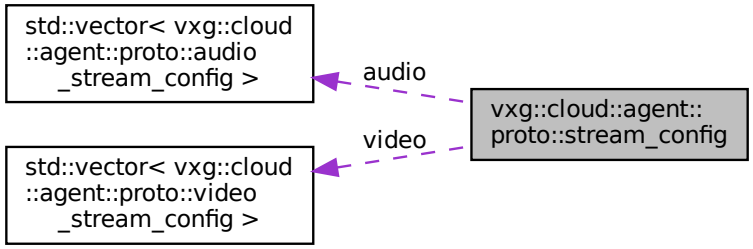
- caps.h

10.40 vxg::cloud::agent::proto::stream_config Struct Reference

Media stream config.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::proto::stream_config:



Data Fields

- **std::vector**< video_stream_config > video

  List of video media stream configs.
- **std::vector**< audio_stream_config > audio

  List of audio media stream configs.

10.40.1 Detailed Description

Media stream config.

Definition at line 222 of file config.h.

10.40.2 Field Documentation

10.40.2.1 audio

```
std::vector<audio_stream_config> vxg::cloud::agent::proto::stream_config::audio
```

List of audio media stream configs.

Definition at line 226 of file config.h.

10.40.2.2 video

```
std::vector<video_stream_config> vxg::cloud::agent::proto::stream_config::video
```

List of video media stream configs.

Definition at line 224 of file config.h.

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

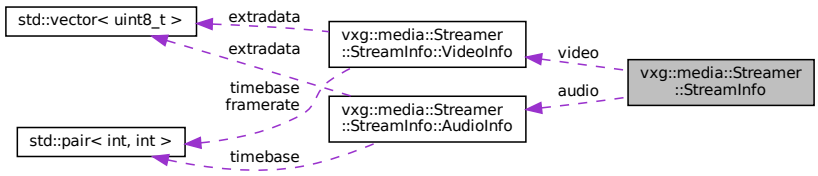
- config.h

10.41 vxg::media::Streamer::StreamInfo Struct Reference

Stream info descriotion.

`#include <streamer/base_streamer.h>`

Collaboration diagram for vxg::media::Streamer::StreamInfo:



Data Structures

- struct AudioInfo

  Audio stream info.
- struct VideoInfo

  Video stream info.

Public Types

- enum StreamType { ST_UNKNOWN, ST_VIDEO, ST_AUDIO, ST_DATA, ST_ANY }

  Stream type.
- enum VideoCodec { VC_UNKNOWN, VC_H264 }

  Video codec type.
- enum AudioCodec { AC_UNKNOWN, AC_AAC, AC_G711_U, AC_G711_A, AC_LPCM, AC_G726, AC_OPUS }

  Audio codec.
- enum DataCodec { DC_UNKNOWN, DC_ONVIF }

  Data codec.

Data Fields

- StreamType type

  Stream type.
- VideoInfo video

  Video stream info. Should be filled if stream type is ST_VIDEO.
- AudioInfo audio

  Audio stream info. Should be filled if stream type is ST_AUDIO.

10.41.1 Detailed Description

Stream info descriotion.

Definition at line 296 of file base_streamer.h.

10.41.2 Member Enumeration Documentation

10.41.2.1 AudioCodec

enum vxg::media::Streamer::StreamInfo::AudioCodec

Audio codec.

Enumerator

AC_UNKNOWN	
AC_AAC	
AC_G711_U	
AC_G711_A	
AC_LPCM	
AC_G726	
AC_OPUS	

Definition at line 336 of file base_streamer.h.

10.41.2.2 DataCodec

enum vxg::media::Streamer::StreamInfo::DataCodec

Data codec.

Enumerator

DC_UNKNOWN	
DC_ONVIF	

Definition at line 369 of file base_streamer.h.

10.41.2.3 StreamType

enum vxg::media::Streamer::StreamInfo::StreamType

Stream type.

Enumerator

ST_UNKNOWN	
ST_VIDEO	
ST_AUDIO	
ST_DATA	
ST_ANY	

Definition at line 298 of file base_streamer.h.

10.41.2.4 VideoCodec

`enum vxg::media::Streamer::StreamInfo::VideoCodec`

Video codec type.

Enumerator

VC_UNKNOWN	
VC_H264	

Definition at line 301 of file base_streamer.h.

10.41.3 Field Documentation

10.41.3.1 audio

`AudioInfo vxg::media::Streamer::StreamInfo::audio`

Audio stream info. Should be filled if stream type is ST_AUDIO.

Definition at line 384 of file base_streamer.h.

10.41.3.2 type

`StreamType vxg::media::Streamer::StreamInfo::type`

Stream type.

Definition at line 380 of file base_streamer.h.

10.41.3.3 video

`VideoInfo vxg::media::Streamer::StreamInfo::video`

Video stream info. Should be filled if stream type is ST_VIDEO.

Definition at line 382 of file base_streamer.h.

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

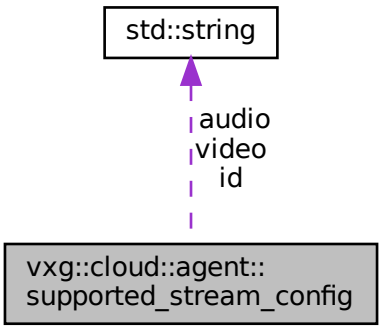
- base_streamer.h

10.42 vxg::cloud::agent::supported_stream_config Struct Reference

Supported stream config.

`#include <agent-proto/objects/config.h>`

Collaboration diagram for vxg::cloud::agent::supported_stream_config:



Data Fields

- **std::string** id

  id: string, name of media stream, unique for the camera
- **std::string** video

  video: optional string, video ES that is sent in this media stream
- **std::string** audio

  audio: optional string, audio ES that is sent in this media stream

10.42.1 Detailed Description

Supported stream config.

Definition at line 1270 of file config.h.

10.42.2 Field Documentation

10.42.2.1 audio

```
std::string vxg::cloud::agent::supported_stream_config::audio
```

audio: optional string, audio ES that is sent in this media stream

Definition at line 1276 of file config.h.

10.42.2.2 id

```
std::string vxg::cloud::agent::supported_stream_config::id
```

id: string, name of media stream, unique for the camera

Definition at line 1272 of file config.h.

10.42.2.3 video

```
std::string vxg::cloud::agent::supported_stream_config::video
```

video: optional string, video ES that is sent in this media stream

Definition at line 1274 of file config.h.

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

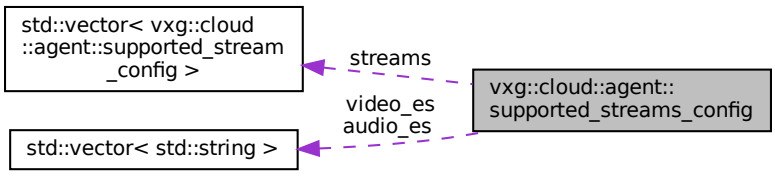
- config.h

10.43 vxg::cloud::agent::supported_streams_config Struct Reference

Supported streams config, list of supported_stream_config.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::supported_streams_config:



Data Fields

- **std::vector**< supported_stream_config > streams

  streams: list of supported_stream_config struct, camera media streams
- **std::vector**< **std::string** > video_es

  list of string, camera video ES
- **std::vector**< **std::string** > audio_es

  list of string, camera audio ES

10.43.1 Detailed Description

Supported streams config, list of supported_stream_config.

Definition at line 1286 of file config.h.

10.43.2 Field Documentation

10.43.2.1 audio_es

std::vector< **std::string**> vxg::cloud::agent::supported_streams_config::audio_es

list of string, camera audio ES

Definition at line 1292 of file config.h.

10.43.2.2 streams

std::vector<supported_stream_config> vxg::cloud::agent::supported_streams_config::streams

streams: list of supported_stream_config struct, camera media streams

Definition at line 1288 of file config.h.

10.43.2.3 video_es

std::vector< **std::string**> vxg::cloud::agent::supported_streams_config::video_es

list of string, camera video ES

Definition at line 1290 of file config.h.

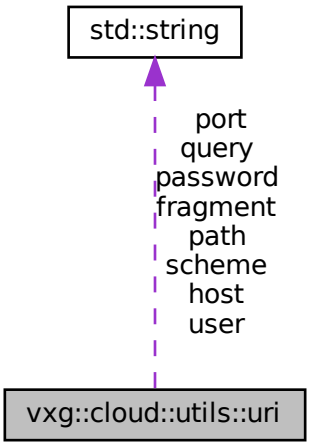
The documentation for this struct was generated from the following file:

- config.h

10.44 vxg::cloud::utils::uri Struct Reference

`#include <utils/utils.h>`

Collaboration diagram for vxg::cloud::utils::uri:



Static Public Member Functions

- static bool parse (const **std::string** &in_uri, uri &result)

Data Fields

- **std::string** scheme
- **std::string** user
- **std::string** password
- **std::string** host
- **std::string** port
- **std::string** path
- **std::string** query
- **std::string** fragment

10.44.1 Detailed Description

Definition at line 66 of file utils.h.

10.44.2 Member Function Documentation

10.44.2.1 parse()

```
static bool vxg::cloud::utils::uri::parse (
            const  std::string & in_uri,
            uri & result )  [inline], [static]
```

Definition at line 76 of file utils.h.

10.44.3 Field Documentation

10.44.3.1 fragment

```
std::string vxg::cloud::utils::uri::fragment
```

Definition at line 74 of file utils.h.

10.44.3.2 host

```
std::string vxg::cloud::utils::uri::host
```

Definition at line 70 of file utils.h.

10.44.3.3 password

```
std::string vxg::cloud::utils::uri::password
```

Definition at line 69 of file utils.h.

10.44.3.4 path

```
std::string vxg::cloud::utils::uri::path
```

Definition at line 72 of file utils.h.

10.44.3.5 port

```
std::string vxg::cloud::utils::uri::port
```

Definition at line 71 of file utils.h.

10.44.3.6 query

```
std::string vxg::cloud::utils::uri::query
```

Definition at line 73 of file utils.h.

10.44.3.7 scheme

```
std::string vxg::cloud::utils::uri::scheme
```

Definition at line 67 of file utils.h.

10.44.3.8 user

```
std::string vxg::cloud::utils::uri::user
```

Definition at line 68 of file utils.h.

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

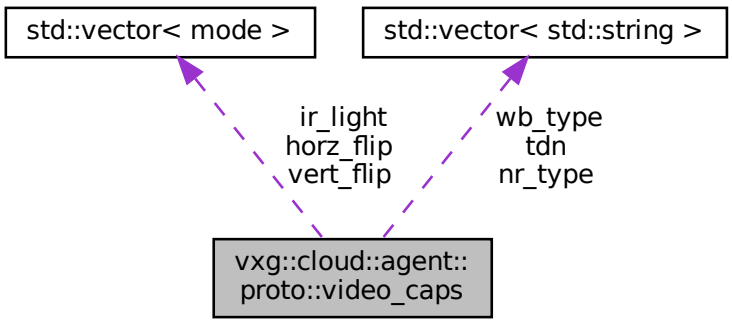
- utils.h

10.45 vxg::cloud::agent::proto::video_caps Struct Reference

Video image capabilities.

`#include <agent-proto/objects/caps.h>`

Collaboration diagram for vxg::cloud::agent::proto::video_caps:



Data Fields

- **std::vector**< mode > vert_flip

  vert_flip: list of string, supported vertical flip modes, possible values [“off”, “on”, “auto”]
- **std::vector**< mode > horz_flip

  horz_flip: list of string, supported horizontal flip modes, possible values [“off”, “on”, “auto”]
- **std::vector**< **std::string** > tdn

  tdn: list of string, supported TDM modes, possible values [“day”, “night”, “auto”]
- **std::vector**< mode > ir_light

  ir_light: list of string, supported IR light modes, possible values [“off”, “on”, “auto”]
- bool brightness

  brightness: bool, True when camera supports brightness control
- bool contrast

  contrast: bool, True when camera supports contrast control
- bool saturation

  saturation: bool, True when camera supports saturation control
- bool sharpness

  sharpness: bool, True when camera supports sharpness control
- **std::vector**< **std::string** > nr_type

  nr_type: list of string, supported noise reduce types.
- bool nr_level

  nr_level: bool, True when noise reduce filter assumes control of NR level
- **std::vector**< **std::string** > wb_type

  wb_type: list of string, supported white balance types.
- bool pwr_frequency

  pwr_frequency: bool, True camera supports compensation of images flickering due to flashing of lamps in indoor environment

10.45.1 Detailed Description

Video image capabilities.

Definition at line 366 of file caps.h.

10.45.2 Field Documentation

10.45.2.1 brightness

```
bool vxg::cloud::agent::proto::video_caps::brightness
```

brightness: bool, True when camera supports brightness control

Definition at line 384 of file caps.h.

10.45.2.2 contrast

```
bool vxg::cloud::agent::proto::video_caps::contrast
```

contrast: bool, True when camera supports contrast control

Definition at line 387 of file caps.h.

10.45.2.3 horz_flip

```
std::vector<mode> vxg::cloud::agent::proto::video_caps::horz_flip
```

horz_flip: list of string, supported horizontal flip modes, possible values [“off”, “on”, “auto”]

Definition at line 373 of file caps.h.

10.45.2.4 ir_light

```
std::vector<mode> vxg::cloud::agent::proto::video_caps::ir_light
```

ir_light: list of string, supported IR light modes, possible values [“off”, “on”, “auto”]

Definition at line 381 of file caps.h.

10.45.2.5 nr_level

```
bool vxg::cloud::agent::proto::video_caps::nr_level
```

nr_level: bool, True when noise reduce filter assumes control of NR level

Definition at line 402 of file caps.h.

10.45.2.6 nr_type

```
std::vector< std::string> vxg::cloud::agent::proto::video_caps::nr_type
```

nr_type: list of string, supported noise reduce types.

Empty list when camera doesn't support it. Example: ["off", "normal", "expert"]

Definition at line 398 of file caps.h.

10.45.2.7 pwr_frequency

```
bool vxg::cloud::agent::proto::video_caps::pwr_frequency
```

pwr_frequency: bool, True camera supports compensation of images flickering due to flashing of lamps in indoor environment

Definition at line 411 of file caps.h.

10.45.2.8 saturation

```
bool vxg::cloud::agent::proto::video_caps::saturation
```

saturation: bool, True when camera supports saturation control

Definition at line 390 of file caps.h.

10.45.2.9 sharpness

```
bool vxg::cloud::agent::proto::video_caps::sharpness
```

sharpness: bool, True when camera supports sharpness control

Definition at line 393 of file caps.h.

10.45.2.10 tdn

```
std::vector< std::string> vxg::cloud::agent::proto::video_caps::tdn
```

tdn: list of string, supported TDM modes, possible values [“day”, “night”, “auto”]

Definition at line 377 of file caps.h.

10.45.2.11 vert_flip

```
std::vector<mode> vxg::cloud::agent::proto::video_caps::vert_flip
```

vert_flip: list of string, supported vertical flip modes, possible values [“off”, “on”, “auto”]

Definition at line 369 of file caps.h.

10.45.2.12 wb_type

```
std::vector< std::string> vxg::cloud::agent::proto::video_caps::wb_type
```

wb_type: list of string, supported white balance types.

Empty list when camera doesn't support it. Example: ["auto", "3200K (Indor)", "4200K (Fluo)", "5600K (Outdoor)"]

Definition at line 407 of file caps.h.

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

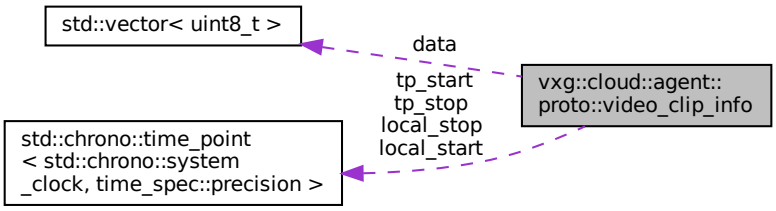
- caps.h

10.46 vxg::cloud::agent::proto::video_clip_info Struct Reference

Video recoding(mp4 file) clip description,.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::proto::video_clip_info:



Data Fields

- cloud::time tp_start
  Clip start time UTC.
- cloud::time tp_stop
  Clip stop time UTC.
- cloud::time local_start
  Clip start time local.
- cloud::time local_stop
  Clip stop time local.
- int video_width
  Video clip picture width.
- int video_height
  Video clip picture height.
- **std::vector**< uint8_t > data
  Video data buffer, we use move semantics internally so no data copying will be invoked.

10.46.1 Detailed Description

Video recoding(mp4 file) clip description,.

Definition at line 452 of file config.h.

10.46.2 Field Documentation

10.46.2.1 data

```
std::vector<uint8_t> vxg::cloud::agent::proto::video_clip_info::data
```

Video data buffer, we use move semantics internally so no data copying will be invoked.

Definition at line 478 of file config.h.

10.46.2.2 local_start

```
cloud::time vxg::cloud::agent::proto::video_clip_info::local_start
```

Clip start time local.

Definition at line 466 of file config.h.

10.46.2.3 local_stop

```
cloud::time vxg::cloud::agent::proto::video_clip_info::local_stop
```

Clip stop time local.

Definition at line 469 of file config.h.

10.46.2.4 tp_start

```
cloud::time vxg::cloud::agent::proto::video_clip_info::tp_start
```

Clip start time UTC.

Definition at line 461 of file config.h.

10.46.2.5 tp_stop

```
cloud::time vxg::cloud::agent::proto::video_clip_info::tp_stop
```

Clip stop time UTC.

Definition at line 463 of file config.h.

10.46.2.6 video_height

```
int vxg::cloud::agent::proto::video_clip_info::video_height
```

Video clip picture height.

Definition at line 474 of file config.h.

10.46.2.7 video_width

```
int vxg::cloud::agent::proto::video_clip_info::video_width
```

Video clip picture width.

Definition at line 472 of file config.h.

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

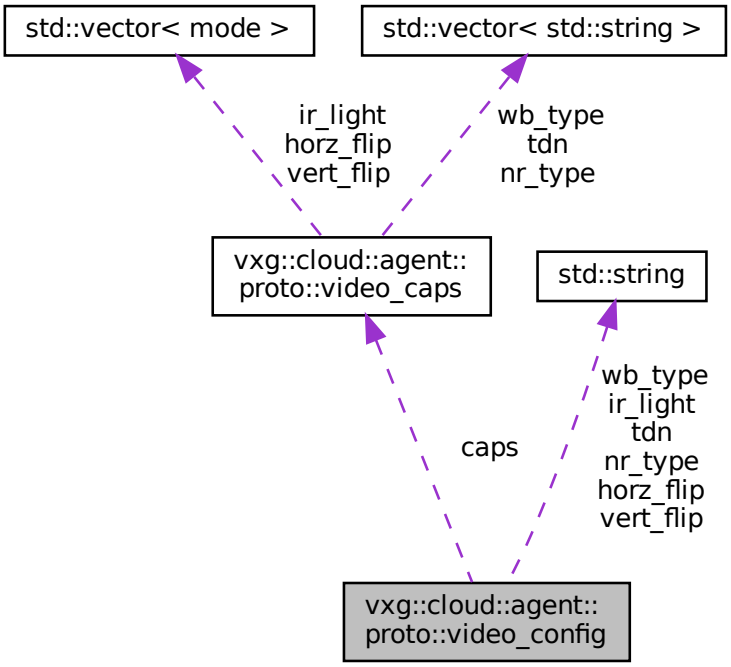
- config.h

10.47 vxg::cloud::agent::proto::video_config Struct Reference

Video image config.

```
#include <agent-proto/objects/config.h>
```

Collaboration diagram for vxg::cloud::agent::proto::video_config:



Data Fields

- **std::string** vert_flip

  vert_flip: optional string, vertical image flip mode: ["off", "on", "auto"]
- **std::string** horz_flip

  horz_flip: optional string, horizontal image flip mode: ["off", "on", "auto"]
- **std::string** tdn

  tdn: optional string, possible values ["day", "night", "auto"]
- **std::string** ir_light

  ir_light: optional string, IR light for night conditions ["off", "on", "auto"]
- int brightness

  brightness: optional int, a brightness value from range 0-100 (%)
- int contrast

  contrast: optional int, a contrast value from range 0-100 (%)
- int saturation

  saturation: optional int, a saturation value from range 0-100 (%)
- int sharpness

  sharpness: optional int, a sharpness value from range 0-100 (%)
- **std::string** nr_type

  nr_type: optional string, one of predefined noise reduce types from caps
- int nr_level

  nr_level: optional int, level of noise reduce when filter requires it 0-100 (%)
- **std::string** wb_type

wb_type: optional string, one of predefined white balance types from caps

- int pwr_frequency

  pwr_frequency: optional int, power line frequency [50, 60] (Hz)

- video_caps caps

  caps

10.47.1 Detailed Description

Video image config.

Definition at line 309 of file config.h.

10.47.2 Field Documentation

10.47.2.1 brightness

```
int vxg::cloud::agent::proto::video_config::brightness
```

brightness: optional int, a brightness value from range 0-100 (%)

Definition at line 326 of file config.h.

10.47.2.2 caps

```
video_caps vxg::cloud::agent::proto::video_config::caps
```

caps

Definition at line 352 of file config.h.

10.47.2.3 contrast

```
int vxg::cloud::agent::proto::video_config::contrast
```

contrast: optional int, a contrast value from range 0-100 (%)

Definition at line 329 of file config.h.

10.47.2.4 horz_flip

```
std::string vxg::cloud::agent::proto::video_config::horz_flip
```

horz_flip: optional string, horizontal image flip mode: [“off”, “on”, “auto”]

Definition at line 316 of file config.h.

10.47.2.5 ir_light

```
std::string vxg::cloud::agent::proto::video_config::ir_light
```

ir_light: optional string, IR light for night conditions [“off”, “on”, “auto”]

Definition at line 323 of file config.h.

10.47.2.6 nr_level

```
int vxg::cloud::agent::proto::video_config::nr_level
```

nr_level: optional int, level of noise reduce when filter requires it 0-100 (%)

Definition at line 342 of file config.h.

10.47.2.7 nr_type

```
std::string vxg::cloud::agent::proto::video_config::nr_type
```

nr_type: optional string, one of predefined noise reduce types from caps

Definition at line 338 of file config.h.

10.47.2.8 pwr_frequency

```
int vxg::cloud::agent::proto::video_config::pwr_frequency
```

pwr_frequency: optional int, power line frequency [50, 60] (Hz)

Definition at line 349 of file config.h.

10.47.2.9 saturation

```
int vxg::cloud::agent::proto::video_config::saturation
```

saturation: optional int, a saturation value from range 0-100 (%)

Definition at line 332 of file config.h.

10.47.2.10 sharpness

```
int vxg::cloud::agent::proto::video_config::sharpness
```

sharpness: optional int, a sharpness value from range 0-100 (%)

Definition at line 335 of file config.h.

10.47.2.11 tdn

```
std::string vxg::cloud::agent::proto::video_config::tdn
```

tdn: optional string, possible values [“day”, “night”, “auto”]

Definition at line 319 of file config.h.

10.47.2.12 vert_flip

```
std::string vxg::cloud::agent::proto::video_config::vert_flip
```

vert_flip: optional string, vertical image flip mode: [“off”, “on”, “auto”]

Definition at line 312 of file config.h.

10.47.2.13 wb_type

```
std::string vxg::cloud::agent::proto::video_config::wb_type
```

wb_type: optional string, one of predefined white balance types from caps

Definition at line 346 of file config.h.

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

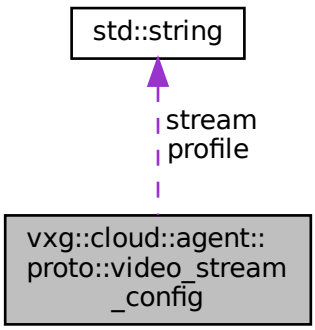
- config.h

10.48 vxg::cloud::agent::proto::video_stream_config Struct Reference

Video stream config.

`#include <agent-proto/objects/config.h>`

Collaboration diagram for vxg::cloud::agent::proto::video_stream_config:



Data Fields

- **std::string** stream

  Mandatory: video ES to use.
- video_format format

  Mandatory: video encoding format.
- **std::string** profile

  Optional: profile that specifies format, when format assumes it.
- int horz

  Mandatory: int (horz) - video resolution width x height.
- int vert

  Mandatory: int (vert) - video resolution width x height.
- double fps

  Mandatory: framerate.
- bool vbr

  Mandatory: prefer VBR; if false or not set CBR should be used.
- int gop

  Mandatory: gop size (I-Frame interval);.
- int brt

  Optional: bitrate, kbps.
- int vbr_brt

  Optional: bitrate for VBR, kbps.
- int quality

  Optional: int [-4..4], quality profile hint for encoder, where 0 means normal.
- int smoothing

  Optional: a smoothing value from range 0-100 (%)

10.48.1 Detailed Description

Video stream config.

Definition at line 86 of file config.h.

10.48.2 Field Documentation

10.48.2.1 brt

```
int vxg::cloud::agent::proto::video_stream_config::brt
```

Optional: bitrate, kbps.

Definition at line 120 of file config.h.

10.48.2.2 format

```
video_format vxg::cloud::agent::proto::video_stream_config::format
```

Mandatory: video encoding format.

Definition at line 93 of file config.h.

10.48.2.3 fps

```
double vxg::cloud::agent::proto::video_stream_config::fps
```

Mandatory: framerate.

Definition at line 108 of file config.h.

10.48.2.4 gop

```
int vxg::cloud::agent::proto::video_stream_config::gop
```

Mandatory: gop size (I-Frame interval);.

Definition at line 116 of file config.h.

10.48.2.5 horz

```
int vxg::cloud::agent::proto::video_stream_config::horz
```

Mandatory: int (horz) - video resolution width x height.

Definition at line 101 of file config.h.

10.48.2.6 profile

```
std::string vxg::cloud::agent::proto::video_stream_config::profile
```

Optional: profile that specifies format, when format assumes it.

Definition at line 97 of file config.h.

10.48.2.7 quality

```
int vxg::cloud::agent::proto::video_stream_config::quality
```

Optional: int [-4..4], quality profile hint for encoder, where 0 means normal.

Definition at line 128 of file config.h.

10.48.2.8 smoothing

```
int vxg::cloud::agent::proto::video_stream_config::smoothing
```

Optional: a smoothing value from range 0-100 (%)

Definition at line 132 of file config.h.

10.48.2.9 stream

```
std::string vxg::cloud::agent::proto::video_stream_config::stream
```

Mandatory: video ES to use.

Definition at line 89 of file config.h.

10.48.2.10 vbr

```
bool vxg::cloud::agent::proto::video_stream_config::vbr
```

Mandatory: prefer VBR; if false or not set CBR should be used.

Definition at line 112 of file config.h.

10.48.2.11 vbr_brt

```
int vxg::cloud::agent::proto::video_stream_config::vbr_brt
```

Optional: bitrate for VBR, kbps.

Definition at line 124 of file config.h.

10.48.2.12 vert

```
int vxg::cloud::agent::proto::video_stream_config::vert
```

Mandatory: int (vert) - video resolution width x height.

Definition at line 104 of file config.h.

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

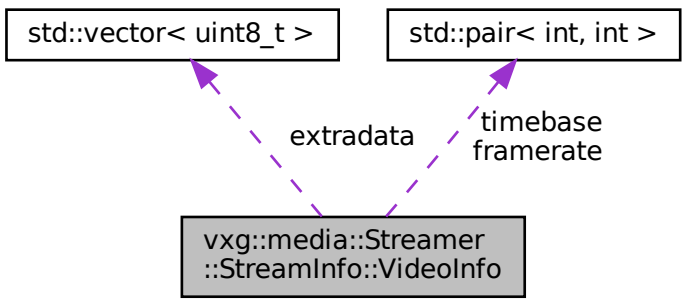
- config.h

10.49 vxg::media::Streamer::StreamInfo::VideoInfo Struct Reference

Video stream info.

```
#include <streamer/base_streamer.h>
```

Collaboration diagram for vxg::media::Streamer::StreamInfo::VideoInfo:



Data Fields

- VideoCodec codec

  Video codec type.
- int width

  Video width if needed.
- int height

  Video height if needed.
- **std::pair**< int, int > framerate

  Video framerate if needed.
- int bitrate

  Video bitrate if needed.
- **std::pair**< int, int > timebase

  Timescale of the timestamps, source fills it with timescale of timestamps source receives, MediaFrame::pts should use this timescale.
- **std::vector**< uint8_t > extradata

  Can be AVC1 extradata or SPS/PPS, source fills it and sink should know and understand this format.

10.49.1 Detailed Description

Video stream info.

This structure as well as ISink::negotiate method aimed to inform sink about streams source provides, if sink don't care the values of this structure may be ignored.

Definition at line 310 of file base_streamer.h.

10.49.2 Field Documentation

10.49.2.1 bitrate

```
int vxg::media::Streamer::StreamInfo::VideoInfo::bitrate
```

Video bitrate if needed.

Definition at line 320 of file base_streamer.h.

10.49.2.2 codec

```
VideoCodec vxg::media::Streamer::StreamInfo::VideoInfo::codec
```

Video codec type.

Definition at line 312 of file base_streamer.h.

10.49.2.3 extradata

```
std::vector<uint8_t> vxg::media::Streamer::StreamInfo::VideoInfo::extradata
```

Can be AVC1 extradata or SPS/PPS, source fills it and sink should know and understand this format.

Definition at line 327 of file base_streamer.h.

10.49.2.4 framerate

```
std::pair<int, int> vxg::media::Streamer::StreamInfo::VideoInfo::framerate
```

Video framerate if needed.

Definition at line 318 of file base_streamer.h.

10.49.2.5 height

```
int vxg::media::Streamer::StreamInfo::VideoInfo::height
```

Video height if needed.

Definition at line 316 of file base_streamer.h.

10.49.2.6 timebase

```
std::pair<int, int> vxg::media::Streamer::StreamInfo::VideoInfo::timebase
```

Timescale of the timestamps, source fills it with timescale of timestamps source receives, MediaFrame::pts should use this timescale.

Definition at line 324 of file base_streamer.h.

10.49.2.7 width

```
int vxg::media::Streamer::StreamInfo::VideoInfo::width
```

Video width if needed.

Definition at line 314 of file base_streamer.h.

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

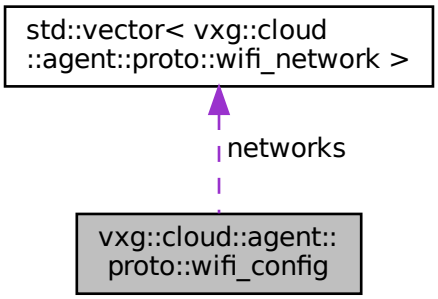
- base_streamer.h

10.50 vxg::cloud::agent::proto::wifi_config Struct Reference

WiFi config.

`#include <agent-proto/objects/config.h>`

Collaboration diagram for vxg::cloud::agent::proto::wifi_config:



Data Fields

- **std::vector**< wifi_network > networks
  List of wifi_network objects.

10.50.1 Detailed Description

WiFi config.

Definition at line 584 of file config.h.

10.50.2 Field Documentation

10.50.2.1 networks

std::vector<wifi_network> vxg::cloud::agent::proto::wifi_config::networks

List of wifi_network objects.

Definition at line 586 of file config.h.

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

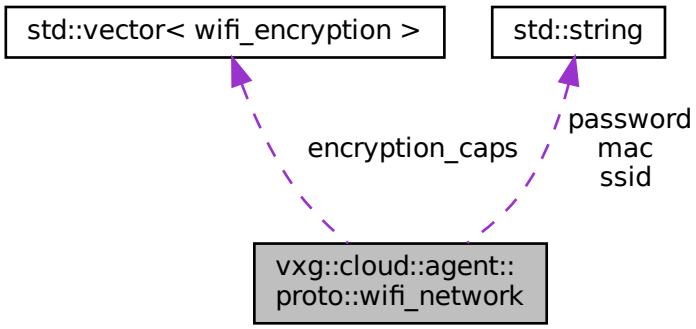
- config.h

10.51 vxg::cloud::agent::proto::wifi_network Struct Reference

WiFi network object.

`#include <agent-proto/objects/config.h>`

Collaboration diagram for vxg::cloud::agent::proto::wifi_network:



Data Fields

- **std::string** ssid

  ssid: string, network SSID
- int signal

  signal: int, signal strength, dB
- **std::string** mac

  mac: string, AP MAC address
- **std::vector**< wifi_encryption > encryption_caps

  encryption_caps: list of string, supported encryption types,
- wifi_encryption encryption

  encryption: string, current encryption type, see encryption_caps for possible values
- **std::string** password

  password: string, network password

10.51.1 Detailed Description

WiFi network object.

Definition at line 555 of file config.h.

10.51.2 Field Documentation

10.51.2.1 encryption

`wifi_encryption vxg::cloud::agent::proto::wifi_network::encryption`

encryption: string, current encryption type, see encryption_caps for possible values

Definition at line 566 of file config.h.

10.51.2.2 encryption_caps

`std::vector<wifi_encryption> vxg::cloud::agent::proto::wifi_network::encryption_caps`

encryption_caps: list of string, supported encryption types,

Definition at line 563 of file config.h.

10.51.2.3 mac

`std::string vxg::cloud::agent::proto::wifi_network::mac`

mac: string, AP MAC address

Definition at line 561 of file config.h.

10.51.2.4 password

`std::string vxg::cloud::agent::proto::wifi_network::password`

password: string, network password

Definition at line 568 of file config.h.

10.51.2.5 signal

`int vxg::cloud::agent::proto::wifi_network::signal`

signal: int, signal strength, dB

Definition at line 559 of file config.h.

10.51.2.6 ssid

`std::string vxg::cloud::agent::proto::wifi_network::ssid`

ssid: string, network SSID

Definition at line 557 of file config.h.

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

- config.h

Chapter 11

File Documentation

11.1 app-dev.md File Reference

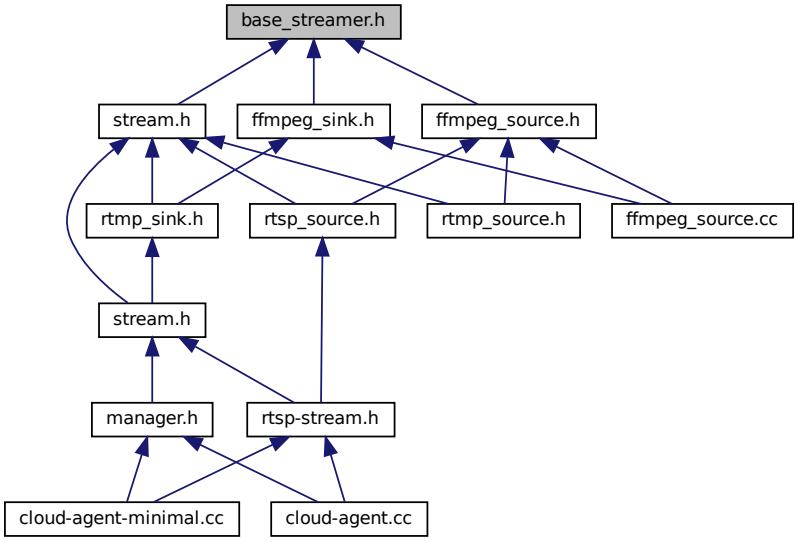
11.2 arm-example.txt File Reference

11.3 base_streamer.h File Reference

```
#include <cstdlib>
#include <future>
#include <map>
#include <queue>
#include <string>
#include <pthread.h>
#include <streamer/stats.h>
#include <utils/logging.h>
#include <utils/utils.h>
```

Include dependency graph for base_streamer.h:

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



Data Structures

- struct vxg::media::Streamer::StreamInfo

  Stream info descriotion.
- struct vxg::media::Streamer::StreamInfo::VideoInfo

  Video stream info.
- struct vxg::media::Streamer::StreamInfo::AudioInfo

  Audio stream info.
- struct vxg::media::Streamer::MediaFrame

  Media frame container.
- class vxg::media::Streamer::ISink
- class vxg::media::Streamer::ISource

  ISource interface class.

Namespaces

- vxg
- vxg::media
- vxg::media::Streamer

Macros

- #define __BASE_STREAMER_H

Enumerations

- enum vxg::media::Streamer::DropDirection { vxg::media::Streamer::DROP_FRONT, vxg::media::Streamer::DROP_BACK }
- enum vxg::media::Streamer::StreamError { vxg::media::Streamer::E_NONE, vxg::media::Streamer::E_FATAL, vxg::media::Streamer::E_EOS }

  Stream error.
- enum vxg::media::Streamer::MediaType { vxg::media::Streamer::UKNOWN, vxg::media::Streamer::VIDEO, vxg::media::Streamer::VIDEO_AVC_SPS, vxg::media::Streamer::VIDEO_AVC_PPS, vxg::media::Streamer::VIDEO_SEQ_HDR, vxg::media::Streamer::AUDIO, vxg::media::Streamer::AUDIO_SEQ_HDR, vxg::media::Streamer::FLV, vxg::media::Streamer::DATA, vxg::media::Streamer::MAX }

  Media frame type.

Variables

- constexpr int vxg::media::Streamer::SINK_THREAD_PRIO
- constexpr int vxg::media::Streamer::SRC_THREAD_PRIO

11.3.1 Macro Definition Documentation

11.3.1.1 __BASE_STREAMER_H

```
#define __BASE_STREAMER_H
```

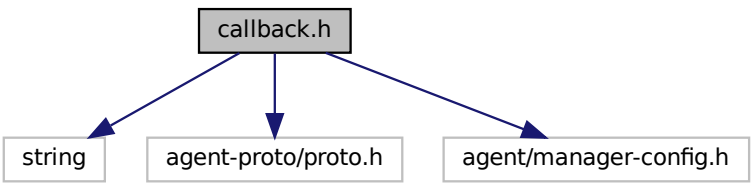
Definition at line 3 of file base_streamer.h.

11.4 build-system.md File Reference

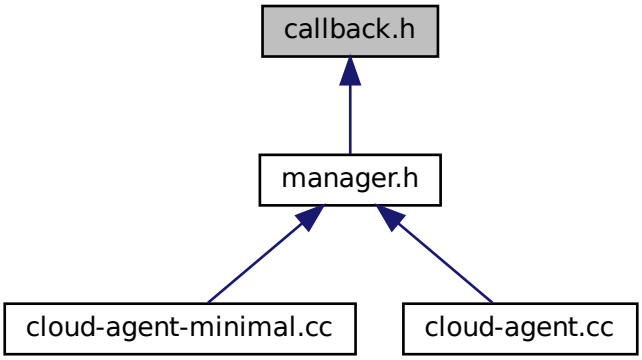
11.5 callback.h File Reference

```
#include <string>
#include <agent-proto/proto.h>
#include <agent/manager-config.h>
```

Include dependency graph for callback.h:



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



Data Structures

- class vxg::cloud::agent::callback

  VXG Cloud manager common callbacks class.

Namespaces

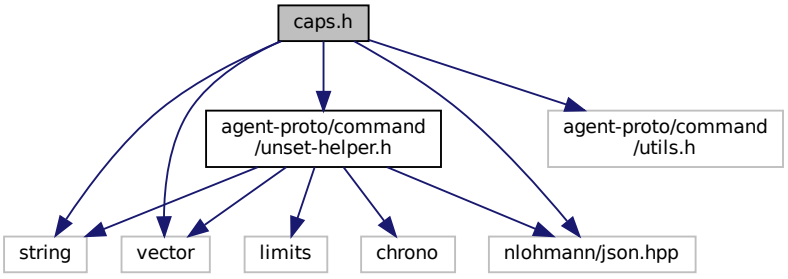
- vxg
- vxg::cloud
- vxg::cloud::agent

  VXG Cloud Agent namespace.

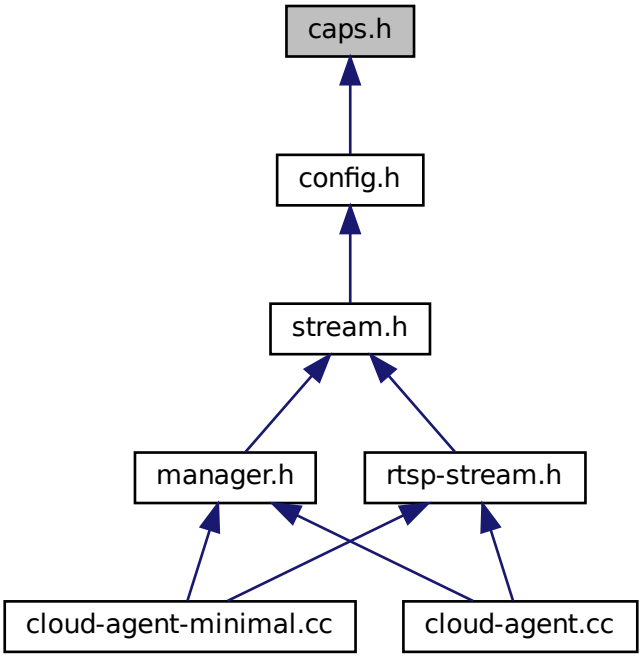
11.6 caps.h File Reference

```
#include <string>
#include <vector>
#include <nlohmann/json.hpp>
#include <agent-proto/command/unset-helper.h>
#include <agent-proto/command/utils.h>
```

Include dependency graph for caps.h:



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



Data Structures

- struct vxg::cloud::agent::proto::stream_caps
  Media stream capabilites.
- struct vxg::cloud::agent::proto::stream_caps::caps_video_object
  Video streams capabilities.
- struct vxg::cloud::agent::proto::stream_caps::caps_audio_object
  Audio streams capabilities.
- struct vxg::cloud::agent::proto::motion_detection_caps
  Motion detection capabilities camera capabilities that limit possible motion detection configuration.
- struct vxg::cloud::agent::proto::video_caps
  Video image capabilities.
- struct vxg::cloud::agent::proto::event_caps
  Events capabilies.
- struct vxg::cloud::agent::proto::audio_caps
  Audio capabilities.
- struct vxg::cloud::agent::proto::osd_caps
  OSD capabilities.

Namespaces

- vxg
- vxg::cloud
- vxg::cloud::agent
  VXG Cloud Agent namespace.
- vxg::cloud::agent::proto

Macros

- #define ignore_exception(...)

Typedefs

- using json = nlohmann::json

Enumerations

- enum vxg::cloud::agent::proto::mode { vxg::cloud::agent::proto::M_OFF, vxg::cloud::agent::proto::M_ON, vxg::cloud::agent::proto::M_AUTO, vxg::cloud::agent::proto::M_INVALID }

  Mode on/off.
- enum vxg::cloud::agent::proto::video_format { vxg::cloud::agent::proto::VF_H264, vxg::cloud::agent::proto::VF_H265, vxg::cloud::agent::proto::VF_MJPEG, vxg::cloud::agent::proto::VF_INVALID }

  Video codec format.
- enum vxg::cloud::agent::proto::audio_format { vxg::cloud::agent::proto::AF_G711A, vxg::cloud::agent::proto::AF_G711U, vxg::cloud::agent::proto::AF_RAW, vxg::cloud::agent::proto::AF_ADPCM, vxg::cloud::agent::proto::AF_MP3, vxg::cloud::agent::proto::AF_NELLY8, vxg::cloud::agent::proto::AF_NELLY16, vxg::cloud::agent::proto::AF_NELLY, vxg::cloud::agent::proto::AF_OPUS, vxg::cloud::agent::proto::AF_AAC, vxg::cloud::agent::proto::AF_SPEEX, vxg::cloud::agent::proto::AF_INVALID }

  Audio codec format.
- enum vxg::cloud::agent::proto::audio_file_format { vxg::cloud::agent::proto::AFF_AU_G711U, vxg::cloud::agent::proto::AFF_MP vxg::cloud::agent::proto::AFF_WAV_PCM, vxg::cloud::agent::proto::AFF_INVALID }

  Audio file format.
- enum vxg::cloud::agent::proto::motion_sensitivity { vxg::cloud::agent::proto::MS_REGION, vxg::cloud::agent::proto::MS_FRAM vxg::cloud::agent::proto::MS_INVALID }

  Motion sensitivity.
- enum vxg::cloud::agent::proto::motion_region_shape { vxg::cloud::agent::proto::MR_RECTANGLE, vxg::cloud::agent::proto::MR_ANY, vxg::cloud::agent::proto::MR_INVALID }

  Motion region shape.
- enum vxg::cloud::agent::proto::ptz_action { vxg::cloud::agent::proto::A_LEFT, vxg::cloud::agent::proto::A_RIGHT, vxg::cloud::agent::proto::A_TOP, vxg::cloud::agent::proto::A_BOTTOM, vxg::cloud::agent::proto::A_ZOOM_IN, vxg::cloud::agent::proto::A_ZOOM_OUT, vxg::cloud::agent::proto::A_STOP, vxg::cloud::agent::proto::A_INVALID }

  PTZ actions.
- enum vxg::cloud::agent::proto::ptz_preset_action { vxg::cloud::agent::proto::PA_CREATE, vxg::cloud::agent::proto::PA_DELETE, vxg::cloud::agent::proto::PA_GOTO, vxg::cloud::agent::proto::PA_UPDATE, vxg::cloud::agent::proto::PA_INVALID }

  PTZ preset action.
- enum vxg::cloud::agent::proto::time_format_n { vxg::cloud::agent::proto::TF_12H, vxg::cloud::agent::proto::TF_24H, vxg::cloud::agent::proto::TF_INVALID }

  3.34 get_osd_conf (SRV) 3.35 osd_conf (CM) 3.36 set_osd_conf (SRV)

11.6.1 Macro Definition Documentation

11.6.1.1 ignore_exception

```
#define ignore_exception(
              ...  )
```

Definition at line 20 of file caps.h.

11.6.2 Typedef Documentation

11.6.2.1 json

```
using json = nlohmann::json
```

Definition at line 12 of file caps.h.

11.7 cloud-agent-minimal.cc File Reference

```
#include <signal.h>
#include <args.hxx>
#include <agent/manager.h>
#include <agent/rtsp-stream.h>
#include <utils/logging.h>
#include <utils/properties.h>
```

Include dependency graph for cloud-agent-minimal.cc:

Functions

- static void signal_handler (int sig)
- bool parse_args (int argc, char **argv)
- int main (int argc, char **argv)

Variables

- static bool quit

  [Includes and namespaces]
- static vxg::properties props
- **std::string** vxg_cloud_token

  [Minimal callback class implementation]
- **std::string** rtsp_url

11.7.1 Function Documentation

11.7.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

[Create and start agent object]

[Create and start agent object]

[Stop agent]

[Stop agent]

Definition at line 85 of file cloud-agent-minimal.cc.

11.7.1.2 parse_args()

```
bool parse_args (
            int argc,
            char ** argv )
```

Definition at line 46 of file cloud-agent-minimal.cc.

11.7.1.3 signal_handler()

```
static void signal_handler (
            int sig )  [static]
```

Definition at line 18 of file cloud-agent-minimal.cc.

11.7.2 Variable Documentation

11.7.2.1 props

```
vxg::properties props  [static]
```

Definition at line 16 of file cloud-agent-minimal.cc.

11.7.2.2 quit

```
bool quit  [static]
```

[Includes and namespaces]

Definition at line 15 of file cloud-agent-minimal.cc.

11.7.2.3 rtsp_url

```
std::string rtsp_url
```

Definition at line 44 of file cloud-agent-minimal.cc.

11.7.2.4 vxg_cloud_token

```
std::string vxg_cloud_token
```

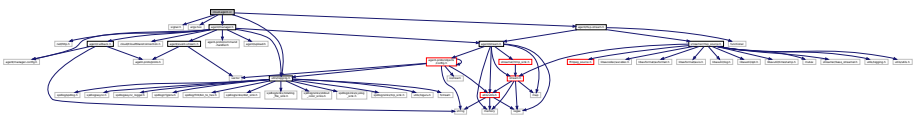
[Minimal callback class implementation]

Definition at line 43 of file cloud-agent-minimal.cc.

11.8 cloud-agent.cc File Reference

```
#include <signal.h>
#include <args.hxx>
#include <agent/manager.h>
#include <agent/rtsp-stream.h>
#include <utils/logging.h>
```

Include dependency graph for cloud-agent.cc:



Functions

- static void signal_handler (int sig)
- bool parse_args (int argc, char **argv)
- int main (int argc, char **argv)

Variables

- static bool quit

  [Includes and namespaces]
- **std::string** vxg_cloud_token

  [Event stream callback class implementation]
- **std::string** rtsp_url

11.8.1 Function Documentation

11.8.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

[Create and start agent object]

[Create and start agent object]

[Stop agent]

[Stop agent]

Definition at line 349 of file cloud-agent.cc.

11.8.1.2 parse_args()

```
bool parse_args (
            int argc,
            char ** argv )
```

Definition at line 317 of file cloud-agent.cc.

11.8.1.3 signal_handler()

```
static void signal_handler (
            int sig )  [static]
```

Definition at line 17 of file cloud-agent.cc.

11.8.2 Variable Documentation

11.8.2.1 quit

```
bool quit  [static]
```

[Includes and namespaces]

Definition at line 14 of file cloud-agent.cc.

11.8.2.2 rtsp_url

```
std::string rtsp_url
```

Definition at line 315 of file cloud-agent.cc.

11.8.2.3 vxg_cloud_token

```
std::string vxg_cloud_token
```

[Event stream callback class implementation]

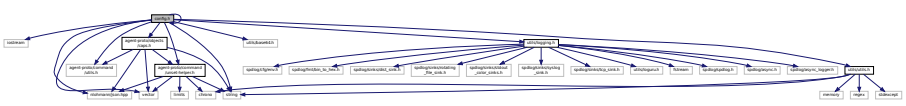
Definition at line 314 of file cloud-agent.cc.

11.9 compile.md File Reference

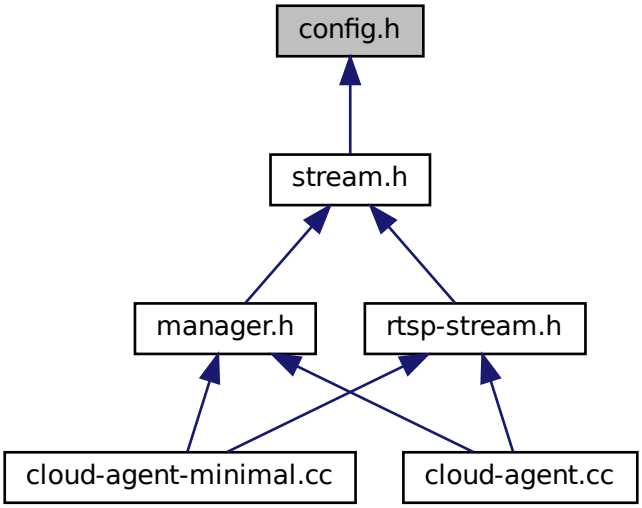
11.10 config.h File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <config.h>
#include <nlohmann/json.hpp>
#include <agent-proto/command/unset-helper.h>
#include <agent-proto/command/utils.h>
#include <agent-proto/objects/caps.h>
#include <utils/base64.h>
#include <utils/logging.h>
#include <utils/utils.h>
```

Include dependency graph for config.h:



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



Data Structures

- struct vxg::cloud::agent::proto::video_stream_config
  Video stream config.
- struct vxg::cloud::agent::proto::audio_stream_config
  Audio media stream config.
- struct vxg::cloud::agent::proto::stream_config
  Media stream config.
- struct vxg::cloud::agent::proto::motion_region
  Motion detection related structs.
- struct vxg::cloud::agent::proto::motion_detection_config
  Motion detection config.
- struct vxg::cloud::agent::proto::video_config
  Video image config.
- struct vxg::cloud::agent::proto::video_clip_info
  Video recoding(mp4 file) clip description,.
- struct vxg::cloud::agent::proto::wifi_network
  WiFi network object.
- struct vxg::cloud::agent::proto::wifi_config
  WiFi config.
- struct vxg::cloud::agent::event_config
  Event config.
- struct vxg::cloud::agent::events_config
  Events config, list of event_config objects.
- struct vxg::cloud::agent::audio_config
  Audio config.
- struct vxg::cloud::agent::ptz_preset

PTZ preset.

- struct vxg::cloud::agent::ptz_config

  PTZ config.

- struct vxg::cloud::agent::ptz_command

  PTZ command.

- struct vxg::cloud::agent::osd_config

  OSD config.

- struct vxg::cloud::agent::access_token

  VXG Cloud access token.

- struct vxg::cloud::agent::access_token::proxy_config

  Socks proxy settings.

- struct vxg::cloud::agent::supported_stream_config

  Supported stream config.

- struct vxg::cloud::agent::supported_streams_config

  Supported streams config, list of supported_stream_config.

Namespaces

- vxg
- vxg::cloud
- vxg::cloud::time_spec

  time point

- nlohmann
- vxg::cloud::agent

  VXG Cloud Agent namespace.

- vxg::cloud::agent::proto

Macros

- #define __CONFIG_H_

Typedefs

- using vxg::cloud::time_spec::precision = **std::chrono::nanoseconds**
- template<typename T >
  using vxg::cloud::time_spec::duration = typename **std::conditional**< **std::is_same**< T, precision >::value, precision, **std::chrono::duration**< T > >::type
- using vxg::cloud::time = **std::chrono::time_point**< **std::chrono::system_clock**, time_spec::precision >
- using vxg::cloud::duration = time_spec::duration< time_spec::precision >
- typedef wifi_config vxg::cloud::agent::proto::wifi_list

  wifi_config

Enumerations

- enum vxg::cloud::agent::proto::event_status { vxg::cloud::agent::proto::ES_OK, vxg::cloud::agent::proto::ES_ERROR, vxg::cloud::agent::proto::ES_INVALID }

  Event status.
- enum vxg::cloud::agent::proto::event_type { vxg::cloud::agent::proto::ET_MOTION, vxg::cloud::agent::proto::ET_SOUND, vxg::cloud::agent::proto::ET_NET, vxg::cloud::agent::proto::ET_RECORD, vxg::cloud::agent::proto::ET_MEMORYCARD, vxg::cloud::agent::proto::ET_WIFI, vxg::cloud::agent::proto::ET_CUSTOM, vxg::cloud::agent::proto::ET_INVALID }

  Types of events.
- enum vxg::cloud::agent::proto::memorycard_status { vxg::cloud::agent::proto::MCS_NONE, vxg::cloud::agent::proto::MCS_NORMAL, vxg::cloud::agent::proto::MCS_NEED_FORM vxg::cloud::agent::proto::MCS_FORMATTING, vxg::cloud::agent::proto::MCS_INITIALIZATION, vxg::cloud::agent::proto::MCS_INVALID }

  Memory card status.
- enum vxg::cloud::agent::proto::wifi_encryption { vxg::cloud::agent::proto::WFE_OPEN, vxg::cloud::agent::proto::WFE_WEP, vxg::cloud::agent::proto::WFE_WPA, vxg::cloud::agent::proto::WFE_WPA2, vxg::cloud::agent::proto::WFE_WPA_ENTERPRISE, vxg::cloud::agent::proto::WFE_WPA2_ENTERPRISE, vxg::cloud::agent::proto::WFE_INVALID }

  WiFi encryption type.
- enum vxg::cloud::agent::proto::wifi_network_state { vxg::cloud::agent::proto::WNS_UNKNOWN, vxg::cloud::agent::proto::WNS_INITIALIZE_0, vxg::cloud::agent::proto::WNS_INIT vxg::cloud::agent::proto::WNS_TRY_CONNECT, vxg::cloud::agent::proto::WNS_RECEIVING_IP, vxg::cloud::agent::proto::WNS_CONNECTED, vxg::cloud::agent::proto::WNS_ }

  WiFi connection state.

Functions

- **std::string** vxg::cloud::agent::proto::name ()

11.10.1 Detailed Description

VXG Cloud CM protocol objects

11.10.2 Macro Definition Documentation

11.10.2.1 __CONFIG_H_

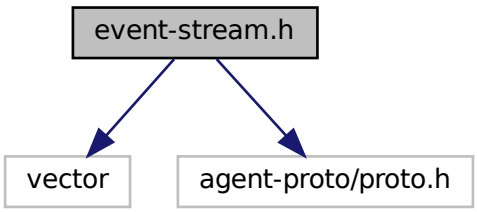
```
#define __CONFIG_H_
```

Definition at line 4 of file config.h.

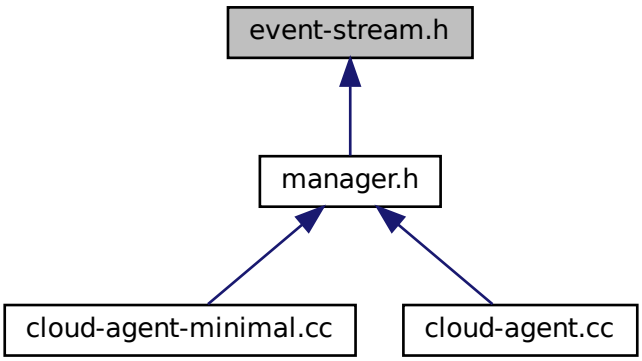
11.11 event-stream.h File Reference

```
#include <vector>
#include <agent-proto/proto.h>
```

Include dependency graph for event-stream.h:



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



Data Structures

- class vxg::cloud::agent::event_stream

  Event stream, abstract class for event generation.

Namespaces

- vxg
- vxg::cloud
- vxg::cloud::agent

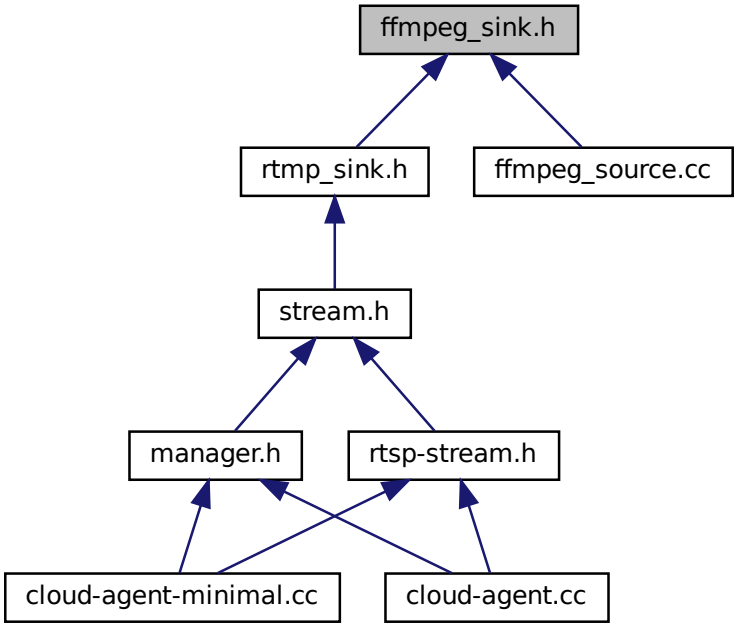
  VXG Cloud Agent namespace.

11.12 ffmpeg_sink.h File Reference

```
#include "base_streamer.h"
#include "ffmpeg_common.h"
```

Include dependency graph for ffmpeg_sink.h:

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



Data Structures

- class vxg::media::ffmpeg::Sink

  Base ffmpeg sink class.

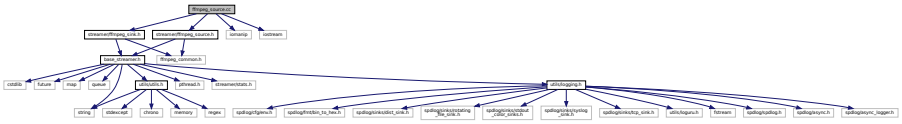
Namespaces

- vxg
- vxg::media
- vxg::media::ffmpeg

11.13 ffmpeg_source.cc File Reference

```
#include <streamer/ffmpeg_sink.h>
#include <streamer/ffmpeg_source.h>
#include <iomanip>
#include <iostream>
```

Include dependency graph for ffmpeg_source.cc:



Namespaces

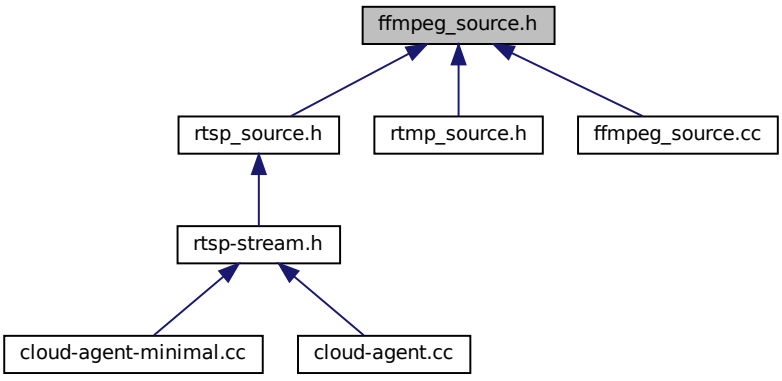
- vxg
- vxg::media

11.14 ffmpeg_source.h File Reference

```
#include "base_streamer.h"
#include "ffmpeg_common.h"
```

Include dependency graph for ffmpeg_source.h:

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



Data Structures

- class vxg::media::ffmpeg::Source

  Base ffmpeg source class.

Namespaces

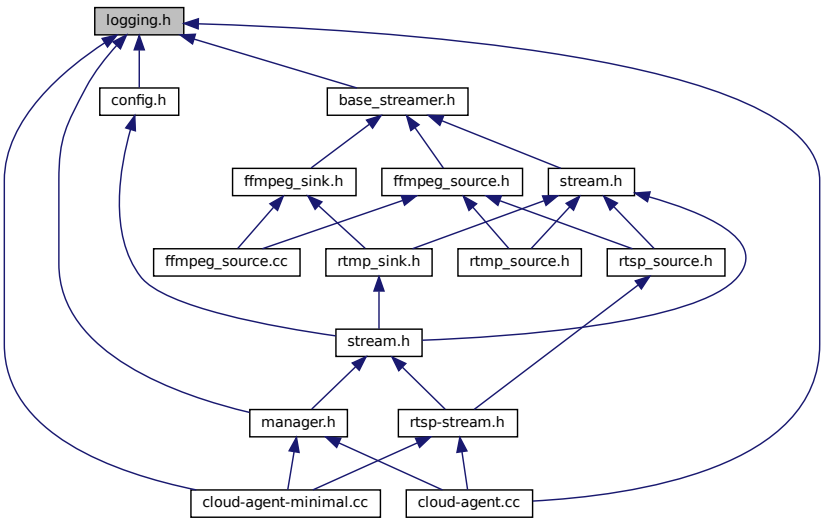
- vxg
- vxg::media
- vxg::media::ffmpeg

11.15 logging.h File Reference

```
#include <spdlog/spdlog.h>
#include <spdlog/async.h>
#include <spdlog/async_logger.h>
#include <spdlog/cfg/env.h>
#include <spdlog/fmt/bin_to_hex.h>
#include <spdlog/sinks/dist_sink.h>
#include <spdlog/sinks/rotating_file_sink.h>
#include <spdlog/sinks/stdout_color_sinks.h>
#include <spdlog/sinks/syslog_sink.h>
#include <spdlog/sinks/tcp_sink.h>
#include <utils/loguru.h>
#include <fstream>
```

Include dependency graph for logging.h:

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



Data Structures

- class vxg::logger

  Logger class, current implementation based on spdlog.
- struct vxg::logger::options

Namespaces

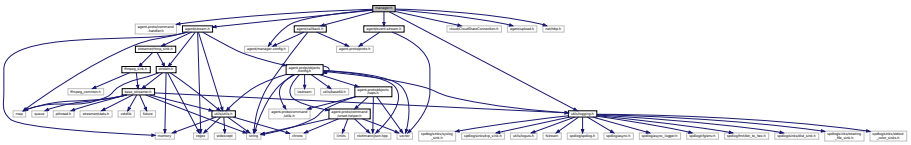
- vxg

11.16 mainpage.md File Reference

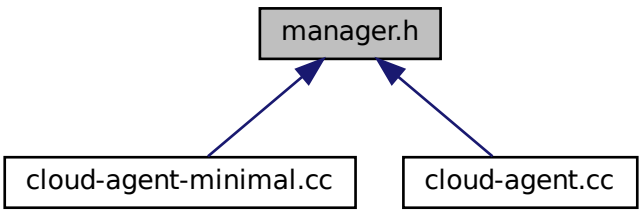
11.17 manager.h File Reference

```
#include <agent-proto/command-handler.h>
#include <agent/callback.h>
#include <agent/event-stream.h>
#include <agent/manager-config.h>
#include <cloud/CloudShareConnection.h>
#include <agent/stream.h>
#include <agent/upload.h>
#include <net/http.h>
#include <utils/logging.h>
```

Include dependency graph for manager.h:



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



Data Structures

- class vxg::cloud::agent::manager

  VXG Cloud agent manager class.

- struct vxg::cloud::agent::manager::event_state::event_state_caps

Namespaces

- vxg
- vxg::cloud
- vxg::cloud::agent

  VXG Cloud Agent namespace.

Functions

- **std::string** vxg::cloud::agent::version ()

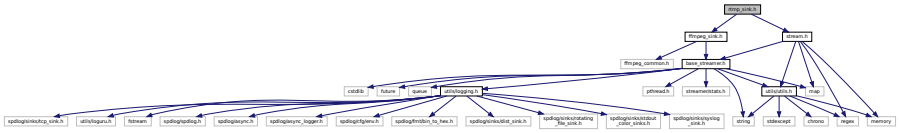
  VXG Cloud Agent library version.

11.18 meson.build File Reference

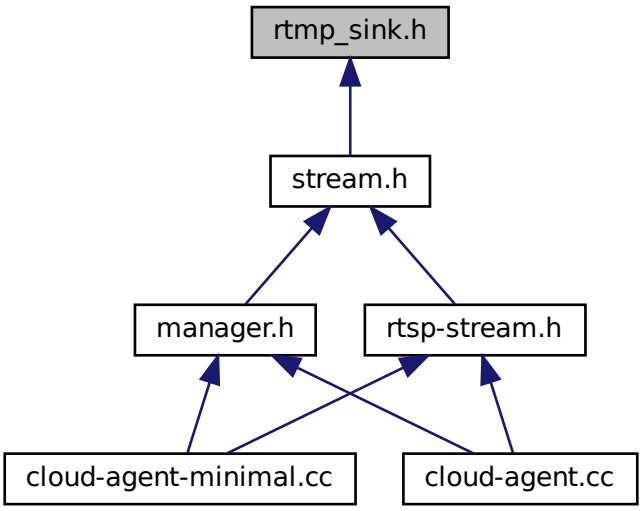
11.19 rtmp_sink.h File Reference

```
#include "ffmpeg_sink.h"
#include "stream.h"
```

Include dependency graph for rtmp_sink.h:



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



Data Structures

- class vxg::media::rtmp_sink

  RTMP sink class.

Namespaces

- vxg
- vxg::media

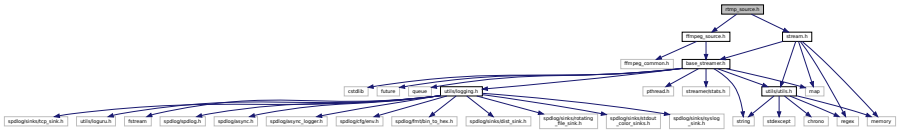
11.19.1 Detailed Description

RTMP sink

11.20 rtmp_source.h File Reference

```
#include "ffmpeg_source.h"
#include "stream.h"
```

Include dependency graph for rtmp_source.h:



Data Structures

- class vxg::media::rtmp_source

  RTMP source class.

Namespaces

- vxg
- vxg::media

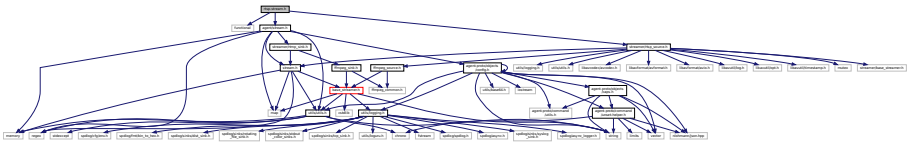
11.20.1 Detailed Description

RTMP source

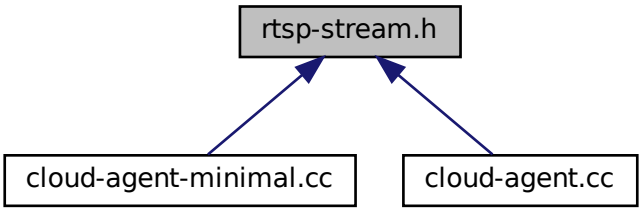
11.21 rtsp-stream.h File Reference

```
#include <functional>
#include <agent/stream.h>
#include <streamer/rtsp_source.h>
```

Include dependency graph for rtsp-stream.h:



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



Data Structures

- class vxg::cloud::agent::media::rtsp_stream

  Implementation of the media::stream with RTSP source and NIY stubs.

Namespaces

- vxg
- vxg::cloud
- vxg::cloud::agent

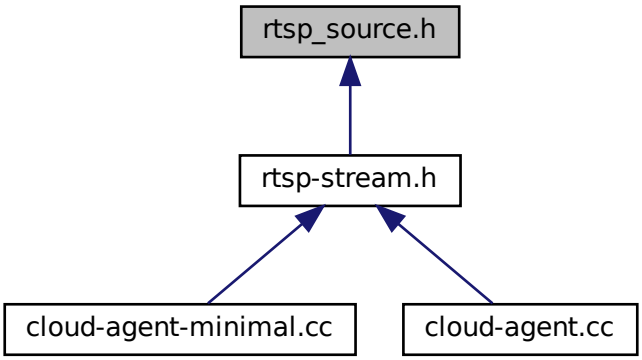
  VXG Cloud Agent namespace.
- vxg::cloud::agent::media

11.22 rtsp_source.h File Reference

```
#include "ffmpeg_source.h"
#include "stream.h"
```

Include dependency graph for rtsp_source.h:

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



Data Structures

- class vxg::media::rtsp_source

  RTSP source class.

Namespaces

- vxg
- vxg::media

11.22.1 Detailed Description

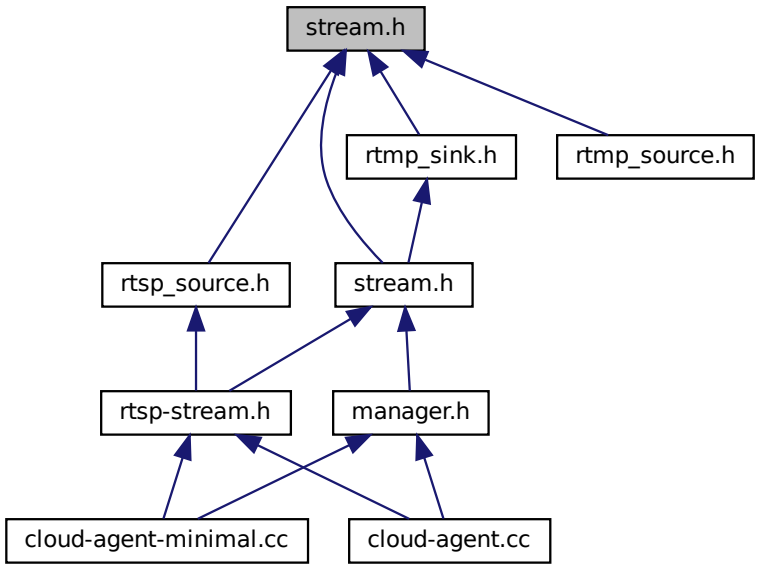
RTSP source

11.23 stream.h File Reference

```
#include <map>
#include <memory>
#include <regex>
#include <streamer/base_streamer.h>
#include <utils/utils.h>
```

Include dependency graph for streamer/stream.h:

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



Data Structures

- class vxg::media::stream

  base media stream abstract class

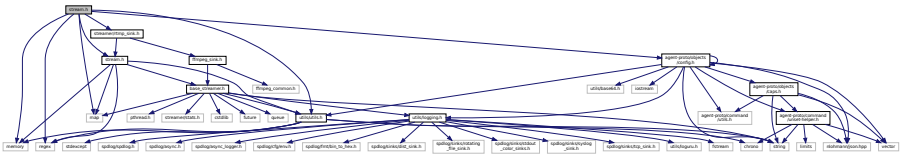
Namespaces

- vxg
- vxg::media

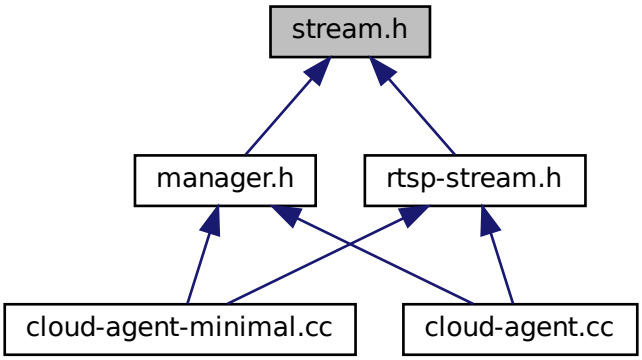
11.24 stream.h File Reference

```
#include <map>
#include <memory>
#include <regex>
#include <agent-proto/objects/config.h>
#include <streamer/rtmp_sink.h>
#include <streamer/stream.h>
#include <utils/utils.h>
```

Include dependency graph for agent/stream.h:



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



Data Structures

- class vxg::cloud::agent::media::stream

  Cloud agent media stream abstract class.

Namespaces

- vxg
- vxg::cloud
- vxg::cloud::agent

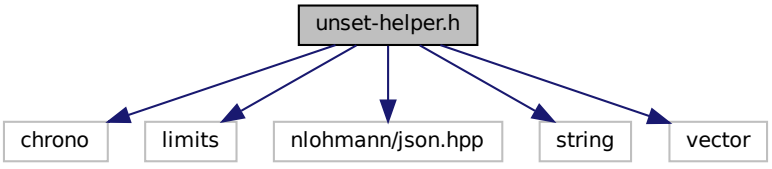
  VXG Cloud Agent namespace.

- vxg::cloud::agent::media

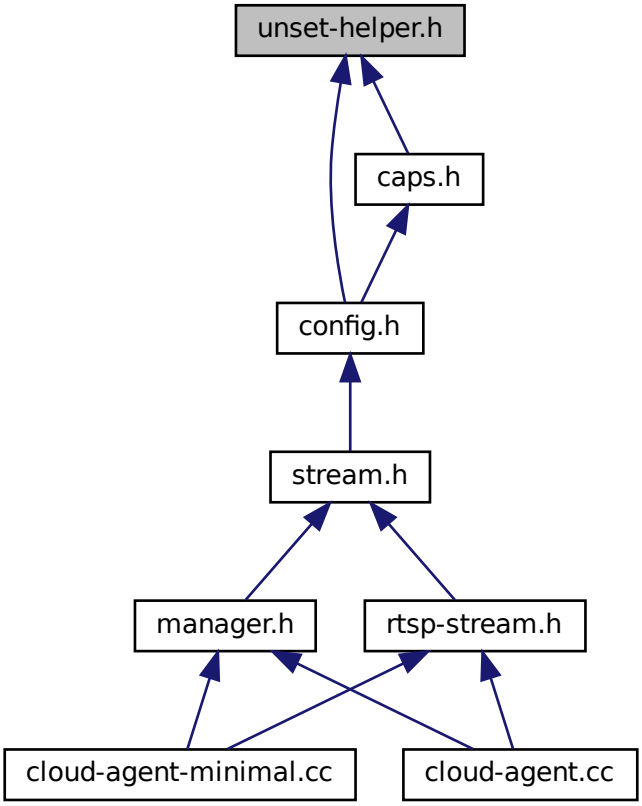
11.25 unset-helper.h File Reference

```
#include <chrono>
#include <limits>
#include <nlohmann/json.hpp>
#include <string>
#include <vector>
```

Include dependency graph for unset-helper.h:



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



Data Structures

- struct alter_bool

  alternative bool class Standard bool type has two states, this class adds 3rd state - undefined.

Namespaces

- vxg
- vxg::cloud
- vxg::cloud::time_spec

  time point

Functions

- **std::string** unset_value_for_impl (**std::string** ∗)
- int unset_value_for_impl (int ∗)

  Returns value of int type that can be treated as unset.
- double unset_value_for_impl (double ∗)

- uint64_t unset_value_for_impl (uint64_t *)
- int64_t unset_value_for_impl (int64_t *)
- vxg::cloud::time unset_value_for_impl (vxg::cloud::time *)
- vxg::cloud::duration unset_value_for_impl (vxg::cloud::duration *)
- nlohmann::json unset_value_for_impl (nlohmann::json *)
- template<typename T >
  T unset_value_for ()

  Template function which returns object value treated as 'unset' or uninitialized.
- template<typename T >
  std::vector< T > unset_value_for_impl (**std::vector**< T > *)
- template<typename T >
  T unset_value_for_impl (T *)
- template<typename T >
  bool __is_unset (T)

  Used for objects constructed from json, helps to check if original json object has specific field.
- template<> bool __is_unset< int > (int t)

  Predicate function checks if int value was not initialized.
- template<> bool __is_unset< std::string > (**std::string** t)
- template<> bool __is_unset< double > (double t)
- template<> bool __is_unset< vxg::cloud::time > (vxg::cloud::time t)
- template<> bool __is_unset< vxg::cloud::duration > (vxg::cloud::duration t)
- template<> bool __is_unset< nlohmann::json > (nlohmann::json t)
- template<> bool __is_unset< std::nullptr_t > (**std::nullptr_t** t)
- template<typename T >
  bool __is_unset (nlohmann::json t)
- template<> bool __is_unset< alter_bool > (alter_bool t)

Variables

- const **std::string** UnsetString
- const vxg::cloud::time UnsetTime
- const vxg::cloud::duration UnsetDuration
- const int UnsetInt
- const double UnsetFloat
- const double UnsetDouble
- const uint64_t UnsetUInt64
- const int64_t UnsetInt64

11.25.1 Function Documentation

11.25.1.1 __is_unset() [1/2]

```
template<typename T >
bool __is_unset (
            nlohmann::json t )  [inline]
```

Definition at line 155 of file unset-helper.h.

11.25.1.2 __is_unset() [2/2]

```
template<typename T >
bool __is_unset (
             T  )  [inline]
```

Used for objects constructed from json, helps to check if original json object has specific field.

You need to declare template specification for new types.

See also

__is_unset<int>(int t)

Template Parameters

T	object of type

Returns

true If object's field was actually set during construction, i.e. original json has such field in it's body.

false If object's field wasn't set, original json has no such field. It's also possible that json has such field but its value is set to value treated as unset value.

See also

__is_unset<>()

Definition at line 104 of file unset-helper.h.

11.25.1.3 __is_unset< alter_bool >()

```
template<>
bool __is_unset< alter_bool > (
             alter_bool t )  [inline]
```

Definition at line 219 of file unset-helper.h.

11.25.1.4 __is_unset< double >()

```
template<>
bool __is_unset< double > (
             double t )  [inline]
```

Definition at line 126 of file unset-helper.h.

11.25.1.5 __is_unset< int >()

```
template<>
bool __is_unset< int > (
             int t )  [inline]
```

Predicate function checks if int value was not initialized.

Template Parameters

int	

Parameters

t	

Returns

true value is uninitalized.

false value is initialized.

See also

unset_value_for<int>()

Definition at line 116 of file unset-helper.h.

11.25.1.6 __is_unset< nlohmann::json >()

```
template<>
bool __is_unset< nlohmann::json > (
            nlohmann::json t )  [inline]
```

Definition at line 141 of file unset-helper.h.

11.25.1.7 __is_unset< std::nullptr_t >()

```
template<>
bool __is_unset<  std::nullptr_t > (
              std::nullptr_t t )  [inline]
```

Definition at line 150 of file unset-helper.h.

11.25.1.8 __is_unset< std::string >()

```
template<>
bool __is_unset<  std::string > (
              std::string t )  [inline]
```

Definition at line 121 of file unset-helper.h.

11.25.1.9 __is_unset< vxg::cloud::duration >()

```
template<>
bool __is_unset< vxg::cloud::duration > (
            vxg::cloud::duration t )  [inline]
```

Definition at line 136 of file unset-helper.h.

11.25.1.10 __is_unset< vxg::cloud::time >()

```
template<>
bool __is_unset< vxg::cloud::time > (
            vxg::cloud::time t )  [inline]
```

Definition at line 131 of file unset-helper.h.

11.25.1.11 unset_value_for()

```
template<typename T >
T unset_value_for ( )
```

Template function which returns object value treated as 'unset' or uninitialized.

Template Parameters

T	

Returns

T Value equals to conditionally 'unset'.

Definition at line 73 of file unset-helper.h.

11.25.1.12 unset_value_for_impl() [1/10]

```
double unset_value_for_impl (
            double *  )  [inline]
```

Definition at line 39 of file unset-helper.h.

11.25.1.13 unset_value_for_impl() [2/10]

```
int unset_value_for_impl (
            int *  )  [inline]
```

Returns value of int type that can be treated as unset.

Returns

int

Definition at line 35 of file unset-helper.h.

11.25.1.14 unset_value_for_impl() [3/10]

```
int64_t unset_value_for_impl (
            int64_t *  )  [inline]
```

Definition at line 47 of file unset-helper.h.

11.25.1.15 unset_value_for_impl() [4/10]

```
nlohmann::json unset_value_for_impl (
            nlohmann::json *  )  [inline]
```

Definition at line 62 of file unset-helper.h.

11.25.1.16 unset_value_for_impl() [5/10]

```
 std::string unset_value_for_impl (
             std::string *  )  [inline]
```

Definition at line 27 of file unset-helper.h.

11.25.1.17 unset_value_for_impl() [6/10]

```
template<typename T >
std::vector<T> unset_value_for_impl (
             std::vector< T > *  )  [inline]
```

Definition at line 78 of file unset-helper.h.

11.25.1.18 unset_value_for_impl() [7/10]

```
template<typename T >
T unset_value_for_impl (
             T * )
```

Definition at line 85 of file unset-helper.h.

11.25.1.19 unset_value_for_impl() [8/10]

```
uint64_t unset_value_for_impl (
             uint64_t * ) [inline]
```

Definition at line 43 of file unset-helper.h.

11.25.1.20 unset_value_for_impl() [9/10]

```
vxg::cloud::duration unset_value_for_impl (
             vxg::cloud::duration * ) [inline]
```

Definition at line 57 of file unset-helper.h.

11.25.1.21 unset_value_for_impl() [10/10]

```
vxg::cloud::time unset_value_for_impl (
             vxg::cloud::time * ) [inline]
```

Definition at line 51 of file unset-helper.h.

11.25.2 Variable Documentation

11.25.2.1 UnsetDouble

```
const double UnsetDouble
```

Definition at line 229 of file unset-helper.h.

11.25.2.2 UnsetDuration

```
const vxg::cloud::duration UnsetDuration
```

Definition at line 225 of file unset-helper.h.

11.25.2.3 UnsetFloat

```
const double UnsetFloat
```

Definition at line 228 of file unset-helper.h.

11.25.2.4 UnsetInt

```
const int UnsetInt
```

Definition at line 227 of file unset-helper.h.

11.25.2.5 UnsetInt64

```
const int64_t UnsetInt64
```

Definition at line 231 of file unset-helper.h.

11.25.2.6 UnsetString

```
const  std::string UnsetString
```

Definition at line 223 of file unset-helper.h.

11.25.2.7 UnsetTime

```
const vxg::cloud::time UnsetTime
```

Definition at line 224 of file unset-helper.h.

11.25.2.8 UnsetUInt64

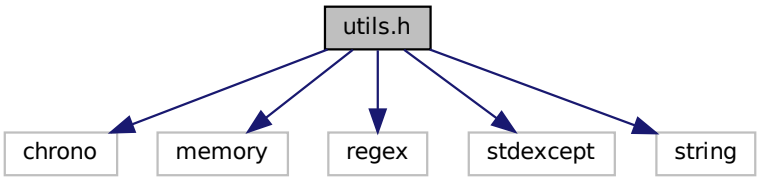
```
const uint64_t UnsetUInt64
```

Definition at line 230 of file unset-helper.h.

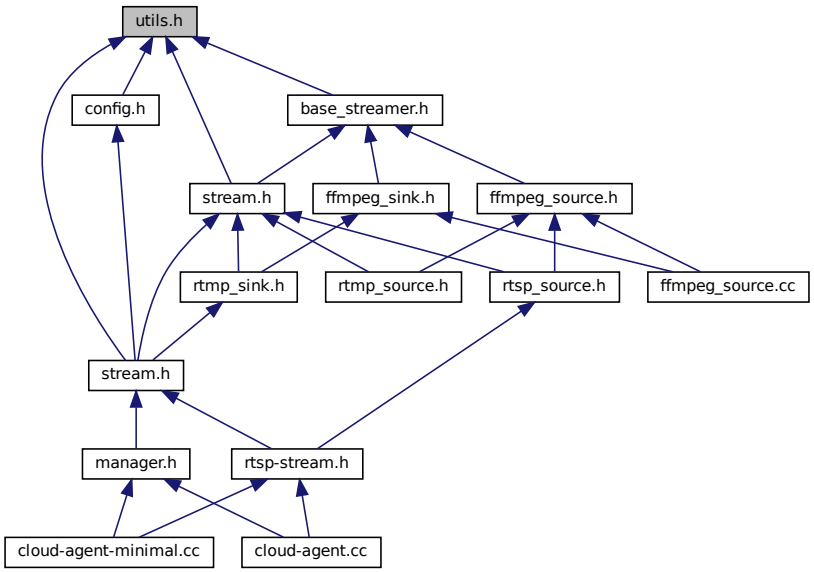
11.26 utils.h File Reference

```
#include <chrono>
#include <memory>
#include <regex>
#include <stdexcept>
#include <string>
```

Include dependency graph for utils.h:



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



Data Structures

- struct vxg::cloud::utils::uri
- struct vxg::cloud::utils::motion::map

Namespaces

- vxg
- vxg::cloud
- vxg::cloud::time_spec
  time point
- vxg::cloud::utils
- vxg::cloud::utils::time
- vxg::cloud::utils::motion
- vxg::cloud::utils::gcc_abi
- std

Functions

- void vxg::cloud::utils::set_thread_name (**std::string** name)
- cloud::time vxg::cloud::utils::time::now ()
- **std::string** vxg::cloud::utils::time::time_to_ISO8601 (**std::time_t**)
- **std::string** vxg::cloud::utils::time::time_to_ISO8601_packed (**std::time_t**)
- **std::string** vxg::cloud::utils::time::now_ISO8601_UTC ()
- **std::string** vxg::cloud::utils::time::now_ISO8601_UTC_packed ()
- **std::time_t** vxg::cloud::utils::time::now_time_UTC ()
- **std::time_t** vxg::cloud::utils::time::ISO8601_to_time (const **std::string** &input)
- **std::string** vxg::cloud::utils::time::to_iso_8601 (cloud::time t)
- **std::string** vxg::cloud::utils::time::to_iso (cloud::time t)
- **std::string** vxg::cloud::utils::time::to_iso2 (cloud::time t)
- **std::string** vxg::cloud::utils::time::to_iso_packed (cloud::time t)
- **std::string** vxg::cloud::utils::time::to_iso_local (cloud::time t)
- cloud::time vxg::cloud::utils::time::from_double (double t)
- double vxg::cloud::utils::time::to_double (cloud::time t)
- cloud::time vxg::cloud::utils::time::from_iso (**std::string** st)
- cloud::time vxg::cloud::utils::time::from_iso2 (**std::string** st)
- cloud::time vxg::cloud::utils::time::from_iso_packed (**std::string** st)
- bool vxg::cloud::utils::time::iso_time_valid (const **std::string** &s)
- cloud::time vxg::cloud::utils::time::null ()
- cloud::time vxg::cloud::utils::time::max ()
- bool vxg::cloud::utils::time::is_iso_packed (const **std::string** &s)
- bool vxg::cloud::utils::time::is_iso (const **std::string** &s)
- template<typename... Args>
  std::string vxg::cloud::utils::string_format (const **std::string** &format, Args... args)
- **std::string** vxg::cloud::utils::string_trim (const **std::string** &name, **std::regex** regx)
- **std::string** vxg::cloud::utils::string_trim (const **std::string** &name)
- **std::vector**< **std::string** > vxg::cloud::utils::string_split (const **std::string** &s, char delimiter)
- bool vxg::cloud::utils::string_startswith (**std::string** const &fullString, **std::string** const &start)
- bool vxg::cloud::utils::string_endswith (**std::string** const &fullString, **std::string** const &ending)
- bool vxg::cloud::utils::string_replace (**std::string** &str, const **std::string** &from, const **std::string** &to)
- **std::string** vxg::cloud::utils::string_urlencode (const **std::string** &value)
- **std::string** vxg::cloud::utils::string_urldecode (const **std::string** &text)
- **std::string** vxg::cloud::utils::string_tolower (const **std::string** &s)

- **std::string** vxg::cloud::utils::string_toupper (const **std::string** &s)
- bool vxg::cloud::utils::string_contains (**std::string** s, char c)
- **std::string** vxg::cloud::utils::dirname (const **std::string** &filepath)
- **std::string** vxg::cloud::utils::gcc_abi::demangle (**std::string** name)
- template<typename T , typename... CONSTRUCTOR_ARGS>
  std::unique_ptr< T > std::make_unique (CONSTRUCTOR_ARGS &&... constructor_args)

Index