Aries Cloud Agent Python Documentation

Nicholas Rempel, Andrew Whitehead, Ian Costanzo, Stephen Klu

Apr 23, 2020

Aries Cloud Agent Python - Modules

Hyperledger Aries Cloud Agent Python (ACA-Py) is a foundation for building decentralized identity applications and services running in non-mobile environments.

This is the Read The Docs site for the Hyperledger Aries Cloud Agent Python. This site contains only the ACA-Py docstrings documentation extracted from the Python Code. For other documentation, please consult the links in the Readme for the ACA-Py GitHub Repo.

If you are getting started with verifiable credentials or Aries, we recommend that you start with this verifiable credentials and agents getting started guide.

Want to quick overview of the deployment model for ACA-Py? See this document.

To investigate the code, use search or click the package links in the left menu to drill into the modules, subpackages and submodules that make up ACA-Py.

Developers that are interested in what DIDComm protocols are supported in ACA-Py should take a look at the protocols package. These should align with the corresponding aries-rfcs protocols. Decorators defined in aries-rfcs and implemented in ACA-Py can be found here. Some general purpose subpackages that might be of interest include wallet and storage. For those agents playing different roles in a verifiable credential exchange, take a look at the issuer, holder and verifier packages.

Please see the ACA-Py Contribution guidelines for how to contribute to ACA-Py, including for how to submit issues about ACA-Py.

CHAPTER 1

aries_cloudagent package

Aries Cloud Agent.

1.1 Subpackages

1.1.1 aries_cloudagent.admin package

Submodules

aries_cloudagent.admin.base_server module

Abstract admin server interface.

class aries_cloudagent.admin.base_server.**BaseAdminServer**

Bases: abc.ABC

Admin HTTP server class.

add_webhook_target(*target_url: str, topic_filter: Sequence[str] = None, max_attempts: int = None*)

Add a webhook target.

remove_webhook_target(*target_url: str*)

Remove a webhook target.

send_webhook(*topic: str, payload: dict*)

Add a webhook to the queue, to send to all registered targets.

start() → None

Start the webserver.

Raises AdminSetupError – If there was an error starting the webserver

stop() → None

Stop the webserver.

aries_cloudagent.admin.error module

Admin error classes.

exception aries_cloudagent.admin.error.**AdminError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base class for Admin-related errors.

exception aries_cloudagent.admin.error.**AdminSetupError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.admin.error.AdminError*

Admin server setup or configuration error.

aries_cloudagent.admin.server module

Admin server classes.

class aries_cloudagent.admin.server.**AdminModulesSchema**(**args*, ***kwargs*)

Bases: sphinx.ext.autodoc.importer._MockObject

Schema for the modules endpoint.

result

Used by autodoc_mock_imports.

class aries_cloudagent.admin.server.**AdminResponder**(*context: aries_cloudagent.config.injection_context.InjectionContext send: Coroutine[T_co, T_contra, V_co], webhook: Coroutine[T_co, T_contra, V_co], **kwargs*)

Bases: *aries_cloudagent.messaging.responder.BaseResponder*

Handle outgoing messages from message handlers.

send_outbound(*message: aries_cloudagent.transport.outbound.message.OutboundMessage*)

Send outbound message.

Parameters **message** – The *OutboundMessage* to be sent

send_webhook(*topic: str*, *payload: dict*)

Dispatch a webhook.

Parameters

- **topic** – the webhook topic identifier
- **payload** – the webhook payload value

class aries_cloudagent.admin.server.**AdminServer**(*host: str, port: int, context: aries_cloudagent.config.injection_context.InjectionContext, outbound_message_router: Coroutine[T_co, T_contra, V_co], webhook_router: Callable, task_queue: aries_cloudagent.utils.task_queue.TaskQueue = None, conductor_stats: Coroutine[T_co, T_contra, V_co] = None*)

Bases: *aries_cloudagent.admin.base_server.BaseAdminServer*

Admin HTTP server class.

add_webhook_target(*target_url: str, topic_filter: Sequence[str] = None, max_attempts: int = None*)
Add a webhook target.

make_application() → <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4a5f6d8>
Get the aiohttp application instance.

on_startup(*app: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4a5f6d8>*)
Perform webserver startup actions.

plugins_handler(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4a5f6d8>*)
Request handler for the loaded plugins list.

Parameters **request** – aiohttp request object

Returns The module list response

redirect_handler(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4a5f6d8>*)
Perform redirect to documentation.

remove_webhook_target(*target_url: str*)
Remove a webhook target.

send_webhook(*topic: str, payload: dict*)
Add a webhook to the queue, to send to all registered targets.

start() → None
Start the webserver.

Raises `AdminSetupError` – If there was an error starting the webserver

status_handler(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4a5f6d8>*)
Request handler for the server status information.

Parameters **request** – aiohttp request object

Returns The web response

status_reset_handler(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4a5f6d8>*)
Request handler for resetting the timing statistics.

Parameters **request** – aiohttp request object

Returns The web response

stop() → None
Stop the webserver.

websocket_handler(*request*)
Send notifications to admin client over websocket.

class aries_cloudagent.admin.server.**AdminStatusSchema**(**args, **kwargs*)
Bases: `sphinx.ext.autodoc.importer._MockObject`

Schema for the status endpoint.

class aries_cloudagent.admin.server.**WebhookTarget**(*endpoint: str, topic_filter: Sequence[str] = None, max_attempts: int = None*)
Bases: `object`

Class for managing webhook target information.

topic_filter
Accessor for the target's topic filter.

1.1.2 aries_cloudagent.cache package

Submodules

aries_cloudagent.cache.base module

Abstract base classes for cache.

class aries_cloudagent.cache.base.**BaseCache**
Bases: abc.ABC

Abstract cache interface.

acquire(*key: str*)
Acquire a lock on a given cache key.

clear(*key: str*)
Remove an item from the cache, if present.

Parameters **key** – the key to remove

flush()
Remove all items from the cache.

get(*key: str*)
Get an item from the cache.

Parameters **key** – the key to retrieve an item for

Returns The record found or *None*

release(*key: str*)
Release the lock on a given cache key.

set(*keys: Union[str, Sequence[str]], value: Any, ttl: int = None*)
Add an item to the cache with an optional ttl.

Parameters

- **keys** – the key or keys for which to set an item
- **value** – the value to store in the cache
- **ttl** – number of second that the record should persist

exception aries_cloudagent.cache.base.**CacheError**(**args, error_code: str = None, **kwargs*)
Bases: *aries_cloudagent.core.error.BaseError*

Base class for cache-related errors.

class aries_cloudagent.cache.base.**CacheKeyLock**(*cache: aries_cloudagent.cache.base.BaseCache, key: str*)
Bases: object

A lock on a particular cache key.

Used to prevent multiple async threads from generating or querying the same semi-expensive data. Not thread safe.

done
Accessor for the done state.

future
Fetch the result in the form of an awaitable future.

parent
Accessor for the parent key lock, if any.

release()
Release the cache lock.

result
Fetch the current result, if any.

set_result(*value: Any*, *ttl: int = None*)
Set the result, updating the cache and any waiters.

aries_cloudagent.cache.basic module

Basic in-memory cache implementation.

class aries_cloudagent.cache.basic.**BasicCache**
Bases: *aries_cloudagent.cache.base.BaseCache*

Basic in-memory cache class.

clear(*key: str*)
Remove an item from the cache, if present.

Parameters **key** – the key to remove

flush()
Remove all items from the cache.

get(*key: str*)
Get an item from the cache.

Parameters **key** – the key to retrieve an item for

Returns The record found or *None*

set(*keys: Union[str, Sequence[str]], value: Any, ttl: int = None*)
Add an item to the cache with an optional ttl.

Overwrites existing cache entries.

Parameters

- **keys** – the key or keys for which to set an item
- **value** – the value to store in the cache
- **ttl** – number of seconds that the record should persist

1.1.3 aries_cloudagent.commands package

Commands module common setup.

aries_cloudagent.commands.**available_commands**()
Index available commands.

`aries_cloudagent.commands.load_command`(*command: str*)
Load the module corresponding with a named command.

`aries_cloudagent.commands.run_command`(*command: str*, *argv: Sequence[str] = None*)
Execute a named command with command line arguments.

Submodules

aries_cloudagent.commands.help module

Help command for indexing available commands.

`aries_cloudagent.commands.help.execute`(*argv: Sequence[str] = None*)
Execute the help command.

aries_cloudagent.commands.provision module

Provision command for setting up agent settings before starting.

exception `aries_cloudagent.commands.provision.ProvisionError`(**args*, *error_code: str = None*, ***kwargs*)
Bases: *aries_cloudagent.core.error.BaseError*

Base exception for provisioning errors.

`aries_cloudagent.commands.provision.execute`(*argv: Sequence[str] = None*)
Entrypoint.

`aries_cloudagent.commands.provision.init_argument_parser`(*parser: argparse.ArgumentParser*)
Initialize an argument parser with the module's arguments.

`aries_cloudagent.commands.provision.provision`(*settings: dict*)
Perform provisioning.

aries_cloudagent.commands.start module

Entrypoint.

`aries_cloudagent.commands.start.execute`(*argv: Sequence[str] = None*)
Entrypoint.

`aries_cloudagent.commands.start.init_argument_parser`(*parser: argparse.ArgumentParser*)
Initialize an argument parser with the module's arguments.

`aries_cloudagent.commands.start.run_loop`(*startup: Coroutine[T_co, T_contra, V_co], shutdown: Coroutine[T_co, T_contra, V_co]*)
Execute the application, handling signals and ctrl-c.

`aries_cloudagent.commands.start.shutdown_app`(*conductor: aries_cloudagent.core.conductor.Conductor*)
Shut down.

`aries_cloudagent.commands.start.start_app`(*conductor: aries_cloudagent.core.conductor.Conductor*)
Start up.

1.1.4 aries_cloudagent.config package

Submodules

aries_cloudagent.config.argparse module

Command line option parsing.

class aries_cloudagent.config.argparse.**AdminGroup**

Bases: *aries_cloudagent.config.argparse.ArgumentGroup*

Admin server settings.

CATEGORIES = ('start',)

GROUP_NAME = 'Admin'

add_arguments(*parser: argparse.ArgumentParser*)

Add admin-specific command line arguments to the parser.

get_settings(*args: argparse.Namespace*)

Extract admin settings.

class aries_cloudagent.config.argparse.**ArgumentGroup**

Bases: abc.ABC

A class representing a group of related command line arguments.

GROUP_NAME = None

add_arguments(*parser: argparse.ArgumentParser*)

Add arguments to the provided argument parser.

get_settings(*args: argparse.Namespace*) → dict

Extract settings from the parsed arguments.

class aries_cloudagent.config.argparse.**DebugGroup**

Bases: *aries_cloudagent.config.argparse.ArgumentGroup*

Debug settings.

CATEGORIES = ('start',)

GROUP_NAME = 'Debug'

add_arguments(*parser: argparse.ArgumentParser*)

Add debug command line arguments to the parser.

get_settings(*args: argparse.Namespace*) → dict

Extract debug settings.

class aries_cloudagent.config.argparse.**GeneralGroup**

Bases: *aries_cloudagent.config.argparse.ArgumentGroup*

General settings.

CATEGORIES = ('general', 'start')

GROUP_NAME = 'General'

add_arguments(*parser: argparse.ArgumentParser*)

Add general command line arguments to the parser.

get_settings(*args: argparse.Namespace*) → dict

Extract general settings.

class aries_cloudagent.config.argparse.**LedgerGroup**

Bases: *aries_cloudagent.config.argparse.ArgumentGroup*

Ledger settings.

CATEGORIES = ('start', 'general')

GROUP_NAME = 'Ledger'

add_arguments(*parser: argparse.ArgumentParser*)

Add ledger-specific command line arguments to the parser.

get_settings(*args: argparse.Namespace*) → dict

Extract ledger settings.

class aries_cloudagent.config.argparse.**LoggingGroup**

Bases: *aries_cloudagent.config.argparse.ArgumentGroup*

Logging settings.

CATEGORIES = ('general', 'start')

GROUP_NAME = 'Logging'

add_arguments(*parser: argparse.ArgumentParser*)

Add logging-specific command line arguments to the parser.

get_settings(*args: argparse.Namespace*) → dict

Extract logging settings.

class aries_cloudagent.config.argparse.**ProtocolGroup**

Bases: *aries_cloudagent.config.argparse.ArgumentGroup*

Protocol settings.

CATEGORIES = ('start',)

GROUP_NAME = 'Protocol'

add_arguments(*parser: argparse.ArgumentParser*)

Add protocol-specific command line arguments to the parser.

get_settings(*args: argparse.Namespace*) → dict

Get protocol settings.

class aries_cloudagent.config.argparse.**TransportGroup**

Bases: *aries_cloudagent.config.argparse.ArgumentGroup*

Transport settings.

CATEGORIES = ('start',)

GROUP_NAME = 'Transport'

add_arguments(*parser: argparse.ArgumentParser*)

Add transport-specific command line arguments to the parser.

get_settings(*args: argparse.Namespace*)

Extract transport settings.

class aries_cloudagent.config.argparse.**WalletGroup**

Bases: *aries_cloudagent.config.argparse.ArgumentGroup*

Wallet settings.

CATEGORIES = ('general', 'start')

GROUP_NAME = 'Wallet'

add_arguments(*parser: argparse.ArgumentParser*)
Add wallet-specific command line arguments to the parser.

get_settings(*args: argparse.Namespace*) → dict
Extract wallet settings.

class aries_cloudagent.config.argparse.**group**(**categories*)
Bases: object

Decorator for registering argument groups.

classmethod get_registered(*category: str = None*)
Fetch the set of registered classes in a category.

aries_cloudagent.config.argparse.**load_argument_groups**(*parser: argparse.ArgumentParser, *groups*)
Log a set of argument groups into a parser.

Returns A callable to convert loaded arguments into a settings dictionary

aries_cloudagent.config.base module

Configuration base classes.

class aries_cloudagent.config.base.**BaseInjector**
Bases: abc.ABC

Base injector class.

copy() → aries_cloudagent.config.base.BaseInjector
Produce a copy of the injector instance.

inject(*base_cls: type, settings: Mapping[str, object] = None, *, required: bool = True*) → object
Get the provided instance of a given class identifier.

Parameters

- **cls** – The base class to retrieve an instance of
- **settings** – An optional mapping providing configuration to the provider

Returns An instance of the base class, or None

class aries_cloudagent.config.base.**BaseProvider**
Bases: abc.ABC

Base provider class.

provide(*settings: aries_cloudagent.config.base.BaseSettings, injector: aries_cloudagent.config.base.BaseInjector*)
Provide the object instance given a config and injector.

class aries_cloudagent.config.base.**BaseSettings**
Bases: collections.abc.Mapping, typing.Generic

Base settings class.

copy() → aries_cloudagent.config.base.BaseSettings
Produce a copy of the settings instance.

extend(*other: Mapping[str, object]*) → aries_cloudagent.config.base.BaseSettings
Merge another mapping to produce a new settings instance.

get_bool(**var_names*, *default=None*) → bool
Fetch a setting as a boolean value.

Parameters

- **var_names** – A list of variable name alternatives
- **default** – The default value to return if none are defined

get_int(**var_names*, *default=None*) → int
Fetch a setting as an integer value.

Parameters

- **var_names** – A list of variable name alternatives
- **default** – The default value to return if none are defined

get_str(**var_names*, *default=None*) → str
Fetch a setting as a string value.

Parameters

- **var_names** – A list of variable name alternatives
- **default** – The default value to return if none are defined

get_value(**var_names*, *default=None*)
Fetch a setting.

Parameters

- **var_names** – A list of variable name alternatives
- **default** – The default value to return if none are defined

Returns The setting value, if defined, otherwise the default value

exception aries_cloudagent.config.base.**ConfigError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

A base exception for all configuration errors.

exception aries_cloudagent.config.base.**InjectorError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.config.base.ConfigError*

The base exception raised by *BaseInjector* implementations.

exception aries_cloudagent.config.base.**ProviderError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.config.base.ConfigError*

The base exception raised by *BaseProvider* implementations.

exception aries_cloudagent.config.base.**SettingsError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.config.base.ConfigError*

The base exception raised by *BaseSettings* implementations.

aries_cloudagent.config.base_context module

Base injection context builder classes.

class aries_cloudagent.config.base_context.**ContextBuilder**(*settings: Mapping[str, object] = None*)

Bases: abc.ABC

Base injection context builder class.

build() → aries_cloudagent.config.injection_context.InjectionContext
Build the new injection context.

update_settings(*settings: Mapping[str, object]*)
Update the context builder with additional settings.

aries_cloudagent.config.default_context module

Classes for configuring the default injection context.

class aries_cloudagent.config.default_context.**DefaultContextBuilder**(*settings: Mapping[str, object] = None*)

Bases: *aries_cloudagent.config.base_context.ContextBuilder*

Default context builder.

bind_providers(*context: aries_cloudagent.config.injection_context.InjectionContext*)
Bind various class providers.

build() → aries_cloudagent.config.injection_context.InjectionContext
Build the new injection context.

load_plugins(*context: aries_cloudagent.config.injection_context.InjectionContext*)
Set up plugin registry and load plugins.

aries_cloudagent.config.error module

Errors for config modules.

exception aries_cloudagent.config.error.**ArgsParseError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.config.base.ConfigError*

Error raised when there is a problem parsing the command-line arguments.

aries_cloudagent.config.injection_context module

Injection context implementation.

class aries_cloudagent.config.injection_context.**InjectionContext**(**, settings: Mapping[str, object] = None, enforce_typing: bool = True*)

Bases: *aries_cloudagent.config.base.BaseInjector*

Manager for configuration settings and class providers.

ROOT_SCOPE = 'application'

copy() → aries_cloudagent.config.injection_context.InjectionContext
Produce a copy of the injector instance.

inject(*base_cls: type*, *settings: Mapping[str, object] = None*, *, *required: bool = True*) → object
Get the provided instance of a given class identifier.

Parameters

- **cls** – The base class to retrieve an instance of
- **settings** – An optional mapping providing configuration to the provider

Returns An instance of the base class, or None

injector
Accessor for scope-specific injector.

injector_for_scope(*scope_name: str*) → aries_cloudagent.config.injector.Injector
Fetch the injector for a specific scope.

Parameters **scope_name** – The unique scope identifier

scope_name
Accessor for the current scope name.

settings
Accessor for scope-specific settings.

start_scope(*scope_name: str*, *settings: Mapping[str, object] = None*) → aries_cloudagent.config.injection_context.InjectionContext
Begin a new named scope.

Parameters

- **scope_name** – The unique name for the scope being entered
- **settings** – An optional mapping of additional settings to apply

Returns A new injection context representing the scope

update_settings(*settings: Mapping[str, object]*)
Update the scope with additional settings.

exception aries_cloudagent.config.injection_context.**InjectionContextError**(**args*, *error_code: str = None*, ***kwargs*)
Bases: *aries_cloudagent.config.base.InjectorError*

Base class for issues in the injection context.

class aries_cloudagent.config.injection_context.**Scope**(*name*, *injector*)
Bases: tuple

injector
Alias for field number 1

name
Alias for field number 0

aries_cloudagent.config.injector module

Standard Injector implementation.

class aries_cloudagent.config.injector.**Injector**(*settings: Mapping[str, object] = None, enforce_typing: bool = True*)

Bases: *aries_cloudagent.config.base.BaseInjector*

Injector implementation with static and dynamic bindings.

bind_instance(*base_cls: type, instance: object*)
Add a static instance as a class binding.

bind_provider(*base_cls: type, provider: aries_cloudagent.config.base.BaseProvider, *, cache: bool = False*)
Add a dynamic instance resolver as a class binding.

clear_binding(*base_cls: type*)
Remove a previously-added binding.

copy() → aries_cloudagent.config.base.BaseInjector
Produce a copy of the injector instance.

get_provider(*base_cls: type*)
Find the provider associated with a class binding.

inject(*base_cls: type, settings: Mapping[str, object] = None, *, required: bool = True*)
Get the provided instance of a given class identifier.

Parameters

- **cls** – The base class to retrieve an instance of
- **params** – An optional dict providing configuration to the provider

Returns An instance of the base class, or None

settings
Accessor for scope-specific settings.

aries_cloudagent.config.ledger module

Ledger configuration.

aries_cloudagent.config.ledger.**accept_taa**(*ledger: aries_cloudagent.ledger.base.BaseLedger, taa_info, provision: bool = False*) → bool
Perform TAA acceptance.

aries_cloudagent.config.ledger.**fetch_genesis_transactions**(*genesis_url: str*) → str
Get genesis transactions.

aries_cloudagent.config.ledger.**ledger_config**(*context: aries_cloudagent.config.injection_context.InjectionContext, public_did: str, provision: bool = False*) → bool
Perform Indy ledger configuration.

aries_cloudagent.config.logging module

Utilities related to logging.

class aries_cloudagent.config.logging.**LoggingConfigurator**

Bases: object

Utility class used to configure logging and print an informative start banner.

classmethod configure(*logging_config_path: str = None, log_level: str = None, log_file: str = None*)

Configure logger.

Parameters

- **logging_config_path** – str: (Default value = None) Optional path to custom logging config
- **log_level** – str: (Default value = None)

classmethod print_banner(*agent_label, inbound_transports, outbound_transports, public_did, admin_server=None, banner_length=40, border_character=':'*)

Print a startup banner describing the configuration.

Parameters

- **agent_label** – Agent Label
- **inbound_transports** – Configured inbound transports
- **outbound_transports** – Configured outbound transports
- **admin_server** – Admin server info
- **public_did** – Public DID
- **banner_length** – (Default value = 40) Length of the banner
- **border_character** – (Default value = ":") Character to use in banner
- **border** –

aries_cloudagent.config.logging.**load_resource**(*path: str, encoding: str = None*) → TextIO

Open a resource file located in a python package or the local filesystem.

Parameters **path** – The resource path in the form of *dir/file* or *package:dir/file*

Returns A file-like object representing the resource

aries_cloudagent.config.provider module

Service provider implementations.

class aries_cloudagent.config.provider.**CachedProvider**(*provider: aries_cloudagent.config.base.BaseProvider*)

Bases: aries_cloudagent.config.base.BaseProvider

Cache the result of another provider.

provide(*config: aries_cloudagent.config.base.BaseSettings, injector: aries_cloudagent.config.base.BaseInjector*)

Provide the object instance given a config and injector.

class aries_cloudagent.config.provider.**ClassProvider**(*instance_cls: Union[str, type], *ctor_args, async_init: str = None, **ctor_kwargs*)

Bases: aries_cloudagent.config.base.BaseProvider

Provider for a particular class.

class Inject(*base_cls: type*)
Bases: object

A class for passing injected arguments to the constructor.

provide(*config: aries_cloudagent.config.base.BaseSettings, injector: aries_cloudagent.config.base.BaseInjector*)
Provide the object instance given a config and injector.

class aries_cloudagent.config.provider.**InstanceProvider**(*instance*)
Bases: *aries_cloudagent.config.base.BaseProvider*

Provider for a previously-created instance.

provide(*config: aries_cloudagent.config.base.BaseSettings, injector: aries_cloudagent.config.base.BaseInjector*)
Provide the object instance given a config and injector.

class aries_cloudagent.config.provider.**StatsProvider**(*provider: aries_cloudagent.config.base.BaseProvider, methods: Sequence[str], *, ignore_missing: bool = True*)
Bases: *aries_cloudagent.config.base.BaseProvider*

Add statistics to the results of another provider.

provide(*config: aries_cloudagent.config.base.BaseSettings, injector: aries_cloudagent.config.base.BaseInjector*)
Provide the object instance given a config and injector.

aries_cloudagent.config.settings module

Settings implementation.

class aries_cloudagent.config.settings.**Settings**(*values: Mapping[str, object] = None*)
Bases: *aries_cloudagent.config.base.BaseSettings*

Mutable settings implementation.

clear_value(*var_name: str*)
Remove a setting.

Parameters **var_name** – The name of the setting

copy() → aries_cloudagent.config.base.BaseSettings
Produce a copy of the settings instance.

extend(*other: Mapping[str, object]*) → aries_cloudagent.config.base.BaseSettings
Merge another settings instance to produce a new instance.

get_value(**var_names, default=None*)
Fetch a setting.

Parameters

- **var_names** – A list of variable name alternatives
- **default** – The default value to return if none are defined

set_default(*var_name: str, value*)
Add a setting if not currently defined.

Parameters

- **var_name** – The name of the setting
- **value** – The value to assign

set_value(*var_name: str*, *value*)
Add a setting.

Parameters

- **var_name** – The name of the setting
- **value** – The value to assign

aries_cloudagent.config.util module

Entrypoint.

class aries_cloudagent.config.util.**ByteSize**(*min_size: int = 0*, *max_size: int = 0*)
Bases: object

Argument value parser for byte sizes.

aries_cloudagent.config.util.**common_config**(*settings: Mapping[str, Any]*)
Perform common app configuration.

aries_cloudagent.config.wallet module

Wallet configuration.

aries_cloudagent.config.wallet.**wallet_config**(*context: aries_cloudagent.config.injection_context.InjectionContext*, *provision: bool = False*)
Initialize the wallet.

1.1.5 aries_cloudagent.core package

Submodules

aries_cloudagent.core.conductor module

The Conductor.

The conductor is responsible for coordinating messages that are received over the network, communicating with the ledger, passing messages to handlers, instantiating concrete implementations of required modules and storing data in the wallet.

class aries_cloudagent.core.conductor.**Conductor**(*context_builder: aries_cloudagent.config.base_context.ContextBuilder*)
Bases: object

Conductor class.

Class responsible for initializing concrete implementations of our require interfaces and routing inbound and outbound message data.

dispatch_complete(*message: aries_cloudagent.transport.inbound.message.InboundMessage*, *completed: aries_cloudagent.utils.task_queue.CompletedTask*)
Handle completion of message dispatch.

get_stats() → dict
Get the current stats tracked by the conductor.

handle_not_delivered(*context: aries_cloudagent.config.injection_context.InjectionContext, outbound: aries_cloudagent.transport.outbound.message.OutboundMessage*)
Handle a message that failed delivery via outbound transports.

handle_not_returned(*context: aries_cloudagent.config.injection_context.InjectionContext, outbound: aries_cloudagent.transport.outbound.message.OutboundMessage*)
Handle a message that failed delivery via an inbound session.

inbound_message_router(*message: aries_cloudagent.transport.inbound.message.InboundMessage, can_respond: bool = False*)
Route inbound messages.

Parameters

- **message** – The inbound message instance
- **can_respond** – If the session supports return routing

outbound_message_router(*context: aries_cloudagent.config.injection_context.InjectionContext, outbound: aries_cloudagent.transport.outbound.message.OutboundMessage, inbound: aries_cloudagent.transport.inbound.message.InboundMessage = None*) → None
Route an outbound message.

Parameters

- **context** – The request context
- **message** – An outbound message to be sent
- **inbound** – The inbound message that produced this response, if available

queue_outbound(*context: aries_cloudagent.config.injection_context.InjectionContext, outbound: aries_cloudagent.transport.outbound.message.OutboundMessage, inbound: aries_cloudagent.transport.inbound.message.InboundMessage = None*)
Queue an outbound message.

Parameters

- **context** – The request context
- **message** – An outbound message to be sent
- **inbound** – The inbound message that produced this response, if available

setup()
Initialize the global request context.

start() → None
Start the agent.

stop(*timeout=1.0*)
Stop the agent.

webhook_router(*topic: str, payload: dict, endpoint: str, max_attempts: int = None*)
Route a webhook through the outbound transport manager.

Parameters

- **topic** – The webhook topic
- **payload** – The webhook payload
- **endpoint** – The endpoint of the webhook target
- **max_attempts** – The maximum number of attempts

aries_cloudagent.core.dispatcher module

The Dispatcher.

The dispatcher is responsible for coordinating data flow between handlers, providing lifecycle hook callbacks storing state for message threads, etc.

class aries_cloudagent.core.dispatcher.**Dispatcher**(*context: aries_cloudagent.config.injection_context.InjectionContext*)

Bases: object

Dispatcher class.

Class responsible for dispatching messages to message handlers and responding to other agents.

complete(*timeout: float = 0.1*)

Wait for pending tasks to complete.

handle_message(*inbound_message: aries_cloudagent.transport.inbound.message.InboundMessage, send_outbound: Coroutine[T_co, T_contra, V_co], send_webhook: Coroutine[T_co, T_contra, V_co] = None*)

Configure responder and message context and invoke the message handler.

Parameters

- **inbound_message** – The inbound message instance
- **send_outbound** – Async function to send outbound messages
- **send_webhook** – Async function to dispatch a webhook

Returns The response from the handler

log_task(*task: aries_cloudagent.utils.task_queue.CompletedTask*)

Log a completed task using the stats collector.

make_message(*parsed_msg: dict*) → aries_cloudagent.messaging.agent_message.AgentMessage

Deserialize a message dict into the appropriate message instance.

Given a dict describing a message, this method returns an instance of the related message class.

Parameters **parsed_msg** – The parsed message

Returns An instance of the corresponding message class for this message

Raises

- MessageParseError – If the message doesn't specify @type
- MessageParseError – If there is no message class registered to handle
- the given type

put_task(*coro: Coroutine[T_co, T_contra, V_co], complete: Callable = None, ident: str = None*) → aries_cloudagent.utils.task_queue.PendingTask

Run a task in the task queue, potentially blocking other handlers.

queue_message(*inbound_message: aries_cloudagent.transport.inbound.message.InboundMessage, send_outbound: Coroutine[T_co, T_contra, V_co], send_webhook: Coroutine[T_co, T_contra, V_co] = None, complete: Callable = None*) → aries_cloudagent.utils.task_queue.PendingTask

Add a message to the processing queue for handling.

Parameters

- **inbound_message** – The inbound message instance

- **send_outbound** – Async function to send outbound messages
- **send_webhook** – Async function to dispatch a webhook
- **complete** – Function to call when the handler has completed

Returns A pending task instance resolving to the handler task

run_task(*coro: Coroutine[T_co, T_contra, V_co], complete: Callable = None, ident: str = None*) → _asyncio.Task

Run a task in the task queue, potentially blocking other handlers.

setup()

Perform async instance setup.

class aries_cloudagent.core.dispatcher.**DispatcherResponder**(*context: aries_cloudagent.messaging.request_context.R inbound_message: aries_cloudagent.transport.inbound.message.In send_outbound: Coroutine[T_co, T_contra, V_co], send_webhook: Coroutine[T_co, T_contra, V_co] = None, **kwargs*)

Bases: *aries_cloudagent.messaging.responder.BaseResponder*

Handle outgoing messages from message handlers.

create_outbound(*message: Union[aries_cloudagent.messaging.agent_message.AgentMessage, str, bytes], **kwargs*) → aries_cloudagent.transport.outbound.message.OutboundMessage

Create an OutboundMessage from a message body.

Parameters **message** – The message payload

send_outbound(*message: aries_cloudagent.transport.outbound.message.OutboundMessage*)

Send outbound message.

Parameters **message** – The *OutboundMessage* to be sent

send_webhook(*topic: str, payload: dict*)

Dispatch a webhook.

Parameters

- **topic** – the webhook topic identifier
- **payload** – the webhook payload value

aries_cloudagent.core.error module

Common exception classes.

exception aries_cloudagent.core.error.**BaseError**(**args, error_code: str = None, **kwargs*)

Bases: Exception

Generic exception class which other exceptions should inherit from.

error_code = None

message
Accessor for the error message.

exception aries_cloudagent.core.error.**ProtocolDefinitionValidationError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Error raised when there is a problem validating a protocol definition.

exception aries_cloudagent.core.error.**ProtocolMinorVersionNotSupported**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Minimum minor version protocol error.

Error raised when protocol support exists but minimum minor version is higher than in @type parameter.

exception aries_cloudagent.core.error.**StartupError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Error raised when there is a problem starting the conductor.

aries_cloudagent.core.plugin_registry module

Handle registration of plugin modules for extending functionality.

class aries_cloudagent.core.plugin_registry.**PluginRegistry**

Bases: object

Plugin registry for indexing application plugins.

init_context(*context: aries_cloudagent.config.injection_context.InjectionContext*)
Call plugin setup methods on the current context.

load_protocol_version(*context: aries_cloudagent.config.injection_context.InjectionContext, mod: module, version_definition: dict = None*)
Load a particular protocol version.

load_protocols(*context: aries_cloudagent.config.injection_context.InjectionContext, plugin: module*)
For modules that don't implement setup, register protocols manually.

plugin_names
Accessor for a list of all plugin modules.

plugins
Accessor for a list of all plugin modules.

register_admin_routes(*app*)
Call route registration methods on the current context.

register_package(*package_name: str*) → Sequence[module]
Register all modules (sub-packages) under a given package name.

register_plugin(*module_name: str*) → module
Register a plugin module.

validate_version(*version_list*, *module_name*)
Validate version dict format.

aries_cloudagent.core.protocol_registry module

Handle registration and publication of supported protocols.

class aries_cloudagent.core.protocol_registry.**ProtocolRegistry**
Bases: object

Protocol registry for indexing message families.

controllers
Accessor for a list of all protocol controller functions.

message_types
Accessor for a list of all message types.

parse_type_string(*message_type*)
Parse message type string and return dict with info.

prepare_disclosed(*context: aries_cloudagent.config.injection_context.InjectionContext, protocols: Sequence[str]*)
Call controllers and return publicly supported message families and roles.

protocols
Accessor for a list of all message protocols.

protocols_matching_query(*query: str*) → Sequence[str]
Return a list of message protocols matching a query string.

register_controllers(**controller_sets*, *version_definition=None*)
Add new controllers.

Parameters **controller_sets** – Mappings of message families to coroutines

register_message_types(**typesets*, *version_definition=None*)
Add new supported message types.

Parameters

- **typesets** – Mappings of message types to register
- **version_definition** – Optional version definition dict

resolve_message_class(*message_type: str*) → type
Resolve a message_type to a message class.

Given a message type identifier, this method returns the corresponding registered message class.

Parameters **message_type** – Message type to resolve

Returns The resolved message class

1.1.6 aries_cloudagent.holder package

Submodules

aries_cloudagent.holder.base module

Base holder class.

class aries_cloudagent.holder.base.**BaseHolder**

Bases: abc.ABC

Base class for holder.

create_credential_request(*credential_offer: dict*, *credential_definition: dict*, *holder_did: str*) → Tuple[str, str]

Create a credential request for the given credential offer.

Parameters

- **credential_offer** – The credential offer to create request for
- **credential_definition** – The credential definition to create an offer for
- **holder_did** – the DID of the agent making the request

Returns A tuple of the credential request and credential request metadata

create_presentation(*presentation_request: dict*, *requested_credentials: dict*, *schemas: dict*, *credential_definitions: dict*, *rev_states: dict = None*) → str

Get credentials stored in the wallet.

Parameters

- **presentation_request** – Valid indy format presentation request
- **requested_credentials** – Indy format requested credentials
- **schemas** – Indy formatted schemas JSON
- **credential_definitions** – Indy formatted credential definitions JSON
- **rev_states** – Indy format revocation states JSON

create_revocation_state(*cred_rev_id: str*, *rev_reg_def: dict*, *rev_reg_delta: dict*, *timestamp: int*, *tails_file_path: str*) → str

Create current revocation state for a received credential.

Parameters

- **cred_rev_id** – credential revocation id in revocation registry
- **rev_reg_def** – revocation registry definition
- **rev_reg_delta** – revocation delta
- **timestamp** – delta timestamp

Returns the revocation state

delete_credential(*credential_id: str*)

Remove a credential stored in the wallet.

Parameters **credential_id** – Credential id to remove

get_credential(*credential_id: str*) → str

Get a credential stored in the wallet.

Parameters **credential_id** – Credential id to retrieve

get_mime_type(*credential_id: str*, *attr: str = None*) → Union[dict, str]

Get MIME type per attribute (or for all attributes).

Parameters

- **credential_id** – credential id
- **attr** – attribute of interest or omit for all

Returns: Attribute MIME type or dict mapping attribute names to MIME types attr_meta_json = all_meta.tags.get(attr)

store_credential(*credential_definition: dict*, *credential_data: dict*, *credential_request_metadata: dict*, *credential_attr_mime_types=None*, *credential_id: str = None*, *rev_reg_def: dict = None*)

Store a credential in the wallet.

Parameters

- **credential_definition** – Credential definition for this credential
- **credential_data** – Credential data generated by the issuer
- **credential_request_metadata** – credential request metadata generated by the issuer
- **credential_attr_mime_types** – dict mapping attribute names to (optional) MIME types to store as non-secret record, if specified
- **credential_id** – optionally override the stored credential id
- **rev_reg_def** – revocation registry definition in json

Returns the ID of the stored credential

exception aries_cloudagent.holder.base.**HolderError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base class for holder exceptions.

aries_cloudagent.holder.indy module

Indy holder implementation.

class aries_cloudagent.holder.indy.**IndyHolder**(*wallet*)

Bases: *aries_cloudagent.holder.base.BaseHolder*

Indy holder class.

RECORD_TYPE_MIME_TYPES = 'attribute-mime-types'

create_credential_request(*credential_offer: dict*, *credential_definition: dict*, *holder_did: str*) → Tuple[str, str]

Create a credential request for the given credential offer.

Parameters

- **credential_offer** – The credential offer to create request for
- **credential_definition** – The credential definition to create an offer for
- **holder_did** – the DID of the agent making the request

Returns A tuple of the credential request and credential request metadata

create_presentation(*presentation_request: dict*, *requested_credentials: dict*, *schemas: dict*, *credential_definitions: dict*, *rev_states: dict = None*) → str

Get credentials stored in the wallet.

Parameters

- **presentation_request** – Valid indy format presentation request
- **requested_credentials** – Indy format requested credentials
- **schemas** – Indy formatted schemas JSON
- **credential_definitions** – Indy formatted credential definitions JSON
- **rev_states** – Indy format revocation states JSON

create_revocation_state(*cred_rev_id: str*, *rev_reg_def: dict*, *rev_reg_delta: dict*, *timestamp: int*, *tails_file_path: str*) → str

Create current revocation state for a received credential.

Parameters

- **cred_rev_id** – credential revocation id in revocation registry
- **rev_reg_def** – revocation registry definition
- **rev_reg_delta** – revocation delta
- **timestamp** – delta timestamp

Returns the revocation state

delete_credential(*credential_id: str*)

Remove a credential stored in the wallet.

Parameters **credential_id** – Credential id to remove

get_credential(*credential_id: str*) → str

Get a credential stored in the wallet.

Parameters **credential_id** – Credential id to retrieve

get_credentials(*start: int*, *count: int*, *wql: dict*)

Get credentials stored in the wallet.

Parameters

- **start** – Starting index
- **count** – Number of records to return
- **wql** – wql query dict

get_credentials_for_presentation_request_by_referent(*presentation_request: dict, referents: Sequence[str], start: int, count: int, extra_query: dict = {}*)

Get credentials stored in the wallet.

Parameters

- **presentation_request** – Valid presentation request from issuer
- **referents** – Presentation request referents to use to search for creds

- **start** – Starting index
- **count** – Maximum number of records to return
- **extra_query** – wql query dict

get_mime_type(*credential_id: str*, *attr: str = None*) → Union[dict, str]

Get MIME type per attribute (or for all attributes).

Parameters

- **credential_id** – credential id
- **attr** – attribute of interest or omit for all

Returns: Attribute MIME type or dict mapping attribute names to MIME types attr_meta_json = all_meta.tags.get(attr)

store_credential(*credential_definition: dict*, *credential_data: dict*, *credential_request_metadata: dict*, *credential_attr_mime_types=None*, *credential_id: str = None*, *rev_reg_def: dict = None*) → str

Store a credential in the wallet.

Parameters

- **credential_definition** – Credential definition for this credential
- **credential_data** – Credential data generated by the issuer
- **credential_request_metadata** – credential request metadata generated by the issuer
- **credential_attr_mime_types** – dict mapping attribute names to (optional) MIME types to store as non-secret record, if specified
- **credential_id** – optionally override the stored credential id
- **rev_reg_def** – revocation registry definition in json

Returns the ID of the stored credential

1.1.7 aries_cloudagent.issuer package

Submodules

aries_cloudagent.issuer.base module

Ledger issuer class.

class aries_cloudagent.issuer.base.**BaseIssuer**

Bases: abc.ABC

Base class for issuer.

create_and_store_credential_definition(*origin_did: str*, *schema: dict*, *signature_type: str = None*, *tag: str = None*, *support_revocation: bool = False*) → Tuple[str, str]

Create a new credential definition and store it in the wallet.

Parameters

- **origin_did** – the DID issuing the credential definition

- **schema_json** – the schema used as a basis
- **signature_type** – the credential definition signature type (default 'CL')
- **tag** – the credential definition tag
- **support_revocation** – whether to enable revocation for this credential def

Returns A tuple of the credential definition ID and JSON

create_and_store_revocation_registry(*origin_did: str*, *cred_def_id: str*, *revoc_def_type: str*, *tag: str*, *max_cred_num: int*, *tails_base_path: str*, *issuance_type: str = None*) → Tuple[str, str, str]

Create a new revocation registry and store it in the wallet.

Parameters

- **origin_did** – the DID issuing the revocation registry
- **cred_def_id** – the identifier of the related credential definition
- **revoc_def_type** – the revocation registry type (default CL_ACCUM)
- **tag** – the unique revocation registry tag
- **max_cred_num** – the number of credentials supported in the registry
- **tails_base_path** – where to store the tails file
- **issuance_type** – optionally override the issuance type

Returns A tuple of the revocation registry ID, JSON, and entry JSON

create_and_store_schema(*origin_did: str, schema_name: str, schema_version: str, attribute_names: Sequence[str]*) → Tuple[str, str]

Create a new credential schema and store it in the wallet.

Parameters

- **origin_did** – the DID issuing the credential definition
- **schema_name** – the schema name
- **schema_version** – the schema version
- **attribute_names** – a sequence of schema attribute names

Returns A tuple of the schema ID and JSON

create_credential(*schema: dict*, *credential_offer: dict*, *credential_request: dict*, *credential_values: dict*, *revoc_reg_id: str = None*, *tails_reader_handle: int = None*) → Tuple[str, str]

Create a credential.

Args schema: Schema to create credential for credential_offer: Credential Offer to create credential for credential_request: Credential request to create credential for credential_values: Values to go in credential revoc_reg_id: ID of the revocation registry tails_reader_handle: Handle for the tails file blob reader

Returns A tuple of created credential and revocation id

create_credential_offer(*credential_definition_id*) → str

Create a credential offer for the given credential definition id.

Parameters **credential_definition_id** – The credential definition to create an offer for

Returns The created credential offer

credential_definition_in_wallet(*credential_definition_id: str*) → bool

Check whether a given credential definition ID is present in the wallet.

Parameters **credential_definition_id** – The credential definition ID to check

make_credential_definition_id(*origin_did: str*, *schema: dict*, *signature_type: str = None*, *tag: str = None*) → str

Derive the ID for a credential definition.

make_schema_id(*origin_did: str*, *schema_name: str*, *schema_version: str*) → str

Derive the ID for a schema.

merge_revocation_registry_deltas(*fro_delta: str*, *to_delta: str*) → str

Merge revocation registry deltas.

Parameters

- **fro_delta** – original delta in JSON format
- **to_delta** – incoming delta in JSON format

Returns Merged delta in JSON format

revoke_credentials(*revoc_reg_id: str, tails_file_path: str, cred_revoc_ids: Sequence[str]*) → str

Revoke a set of credentials in a revocation registry.

Parameters

- **revoc_reg_id** – ID of the revocation registry
- **tails_file_path** – path to the local tails file
- **cred_revoc_ids** – sequences of credential indexes in the revocation registry

Returns the combined revocation delta

exception aries_cloudagent.issuer.base.**IssuerError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Generic issuer error.

exception aries_cloudagent.issuer.base.**IssuerRevocationRegistryFullError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.issuer.base.IssuerError*

Revocation registry is full when issuing a new credential.

aries_cloudagent.issuer.indy module

Indy issuer implementation.

class aries_cloudagent.issuer.indy.**IndyIssuer**(*wallet*)

Bases: *aries_cloudagent.issuer.base.BaseIssuer*

Indy issuer class.

create_and_store_credential_definition(*origin_did: str, schema: dict, signature_type: str = None, tag: str = None, support_revocation: bool = False*) → Tuple[str, str]

Create a new credential definition and store it in the wallet.

Parameters

- **origin_did** – the DID issuing the credential definition
- **schema** – the schema used as a basis
- **signature_type** – the credential definition signature type (default 'CL')
- **tag** – the credential definition tag
- **support_revocation** – whether to enable revocation for this credential def

Returns A tuple of the credential definition ID and JSON

create_and_store_revocation_registry(*origin_did: str, cred_def_id: str, revoc_def_type: str, tag: str, max_cred_num: int, tails_base_path: str, issuance_type: str = None*) → Tuple[str, str, str]

Create a new revocation registry and store it in the wallet.

Parameters

- **origin_did** – the DID issuing the revocation registry
- **cred_def_id** – the identifier of the related credential definition
- **revoc_def_type** – the revocation registry type (default CL_ACCUM)
- **tag** – the unique revocation registry tag
- **max_cred_num** – the number of credentials supported in the registry
- **tails_base_path** – where to store the tails file
- **issuance_type** – optionally override the issuance type

Returns A tuple of the revocation registry ID, JSON, and entry JSON

create_and_store_schema(*origin_did: str, schema_name: str, schema_version: str, attribute_names: Sequence[str]*) → Tuple[str, str]

Create a new credential schema and store it in the wallet.

Parameters

- **origin_did** – the DID issuing the credential definition
- **schema_name** – the schema name
- **schema_version** – the schema version
- **attribute_names** – a sequence of schema attribute names

Returns A tuple of the schema ID and JSON

create_credential(*schema: dict, credential_offer: dict, credential_request: dict, credential_values: dict, revoc_reg_id: str = None, tails_file_path: str = None*) → Tuple[str, str]

Create a credential.

Args schema: Schema to create credential for credential_offer: Credential Offer to create credential for credential_request: Credential request to create credential for credential_values: Values to go in credential revoc_reg_id: ID of the revocation registry tails_file_path: Path to the local tails file

Returns A tuple of created credential and revocation id

create_credential_offer(*credential_definition_id: str*) → str

Create a credential offer for the given credential definition id.

Parameters **credential_definition_id** – The credential definition to create an offer for

Returns The created credential offer

credential_definition_in_wallet(*credential_definition_id: str*) → bool

Check whether a given credential definition ID is present in the wallet.

Parameters **credential_definition_id** – The credential definition ID to check

make_credential_definition_id(*origin_did: str, schema: dict, signature_type: str = None, tag: str = None*) → str

Derive the ID for a credential definition.

make_schema_id(*origin_did: str, schema_name: str, schema_version: str*) → str

Derive the ID for a schema.

merge_revocation_registry_deltas(*fro_delta: str, to_delta: str*) → str

Merge revocation registry deltas.

Parameters

- **fro_delta** – original delta in JSON format
- **to_delta** – incoming delta in JSON format

Returns Merged delta in JSON format

revoke_credentials(*revoc_reg_id: str, tails_file_path: str, cred_revoc_ids: Sequence[str]*) → str

Revoke a set of credentials in a revocation registry.

Parameters

- **revoc_reg_id** – ID of the revocation registry
- **tails_file_path** – path to the local tails file
- **cred_revoc_ids** – sequences of credential indexes in the revocation registry

Returns the combined revocation delta

aries_cloudagent.issuer.util module

1.1.8 aries_cloudagent.ledger package

Submodules

aries_cloudagent.ledger.base module

Ledger base class.

class aries_cloudagent.ledger.base.**BaseLedger**

Bases: abc.ABC

Base class for ledger.

LEDGER_TYPE = None

accept_txn_author_agreement(*taa_record: dict*, *mechanism: str*, *accept_time: int = None*)

Save a new record recording the acceptance of the TAA.

create_and_send_credential_definition(*issuer: aries_cloudagent.issuer.base.BaseIssuer*, *schema_id: str*, *signature_type: str = None*, *tag: str = None*, *support_revocation: bool = False*) → Tuple[str, dict]

Send credential definition to ledger and store relevant key matter in wallet.

Parameters

- **issuer** – The issuer instance to use for credential definition creation
- **schema_id** – The schema id of the schema to create cred def for
- **signature_type** – The signature type to use on the credential definition
- **tag** – Optional tag to distinguish multiple credential definitions
- **support_revocation** – Optional flag to enable revocation for this cred def

create_and_send_schema(*issuer: aries_cloudagent.issuer.base.BaseIssuer, schema_name: str, schema_version: str, attribute_names: Sequence[str]*) → Tuple[str, dict]

Send schema to ledger.

Parameters

- **issuer** – The issuer instance to use for schema creation
- **schema_name** – The schema name
- **schema_version** – The schema version
- **attribute_names** – A list of schema attributes

did_to_nym(*did: str*) → str

Remove the ledger's DID prefix to produce a nym.

fetch_txn_author_agreement()

Fetch the current AML and TAA from the ledger.

get_credential_definition(*credential_definition_id: str*) → dict

Get a credential definition from the cache if available, otherwise the ledger.

Parameters **credential_definition_id** – The schema id of the schema to fetch cred def for

get_endpoint_for_did(*did: str*) → str

Fetch the endpoint for a ledger DID.

Parameters **did** – The DID to look up on the ledger or in the cache

get_key_for_did(*did: str*) → str

Fetch the verkey for a ledger DID.

Parameters **did** – The DID to look up on the ledger or in the cache

get_latest_txn_author_acceptance()
Look up the latest TAA acceptance.

get_revoc_reg_def(*revoc_reg_id: str*) → dict
Look up a revocation registry definition by ID.

get_revoc_reg_delta(*revoc_reg_id: str, timestamp_from=0, timestamp_to=None) -> (<class 'dict'>, <class 'int'>*)
Look up a revocation registry delta by ID.

get_revoc_reg_entry(*revoc_reg_id: str, timestamp: int*)
Get revocation registry entry by revocation registry ID and timestamp.

get_schema(*schema_id: str*) → dict
Get a schema from the cache if available, otherwise fetch from the ledger.

Parameters **schema_id** – The schema id (or stringified sequence number) to retrieve

get_txn_author_agreement(*reload: bool = False*)
Get the current transaction author agreement, fetching it if necessary.

nym_to_did(*nym: str*) → str
Format a nym with the ledger's DID prefix.

register_nym(*did: str, verkey: str, alias: str = None, role: str = None*)
Register a nym on the ledger.

Parameters

- **did** – DID to register on the ledger.
- **verkey** – The verification key of the keypair.
- **alias** – Human-friendly alias to assign to the DID.
- **role** – For permissioned ledgers, what role should the new DID have.

send_revoc_reg_def(*revoc_reg_def: dict, issuer_did: str = None*)
Publish a revocation registry definition to the ledger.

send_revoc_reg_entry(*revoc_reg_id: str, revoc_def_type: str, revoc_reg_entry: dict, issuer_did: str = None*)
Publish a revocation registry entry to the ledger.

taa_digest(*version: str, text: str*)
Generate the digest of a TAA record.

update_endpoint_for_did(*did: str, endpoint: str*) → bool
Check and update the endpoint on the ledger.

Parameters

- **did** – The ledger DID
- **endpoint** – The endpoint address

aries_cloudagent.ledger.error module

Ledger related errors.

exception aries_cloudagent.ledger.error.**BadLedgerRequestError**(**args, error_code: str = None, **kwargs*)

Bases: aries_cloudagent.ledger.error.LedgerError

The current request cannot proceed.

exception aries_cloudagent.ledger.error.**ClosedPoolError**(**args, error_code: str = None, **kwargs*)

Bases: aries_cloudagent.ledger.error.LedgerError

Indy pool is closed.

exception aries_cloudagent.ledger.error.**LedgerConfigError**(**args, error_code: str = None, **kwargs*)

Bases: aries_cloudagent.ledger.error.LedgerError

Base class for ledger configuration errors.

exception aries_cloudagent.ledger.error.**LedgerError**(**args, error_code: str = None, **kwargs*)

Bases: aries_cloudagent.core.error.BaseError

Base class for ledger errors.

exception aries_cloudagent.ledger.error.**LedgerTransactionError**(**args, error_code: str = None, **kwargs*)

Bases: aries_cloudagent.ledger.error.LedgerError

The ledger rejected the transaction.

aries_cloudagent.ledger.indy module

Indy ledger implementation.

class aries_cloudagent.ledger.indy.**IndyLedger**(*pool_name: str, wallet: aries_cloudagent.wallet.base.BaseWallet, *, keepalive: int = 0, cache: aries_cloudagent.cache.base.BaseCache = None, cache_duration: int = 600, read_only: bool = False*)

Bases: aries_cloudagent.ledger.base.BaseLedger

Indy ledger class.

LEDGER_TYPE = 'indy'

accept_txn_author_agreement(*taa_record: dict, mechanism: str, accept_time: int = None*)

Save a new record recording the acceptance of the TAA.

check_existing_schema(*public_did: str, schema_name: str, schema_version: str, attribute_names: Sequence[str]*) → Tuple[str, dict]

Check if a schema has already been published.

check_pool_config() → bool

Check if a pool config has been created.

close()

Close the pool ledger.

create_and_send_credential_definition(*issuer: aries_cloudagent.issuer.base.BaseIssuer, schema_id: str, signature_type: str = None, tag: str = None, support_revocation: bool = False*) → Tuple[str, dict]

Send credential definition to ledger and store relevant key matter in wallet.

Parameters

- **issuer** – The issuer instance to use for credential definition creation
- **schema_id** – The schema id of the schema to create cred def for
- **signature_type** – The signature type to use on the credential definition
- **tag** – Optional tag to distinguish multiple credential definitions
- **support_revocation** – Optional flag to enable revocation for this cred def

create_and_send_schema(*issuer: aries_cloudagent.issuer.base.BaseIssuer, schema_name: str, schema_version: str, attribute_names: Sequence[str]*) → Tuple[str, dict]

Send schema to ledger.

Parameters

- **issuer** – The issuer instance creating the schema
- **schema_name** – The schema name
- **schema_version** – The schema version
- **attribute_names** – A list of schema attributes

create_pool_config(*genesis_transactions: str, recreate: bool = False*)

Create the pool ledger configuration.

credential_definition_id2schema_id(*credential_definition_id*)

From a credential definition, get the identifier for its schema.

Parameters **credential_definition_id** – The identifier of the credential definition from which to identify a schema

fetch_credential_definition(*credential_definition_id: str*) → dict

Get a credential definition from the ledger by id.

Parameters **credential_definition_id** – The cred def id of the cred def to fetch

fetch_schema_by_id(*schema_id: str*) → dict

Get schema from ledger.

Parameters **schema_id** – The schema id (or stringified sequence number) to retrieve

Returns Indy schema dict

fetch_schema_by_seq_no(*seq_no: int*)

Fetch a schema by its sequence number.

Parameters **seq_no** – schema ledger sequence number

Returns Indy schema dict

fetch_txn_author_agreement() → dict

Fetch the current AML and TAA from the ledger.

get_credential_definition(*credential_definition_id: str*) → dict

Get a credential definition from the cache if available, otherwise the ledger.

Parameters **credential_definition_id** – The schema id of the schema to fetch cred def for

get_endpoint_for_did(*did: str*) → str
Fetch the endpoint for a ledger DID.

Parameters **did** – The DID to look up on the ledger or in the cache

get_indy_storage() → aries_cloudagent.storage.indy.IndyStorage
Get an IndyStorage instance for the current wallet.

get_key_for_did(*did: str*) → str
Fetch the verkey for a ledger DID.

Parameters **did** – The DID to look up on the ledger or in the cache

get_latest_txn_author_acceptance() → dict
Look up the latest TAA acceptance.

get_revoc_reg_def(*revoc_reg_id: str*) → dict
Get revocation registry definition by ID.

get_revoc_reg_delta(*revoc_reg_id: str, timestamp_from=0, timestamp_to=None) -> (<class 'dict'>, <class 'int'>*)
Look up a revocation registry delta by ID.

:param revoc_reg_id revocation registry id :param timestamp_from from time. a total number of seconds from Unix Epoch :param timestamp_to to time. a total number of seconds from Unix Epoch

:returns delta response, delta timestamp

get_revoc_reg_entry(*revoc_reg_id: str, timestamp: int*)
Get revocation registry entry by revocation registry ID and timestamp.

get_schema(*schema_id: str*) → dict
Get a schema from the cache if available, otherwise fetch from the ledger.

Parameters **schema_id** – The schema id (or stringified sequence number) to retrieve

get_txn_author_agreement(*reload: bool = False*) → dict
Get the current transaction author agreement, fetching it if necessary.

nym_to_did(*nym: str*) → str
Format a nym with the ledger's DID prefix.

open()
Open the pool ledger, creating it if necessary.

register_nym(*did: str, verkey: str, alias: str = None, role: str = None*)
Register a nym on the ledger.

Parameters

- **did** – DID to register on the ledger.
- **verkey** – The verification key of the keypair.
- **alias** – Human-friendly alias to assign to the DID.
- **role** – For permissioned ledgers, what role should the new DID have.

send_revoc_reg_def(*revoc_reg_def: dict, issuer_did: str = None*)
Publish a revocation registry definition to the ledger.

send_revoc_reg_entry(*revoc_reg_id: str*, *revoc_def_type: str*, *revoc_reg_entry: dict*, *issuer_did: str = None*)
Publish a revocation registry entry to the ledger.

taa_digest(*version: str*, *text: str*)
Generate the digest of a TAA record.

taa_rough_timestamp() → int
Get a timestamp accurate to the day.

Anything more accurate is a privacy concern.

update_endpoint_for_did(*did: str*, *endpoint: str*) → bool
Check and update the endpoint on the ledger.

Parameters

- **did** – The ledger DID
- **endpoint** – The endpoint address

aries_cloudagent.ledger.provider module

Default ledger provider classes.

class aries_cloudagent.ledger.provider.**LedgerProvider**
Bases: *aries_cloudagent.config.base.BaseProvider*

Provider for the default ledger implementation.

LEDGER_CLASSES = {'indy': 'aries_cloudagent.ledger.indy.IndyLedger'}

provide(*settings: aries_cloudagent.config.base.BaseSettings*, *injector: aries_cloudagent.config.base.BaseInjector*)
Create and open the ledger instance.

aries_cloudagent.ledger.routes module

Ledger admin routes.

class aries_cloudagent.ledger.routes.**AMLRecordSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)
Bases: marshmallow.schema.Schema

Ledger AML record.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.ledger.routes.**TAAAcceptSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)
Bases: marshmallow.schema.Schema

Input schema for accepting the TAA.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.ledger.routes.**TAAAcceptanceSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

TAA acceptance record.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.ledger.routes.**TAAInfoSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Transaction author agreement info.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.ledger.routes.**TAARecordSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Ledger TAA record.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.ledger.routes.**TAAResultSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Result schema for a transaction author agreement.

opts = <marshmallow.schema.SchemaOpts object>

aries_cloudagent.ledger.routes.**get_did_endpoint**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3e18278>*)

Request handler for getting a verkey for a DID from the ledger.

Parameters **request** – aiohttp request object

aries_cloudagent.ledger.routes.**get_did_verkey**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3e18278>*)

Request handler for getting a verkey for a DID from the ledger.

Parameters **request** – aiohttp request object

aries_cloudagent.ledger.routes.**ledger_accept_taa**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3e18278>*)

Request handler for accepting the current transaction author agreement.

Parameters **request** – aiohttp request object

Returns The DID list response

`aries_cloudagent.ledger.routes.`**`ledger_get_taa`**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3e18278>*)

Request handler for fetching the transaction author agreement.

Parameters **`request`** – aiohttp request object

Returns The TAA information including the AML

`aries_cloudagent.ledger.routes.`**`register`**(*app: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3e18278>*)

Register routes.

`aries_cloudagent.ledger.routes.`**`register_ledger_nym`**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3e18278>*)

Request handler for registering a NYM with the ledger.

Parameters **`request`** – aiohttp request object

aries_cloudagent.ledger.util module

Ledger utilities.

1.1.9 aries_cloudagent.messaging package

Subpackages

aries_cloudagent.messaging.ack package

Submodules

aries_cloudagent.messaging.ack.message module

Represents an explicit ack message as per Aries RFC 15.

class `aries_cloudagent.messaging.ack.message.`**`Ack`**(*status: str = None, **kwargs*)

Bases: `aries_cloudagent.messaging.agent_message.AgentMessage`

Base class representing an explicit ack message.

Subclass to adopt, specify Meta message type and handler class.

class Meta

Bases: `object`

Ack metadata.

`schema_class = 'AckSchema'`

class `aries_cloudagent.messaging.ack.message.`**`AckSchema`**(**args, **kwargs*)

Bases: `aries_cloudagent.messaging.agent_message.AgentMessageSchema`

Schema for Ack base class.

class Meta

Bases: `object`

Ack schema metadata.

model_class
alias of *Ack*

status = <fields.Constant(default='OK', attribute=None, validate=None, required=True,

aries_cloudagent.messaging.credential_definitions package

Submodules

aries_cloudagent.messaging.credential_definitions.routes module

Credential definition admin routes.

class aries_cloudagent.messaging.credential_definitions.routes.**CredentialDefinitionGetResul**

Bases: marshmallow.schema.Schema

Results schema for schema get request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.credential_definitions.routes.**CredentialDefinitionSchema**(

Bases: marshmallow.schema.Schema

Credential definition schema.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.credential_definitions.routes.**CredentialDefinitionSendRequ**

Bases: marshmallow.schema.Schema

Request schema for schema send request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.credential_definitions.routes.**CredentialDefinitionSendResu**

Bases: marshmallow.schema.Schema

Results schema for schema send request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.credential_definitions.routes.**CredentialDefinitionsCreated**

Bases: marshmallow.schema.Schema

Results schema for cred-defs-created request.

opts = <marshmallow.schema.SchemaOpts object>

`aries_cloudagent.messaging.credential_definitions.routes.`**`credential_definitions_created`**(*req <sp ob- ject at 0x7*

Request handler for retrieving credential definitions that current agent created.

Parameters **`request`** – aiohttp request object

Returns The identifiers of matching credential definitions.

`aries_cloudagent.messaging.credential_definitions.routes.`**`credential_definitions_get_credent`**

Request handler for getting a credential definition from the ledger.

Parameters **`request`** – aiohttp request object

Returns The credential definition details.

`aries_cloudagent.messaging.credential_definitions.routes.`**`credential_definitions_send_creden`**

Request handler for sending a credential definition to the ledger.

Parameters **`request`** – aiohttp request object

Returns The credential definition identifier

`aries_cloudagent.messaging.credential_definitions.routes.`**`register`**(*app: <sphinx.ext.autodoc.importer._Mock object at 0x7fcaf3568588>*)

Register routes.

aries_cloudagent.messaging.credential_definitions.util module

Credential definition utilities.

aries_cloudagent.messaging.decorators package

Submodules

aries_cloudagent.messaging.decorators.attach_decorator module

A message decorator for attachments.

An attach decorator embeds content or specifies appended content.

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecorator**(*, *ident: str = None, description: str = None, filename: str = None, mime_type: str = None, lastmod_time: str = None, byte_count: int = None, data: aries_cloudagent.m **kwargs*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing attach decorator.

class Meta

Bases: object

AttachDecorator metadata.

schema_class = 'AttachDecoratorSchema'

classmethod from_indy_dict(*indy_dict: dict*, *, *ident: str = None*, *description: str = None*, *filename: str = None*, *lastmod_time: str = None*, *byte_count: int = None*)

Create *AttachDecorator* instance from indy object (dict).

Given indy object (dict), JSON dump, base64-encode, and embed it as data; mark *application/json* MIME type.

Parameters

- **indy_dict** – indy (dict) data structure
- **ident** – optional attachment identifier (default random UUID4)
- **description** – optional attachment description
- **filename** – optional attachment filename

- **lastmod_time** – optional attachment last modification time
- **byte_count** – optional attachment byte count

indy_dict

Return indy data structure encoded in attachment.

Returns: dict with indy object in data attachment

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorData**(*, *jws_: aries_cloudag = None, sha256_: str = None, links_: Union[list, str] = None, base64_: str = None, json_: str = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Attach decorator data.

class Meta

Bases: object

AttachDecoratorData metadata.

schema_class = 'AttachDecoratorDataSchema'

base64

Accessor for base64 decorator data, or None.

header_map(*idx: int = 0, jose: bool = True*) → Mapping[KT, VT_co]

Accessor for header info at input index, default 0 or unique for singly-signed.

Parameters

- **idx** – index of interest, zero-based (default 0)
- **jose** – True to return unprotected header attributes, False for protected only

json

Accessor for json decorator data, or None.

jws

Accessor for JWS, or None.

links

Accessor for links decorator data, or None.

sha256
Accessor for sha256 decorator data, or None.

sign(*verkeys: Union[str, Sequence[str]], wallet: aries_cloudagent.wallet.base.BaseWallet*)
Sign base64 data value of attachment.

Parameters

- **verkeys** – verkey(s) of the signing party (in raw or DID key format)
- **wallet** – The wallet to use for the signature

signatures
Accessor for number of signatures.

signed
Accessor for signed content (payload), None for unsigned.

verify(*wallet: aries_cloudagent.wallet.base.BaseWallet*) → bool
Verify the signature(s).

Parameters **wallet** – Wallet to use to verify signature

Returns True if verification succeeds else False

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorData1JWS**(*, *header: aries_cl protected: str = None, signature: str*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Single Detached JSON Web Signature for inclusion in attach decorator data.

class Meta
Bases: object

AttachDecoratorData1JWS metadata.

schema_class = 'AttachDecoratorData1JWSSchema'

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorData1JWSSchema**

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Single attach decorator data JWS schema.

class Meta
Bases: object

Single attach decorator data JWS schema metadata.

model_class
alias of *AttachDecoratorData1JWS*

header = <fields.Nested(default=<marshmallow.missing>, attribute=None, validate=None,

protected = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<ar

signature = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<ar

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorDataJWS**(*, *header: aries_clo = None, protected: str = None, signature: str = None, signatures: Sequence[a = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Detached JSON Web Signature for inclusion in attach decorator data.

May hold one signature in flattened format, or multiple signatures in the "signatures" member.

class Meta

Bases: object

AttachDecoratorDataJWS metadata.

schema_class = 'AttachDecoratorDataJWSSchema'

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorDataJWSHeader**(*k s*

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Attach decorator data JWS header.

class Meta

Bases: object

AttachDecoratorDataJWS metadata.

schema_class = 'AttachDecoratorDataJWSHeaderSchema'

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorDataJWSHeaderS**

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Attach decorator data JWS header schema.

class Meta

Bases: object

Attach decorator data schema metadata.

model_class
alias of *AttachDecoratorDataJWSHeader*

kid = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<aries_cl

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorDataJWSSchema**(*

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Schema for detached JSON Web Signature for inclusion in attach decorator data.

class Meta
Bases: object

Metadata for schema for detached JWS for inclusion in attach deco data.

model_class
alias of *AttachDecoratorDataJWS*

header = <fields.Nested(default=<marshmallow.missing>, attribute=None, validate=None,

protected = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<ar

signature = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<ar

signatures = <fields.List(default=<marshmallow.missing>, attribute=None, validate=None

validate_single_xor_multi_sig(*data: Mapping[KT, VT_co], **kwargs*)
Ensure model is for either 1 or many sigatures, not mishmash of both.

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorDataSchema**(**arg*
***kw*

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Attach decorator data schema.

class Meta
Bases: object

Attach decorator data schema metadata.

model_class
alias of *AttachDecoratorData*

base64_ = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<arie

json_ = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, r

jws_ = <fields.Nested(default=<marshmallow.missing>, attribute=None, validate=None, re

links_ = <fields.List(default=<marshmallow.missing>, attribute=None, validate=None, re

sha256_ = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<arie

validate_data_spec(*data: Mapping[KT, VT_co], **kwargs*)
Ensure model chooses exactly one of base64, json, or links.

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorSchema**(**args*, ***kwargs*)

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Attach decorator schema used in serialization/deserialization.

class Meta
Bases: object

AttachDecoratorSchema metadata.

model_class
alias of *AttachDecorator*

byte_count = <fields.Integer(default=<marshmallow.missing>, attribute=None, validate=N

data = <fields.Nested(default=<marshmallow.missing>, attribute=None, validate=None, re

description = <fields.String(default=<marshmallow.missing>, attribute=None, validate=N

filename = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None

ident = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, r

lastmod_time = <fields.String(default=<marshmallow.missing>, attribute=None, validate=

mime_type = <fields.String(default=<marshmallow.missing>, attribute=None, validate=Non

aries_cloudagent.messaging.decorators.attach_decorator.**did_key**(*verkey: str*) → str

Qualify verkey into DID key if need be.

aries_cloudagent.messaging.decorators.attach_decorator.**raw_key**(*verkey: str*) → str

Strip qualified key to raw key if need be.

aries_cloudagent.messaging.decorators.base module

Classes for managing a collection of decorators.

class aries_cloudagent.messaging.decorators.base.**BaseDecoratorSet**(*models: dict = None*)

Bases: collections.OrderedDict

Collection of decorators.

add_model(*key: str, model: Type[aries_cloudagent.messaging.models.base.BaseModel]*)
Add a registered decorator model.

copy() → aries_cloudagent.messaging.decorators.base.BaseDecoratorSet
Return a copy of the decorator set.

extract_decorators(*message: Mapping[KT, VT_co], schema: Type[<sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf44a2438>] = None, serialized: bool = True, skip_attrs: Sequence[str] = None*) → collections.OrderedDict
Extract decorators and return the remaining properties.

field(*name: str*) → aries_cloudagent.messaging.decorators.base.BaseDecoratorSet
Access a named decorated field.

fields
Acessor for the set of currently defined fields.

has_field(*name: str*) → bool
Check for the existence of a named decorator field.

load_decorator(*key: str, value, serialized=False*)
Convert a decorator value to its loaded representation.

models
Accessor for the models dictionary.

prefix
Accessor for the decorator prefix.

remove_field(*name: str*)
Remove a named decorated field.

remove_model(*key: str*)
Remove a registered decorator model.

to_dict(*prefix: str = None*) → collections.OrderedDict
Convert to a dictionary (serialize).

Raises `BaseModelError` – on decorator validation errors

exception `aries_cloudagent.messaging.decorators.base.`**DecoratorError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base error for decorator issues.

aries_cloudagent.messaging.decorators.default module

Default decorator set implementation.

class `aries_cloudagent.messaging.decorators.default.`**DecoratorSet**(*models: dict = None*)

Bases: *aries_cloudagent.messaging.decorators.base.BaseDecoratorSet*

Default decorator set implementation.

aries_cloudagent.messaging.decorators.localization_decorator module

The localization decorator (~l10n) for message localization information.

class aries_cloudagent.messaging.decorators.localization_decorator.**LocalizationDecorator**(*, *lo ca st = N lo ca iz- ab Se qu = N ca a- lo Se qu = N*

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing the localization decorator.

class Meta

Bases: object

LocalizationDecorator metadata.

schema_class = 'LocalizationDecoratorSchema'

class aries_cloudagent.messaging.decorators.localization_decorator.**LocalizationDecoratorSc**

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Localization decorator schema used in serialization/deserialization.

class Meta

Bases: object

LocalizationDecoratorSchema metadata.

model_class

alias of *LocalizationDecorator*

catalogs

Used by autodoc_mock_imports.

locale

Used by autodoc_mock_imports.

localizable

Used by autodoc_mock_imports.

aries_cloudagent.messaging.decorators.please_ack_decorator module

The please-ack decorator to request acknowledgement.

class aries_cloudagent.messaging.decorators.please_ack_decorator.**PleaseAckDecorator**(*message_str = None, on: Sequence[st = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing the please-ack decorator.

class Meta

Bases: object

PleaseAckDecorator metadata.

schema_class = 'PleaseAckDecoratorSchema'

class aries_cloudagent.messaging.decorators.please_ack_decorator.**PleaseAckDecoratorSchema**(

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

PleaseAck decorator schema used in serialization/deserialization.

class Meta

Bases: object

PleaseAckDecoratorSchema metadata.

model_class

alias of *PleaseAckDecorator*

message_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=No

on = <fields.List(default=<marshmallow.missing>, attribute=None, validate=None, requir

aries_cloudagent.messaging.decorators.signature_decorator module

Model and schema for working with field signatures within message bodies.

class aries_cloudagent.messaging.decorators.signature_decorator.**SignatureDecorator**(*, *signature_type: str = None*, *signature: str = None*, *sig_data: str = None*, *signer: str = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing a field value signed by a known verkey.

class Meta

Bases: object

SignatureDecorator metadata.

schema_class = 'SignatureDecoratorSchema'

TYPE_ED25519SHA512 = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/signature/1.0/ed25519Sha512_

classmethod create(*value*, *signer: str*, *wallet: aries_cloudagent.wallet.base.BaseWallet*, *timestamp=None*) → aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator

Create a Signature.

Sign a field value and return a newly constructed *SignatureDecorator* representing the resulting signature.

Parameters

- **value** – Value to sign
- **signer** – Verkey of the signing party
- **wallet** – The wallet to use for the signature

Returns The created *SignatureDecorator* object

decode() -> (<class 'object'>, <class 'int'>)

Decode the signature to its timestamp and value.

Returns A tuple of (decoded message, timestamp)

verify(*wallet: aries_cloudagent.wallet.base.BaseWallet*) → bool

Verify the signature against the signer's public key.

Parameters **wallet** – Wallet to use to verify signature

Returns True if verification succeeds else False

class aries_cloudagent.messaging.decorators.signature_decorator.**SignatureDecoratorSchema**(*a
**

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

SignatureDecorator schema.

class Meta

Bases: object

SignatureDecoratorSchema metadata.

model_class

alias of *SignatureDecorator*

sig_data

Used by autodoc_mock_imports.

signature

Used by autodoc_mock_imports.

signature_type

Used by autodoc_mock_imports.

signer

Used by autodoc_mock_imports.

aries_cloudagent.messaging.decorators.thread_decorator module

A message decorator for threads.

A thread decorator identifies a message that may require additional context from previous messages.

class aries_cloudagent.messaging.decorators.thread_decorator.**ThreadDecorator**(*, *thid: str = None, pthid: str = None, sender_order: int = None, received_orders: Mapping[KT, VT_co] = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing thread decorator.

class Meta

Bases: object

ThreadDecorator metadata.

schema_class = 'ThreadDecoratorSchema'

pthid
Accessor for parent thread identifier.

Returns This thread's *pthid*

received_orders
Get received orders.

Returns The highest sender_order value that the sender has seen from other sender(s) on the thread.

sender_order
Get sender order.

Returns A number that tells where this message fits in the sequence of all messages that the current sender has contributed to this thread

thid
Accessor for thread identifier.

Returns This thread's *thid*

class aries_cloudagent.messaging.decorators.thread_decorator.**ThreadDecoratorSchema**(**args*, ***kwargs*)

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Thread decorator schema used in serialization/deserialization.

class Meta
Bases: object

ThreadDecoratorSchema metadata.

model_class
alias of *ThreadDecorator*

pthid
Used by autodoc_mock_imports.

received_orders
Used by autodoc_mock_imports.

sender_order
Used by autodoc_mock_imports.

thid
Used by autodoc_mock_imports.

aries_cloudagent.messaging.decorators.timing_decorator module

The timing decorator (~timing).

This decorator allows the timing of agent messages to be communicated and constrained.

class aries_cloudagent.messaging.decorators.timing_decorator.**TimingDecorator**(*, *in_time: Union[str, datetime.datetime] = None*, *out_time: Union[str, datetime.datetime] = None*, *stale_time: Union[str, datetime.datetime] = None*, *expires_time: Union[str, datetime.datetime] = None*, *delay_milli: int = None*, *wait_until_time: Union[str, datetime.datetime] = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing the timing decorator.

class Meta

Bases: object

TimingDecorator metadata.

schema_class = 'TimingDecoratorSchema'

class aries_cloudagent.messaging.decorators.timing_decorator.**TimingDecoratorSchema**(**args*, ***kwargs*)

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Timing decorator schema used in serialization/deserialization.

class Meta

Bases: object

TimingDecoratorSchema metadata.

model_class
alias of *TimingDecorator*

delay_milli
Used by autodoc_mock_imports.

expires_time
Used by autodoc_mock_imports.

in_time
Used by autodoc_mock_imports.

out_time
Used by autodoc_mock_imports.

stale_time
Used by autodoc_mock_imports.

wait_until_time
Used by autodoc_mock_imports.

aries_cloudagent.messaging.decorators.transport_decorator module

The transport decorator (~transport).

This decorator allows changes to agent response behaviour and queue status updates.

class aries_cloudagent.messaging.decorators.transport_decorator.**TransportDecorator**(*, *return_route: str = None*, *return_route_ str = None*, *queued_me int = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing the transport decorator.

class Meta
Bases: object

TransportDecorator metadata.

schema_class = 'TransportDecoratorSchema'

class aries_cloudagent.messaging.decorators.transport_decorator.**TransportDecoratorSchema**(*a **

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Transport decorator schema used in serialization/deserialization.

class Meta
Bases: object

TransportDecoratorSchema metadata.

model_class
alias of *TransportDecorator*

queued_message_count
Used by autodoc_mock_imports.

return_route
Used by autodoc_mock_imports.

return_route_thread
Used by autodoc_mock_imports.

aries_cloudagent.messaging.models package

Submodules

aries_cloudagent.messaging.models.base module

Base classes for Models and Schemas.

class aries_cloudagent.messaging.models.base.**BaseModel**
Bases: abc.ABC

Base model that provides convenience methods.

class Meta
Bases: object

BaseModel meta data.

schema_class = None

Schema
Accessor for the model's schema class.

Returns The schema class

classmethod deserialize(*obj*)
Convert from JSON representation to a model instance.

Parameters **obj** – The dict to load into a model instance

Returns A model instance for this data

classmethod from_json(*json_repr: Union[str, bytes]*)
Parse a JSON string into a model instance.

Parameters **json_repr** – JSON string

Returns A model instance representation of this JSON

serialize(*as_string=False*) → dict
Create a JSON-compatible dict representation of the model instance.

Parameters **as_string** – Return a string of JSON instead of a dict

Returns A dict representation of this model, or a JSON string if as_string is True

to_json() → str
Create a JSON representation of the model instance.

Returns A JSON representation of this message

exception aries_cloudagent.messaging.models.base.**BaseModelError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base exception class for base model errors.

class aries_cloudagent.messaging.models.base.**BaseModelSchema**(**args, **kwargs*)

Bases: sphinx.ext.autodoc.importer._MockObject

BaseModel schema.

class Meta

Bases: object

BaseModelSchema metadata.

model_class = None

ordered = True

skip_values = [None]

Model

Accessor for the schema's model class.

Returns The model class

make_model(*data: dict, **kwargs*)

Return model instance after loading.

Returns A model instance

remove_skipped_values(*data, **kwargs*)

Remove values that are are marked to skip.

Returns Returns this modified data

skip_dump_only(*data, **kwargs*)

Skip fields that are only expected during serialization.

Parameters **data** – The incoming data to clean

Returns The modified data

aries_cloudagent.messaging.models.base.**resolve_class**(*the_cls, relative_cls: type = None*)

Resolve a class.

Parameters

- **the_cls** – The class to resolve
- **relative_cls** – Relative class to resolve from

Returns The resolved class

Raises ClassNotFoundError – If the class could not be loaded

aries_cloudagent.messaging.models.base.**resolve_meta_property**(*obj, prop_name: str, defval=None*)

Resolve a meta property.

Parameters

- **prop_name** – The property to resolve
- **defval** – The default value

Returns The meta property

aries_cloudagent.messaging.models.base_record module

Classes for BaseStorage-based record management.

class aries_cloudagent.messaging.models.base_record.**BaseExchangeRecord**(*id: str = None, state: str = None, *, trace: bool = False, **kwargs*)

Bases: *aries_cloudagent.messaging.models.base_record.BaseRecord*

Represents a base record with event tracing capability.

class aries_cloudagent.messaging.models.base_record.**BaseExchangeSchema**(**args, **kwargs*)

Bases: *aries_cloudagent.messaging.models.base_record.BaseRecordSchema*

Base schema for exchange records.

class Meta

Bases: object

BaseExchangeSchema metadata.

model_class

alias of *BaseExchangeRecord*

trace = <fields.Boolean(default=False, attribute=None, validate=None, required=False,

class aries_cloudagent.messaging.models.base_record.**BaseRecord**(*id: str = None, state: str = None, *, created_at: Union[str, datetime.datetime] = None, updated_at: Union[str, datetime.datetime] = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Represents a single storage record.

CACHE_ENABLED = False

CACHE_TTL = 60

LOG_STATE_FLAG = None

class Meta

Bases: `object`

BaseRecord metadata.

RECORD_ID_NAME = 'id'

RECORD_TYPE = None

TAG_NAMES = {'state'}

WEBHOOK_TOPIC = None

classmethod cache_key(*record_id: str*, *record_type: str = None*)

Assemble a cache key.

Parameters

- **record_id** – The record identifier
- **The cache type identifier, defaulting to RECORD_TYPE** (*record_type*) –

clear_cached(*context: aries_cloudagent.config.injection_context.InjectionContext*)

Clear the cached value of this record, if any.

classmethod clear_cached_key(*context: aries_cloudagent.config.injection_context.InjectionContext*, *cache_key: str*)

Shortcut method to clear a cached key value, if any.

Parameters

- **context** – The injection context to use
- **cache_key** – The unique cache identifier

delete_record(*context: aries_cloudagent.config.injection_context.InjectionContext*)

Remove the stored record.

Parameters **context** – The injection context to use

classmethod from_storage(*record_id: str, record: Mapping[str, Any]*)

Initialize a record from its stored representation.

Parameters

- **record_id** – The unique record identifier
- **record** – The stored representation

classmethod get_cached_key(*context: aries_cloudagent.config.injection_context.InjectionContext*, *cache_key: str*)

Shortcut method to fetch a cached key value.

Parameters

- **context** – The injection context to use
- **cache_key** – The unique cache identifier

classmethod get_tag_map() → Mapping[str, str]

Accessor for the set of defined tags.

classmethod log_state(*context: aries_cloudagent.config.injection_context.InjectionContext, msg: str, params: dict = None, override: bool = False*)

Print a message with increased visibility (for testing).

post_save(*context: aries_cloudagent.config.injection_context.InjectionContext, new_record: bool, last_state: str, webhook: bool = None*)

Perform post-save actions.

Parameters

- **context** – The injection context to use
- **new_record** – Flag indicating if the record was just created
- **last_state** – The previous state value
- **webhook** – Adjust whether the webhook is called

classmethod prefix_tag_filter(*tag_filter: dict*)

Prefix unencrypted tags used in the tag filter.

classmethod query(*context: aries_cloudagent.config.injection_context.InjectionContext, tag_filter: dict = None, post_filter_positive: dict = None, post_filter_negative: dict = None*) → Sequence[aries_cloudagent.messaging.models.base_record.BaseRecord]

Query stored records.

Parameters

- **context** – The injection context to use
- **tag_filter** – An optional dictionary of tag filter clauses
- **post_filter_positive** – Additional value filters to apply matching positively
- **post_filter_negative** – Additional value filters to apply matching negatively

record_tags

Accessor to define implementation-specific tags.

record_value

Accessor to define custom properties for the JSON record value.

classmethod retrieve_by_id(*context: aries_cloudagent.config.injection_context.InjectionContext, record_id: str, cached: bool = True*) → aries_cloudagent.messaging.models.base_record.BaseRecord

Retrieve a stored record by ID.

Parameters

- **context** – The injection context to use
- **record_id** – The ID of the record to find
- **cached** – Whether to check the cache for this record

classmethod retrieve_by_tag_filter(*context: aries_cloudagent.config.injection_context.InjectionContext, tag_filter: dict, post_filter: dict = None*) → aries_cloudagent.messaging.models.base_record.BaseRecord

Retrieve a record by tag filter.

Parameters

- **context** – The injection context to use
- **tag_filter** – The filter dictionary to apply

- **post_filter** – Additional value filters to apply after retrieval

save(*context: aries_cloudagent.config.injection_context.InjectionContext*, **, reason: str = None, log_params: Mapping[str, Any] = None, log_override: bool = False, webhook: bool = None*) → str

Persist the record to storage.

Parameters

- **context** – The injection context to use
- **reason** – A reason to add to the log
- **log_params** – Additional parameters to log
- **webhook** – Flag to override whether the webhook is sent

send_webhook(*context: aries_cloudagent.config.injection_context.InjectionContext, payload: Any, topic: str = None*)

Send a standard webhook.

Parameters

- **context** – The injection context to use
- **payload** – The webhook payload
- **topic** – The webhook topic, defaulting to WEBHOOK_TOPIC

classmethod set_cached_key(*context: aries_cloudagent.config.injection_context.InjectionContext, cache_key: str, value: Any, ttl=None*)

Shortcut method to set a cached key value.

Parameters

- **context** – The injection context to use
- **cache_key** – The unique cache identifier
- **value** – The value to cache
- **ttl** – The cache ttl

storage_record

Accessor for a *StorageRecord* representing this record.

classmethod strip_tag_prefix(*tags: dict*)

Strip tilde from unencrypted tag names.

tags

Accessor for the record tags generated for this record.

value

Accessor for the JSON record value generated for this record.

webhook_payload

Return a JSON-serialized version of the record for the webhook.

webhook_topic

Return the webhook topic value.

class aries_cloudagent.messaging.models.base_record.**BaseRecordSchema**(**args, **kwargs*)

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Schema to allow serialization/deserialization of base records.

class Meta

Bases: object

BaseRecordSchema metadata.

created_at = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<a

state = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, r

updated_at = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<a

aries_cloudagent.messaging.models.base_record.**match_post_filter**(*record: dict, post_filter: dict, positive: bool = True*) → bool

Determine if a record value matches the post-filter.

Parameters

- **record** – record to check
- **post_filter** – filter to apply (empty or None filter matches everything)
- **positive** – whether matching all filter criteria positively or negatively

aries_cloudagent.messaging.schemas package

Submodules

aries_cloudagent.messaging.schemas.routes module

Credential schema admin routes.

class aries_cloudagent.messaging.schemas.routes.**SchemaGetResultsSchema**(**, only=None, exclude=(), many=False, context=None, load_only=(), dump_only=(), partial=False, unknown=None*)

Bases: marshmallow.schema.Schema

Results schema for schema get request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.schemas.routes.**SchemaSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Content for returned schema.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.schemas.routes.**SchemaSendRequestSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Request schema for schema send request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.schemas.routes.**SchemaSendResultsSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Results schema for schema send request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.schemas.routes.**SchemasCreatedResultsSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Results schema for a schemas-created request.

opts = <marshmallow.schema.SchemaOpts object>

aries_cloudagent.messaging.schemas.routes.**register**(*app: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf400a4e0>*)

Register routes.

aries_cloudagent.messaging.schemas.routes.**schemas_created**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf400a4e0>*)

Request handler for retrieving schemas that current agent created.

Parameters **request** – aiohttp request object

Returns The identifiers of matching schemas

aries_cloudagent.messaging.schemas.routes.**schemas_get_schema**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf400a4e0>*)

Request handler for sending a credential offer.

Parameters **request** – aiohttp request object

Returns The schema details.

aries_cloudagent.messaging.schemas.routes.**schemas_send_schema**(*request: <sphinx.ext.autodoc.importer._MockObjec object at 0x7fcaf400a4e0>*)

Request handler for sending a credential offer.

Parameters **request** – aiohttp request object

Returns The schema id sent

aries_cloudagent.messaging.schemas.util module

Schema utilities.

Submodules

aries_cloudagent.messaging.agent_message module

Agent message base class and schema.

class aries_cloudagent.messaging.agent_message.**AgentMessage**(*_id: str = None, _decorators: aries_cloudagent.messaging.decorators.base. = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Agent message base class.

Handler

Accessor for the agent message's handler class.

Returns Handler class

class Meta

Bases: object

AgentMessage metadata.

handler_class = None

message_type = None

schema_class = None

add_trace_decorator(*target: str = 'log', full_thread: bool = True*)

Create a new trace decorator.

Parameters

- **target** – The trace target
- **full_thread** – Full thread flag

add_trace_report(*val: Union[aries_cloudagent.messaging.decorators.trace_decorator.TraceReport, dict]*)

Append a new trace report.

Parameters **val** – The trace target

assign_thread_from(*msg: aries_cloudagent.messaging.agent_message.AgentMessage*)

Copy thread information from a previous message.

Parameters **msg** – The received message containing optional thread information

assign_thread_id(*thid: str, pthid: str = None*)

Assign a specific thread ID.

Parameters

- **thid** – The thread identifier
- **pthid** – The parent thread identifier

assign_trace_decorator(*context, trace*)

Copy trace from a json structure.

Parameters **trace** – string containing trace json stucture

assign_trace_from(*msg: aries_cloudagent.messaging.agent_message.AgentMessage*)

Copy trace information from a previous message.

Parameters **msg** – The received message containing optional trace information

get_signature(*field_name: str*) → aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator

Get the signature for a named field.

Parameters **field_name** – Field name to get the signature for

Returns A SignatureDecorator for the requested field name

set_signature(*field_name: str*, *signature: aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator*

Add or replace the signature for a named field.

Parameters

- **field_name** – Field to set signature on
- **signature** – Signature for the field

sign_field(*field_name: str*, *signer_verkey: str*, *wallet: aries_cloudagent.wallet.base.BaseWallet*, *timestamp=None*) → aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator

Create and store a signature for a named field.

Parameters

- **field_name** – Field to sign
- **signer_verkey** – Verkey of signer
- **wallet** – Wallet to use for signature
- **timestamp** – Optional timestamp for signature

Returns A SignatureDecorator for newly created signature

Raises `ValueError` – If field_name doesn't exist on this message

verify_signatures(*wallet: aries_cloudagent.wallet.base.BaseWallet*) → bool

Verify all associated field signatures.

Parameters **wallet** – Wallet to use in verification

Returns True if all signatures verify, else false

verify_signed_field(*field_name: str*, *wallet: aries_cloudagent.wallet.base.BaseWallet*, *signer_verkey: str = None*) → str

Verify a specific field signature.

Parameters

- **field_name** – The field name to verify
- **wallet** – Wallet to use for the verification
- **signer_verkey** – Verkey of signer to use

Returns The verkey of the signer

Raises

- `ValueError` – If field_name does not exist on this message
- `ValueError` – If the verification fails
- `ValueError` – If the verkey of the signature does not match the
- provided verkey

exception aries_cloudagent.messaging.agent_message.**AgentMessageError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.models.base.BaseModelError*

Base exception for agent message issues.

class aries_cloudagent.messaging.agent_message.**AgentMessageSchema**(**args*, ***kwargs*)

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

AgentMessage schema.

class Meta

Bases: object

AgentMessageSchema metadata.

model_class = None

signed_fields = None

check_dump_decorators(*obj*, ***kwargs*)

Pre-dump hook to validate and load the message decorators.

Parameters **obj** – The AgentMessage object

Raises BaseModelError – If a decorator does not validate

dump_decorators(*data*, ***kwargs*)

Post-dump hook to write the decorators to the serialized output.

Parameters **obj** – The serialized data

Returns The modified data

extract_decorators(*data: Mapping[KT, VT_co], **kwargs*)

Pre-load hook to extract the decorators and check the signed fields.

Parameters **data** – Incoming data to parse

Returns Parsed and modified data

Raises

- ValidationError – If a field signature does not correlate
- to a field in the message
- ValidationError – If the message defines both a field signature
- and a value for the same field
- ValidationError – If there is a missing field signature

populate_decorators(*obj*, ***kwargs*)

Post-load hook to populate decorators on the message.

Parameters **obj** – The AgentMessage object

Returns The AgentMessage object with populated decorators

replace_signatures(*data*, ***kwargs*)

Post-dump hook to write the signatures to the serialized output.

Parameters **obj** – The serialized data

Returns The modified data

aries_cloudagent.messaging.base_handler module

A Base handler class for all message handlers.

class aries_cloudagent.messaging.base_handler.**BaseHandler**

Bases: abc.ABC

Abstract base class for handlers.

handle(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Abstract method for handler logic.

Parameters

- **context** – Request context object
- **responder** – A responder object

exception aries_cloudagent.messaging.base_handler.**HandlerException**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Exception base class for generic handler errors.

aries_cloudagent.messaging.error module

Messaging-related error classes and codes.

exception aries_cloudagent.messaging.error.**MessageParseError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Message parse error.

error_code = 'message_parse_error'

exception aries_cloudagent.messaging.error.**MessagePrepareError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Message preparation error.

error_code = 'message_prepare_error'

aries_cloudagent.messaging.request_context module

Request context class.

A request context provides everything required by handlers and other parts of the system to process a message.

class aries_cloudagent.messaging.request_context.**RequestContext**(*, *base_context: aries_cloudagent.config.injection_conte = None*, *settings: Mapping[str, object] = None*)

Bases: *aries_cloudagent.config.injection_context.InjectionContext*

Context established by the Conductor and passed into message handlers.

connection_ready
Accessor for the flag indicating an active connection with the sender.

Returns True if the connection is active, else False

connection_record
Accessor for the related connection record.

copy() → aries_cloudagent.messaging.request_context.RequestContext
Produce a copy of the request context instance.

default_endpoint
Accessor for the default agent endpoint (from agent config).

Returns The default agent endpoint

default_label
Accessor for the default agent label (from agent config).

Returns The default label

message
Accessor for the deserialized message instance.

Returns This context's agent message

message_receipt
Accessor for the message receipt information.

Returns This context's message receipt information

aries_cloudagent.messaging.responder module

A message responder.

The responder is provided to message handlers to enable them to send a new message in response to the message being handled.

class aries_cloudagent.messaging.responder.**BaseResponder**(*, *connection_id: str = None*, *reply_session_id: str = None*, *reply_to_verkey: str = None*)

Bases: abc.ABC

Interface for message handlers to send responses.

create_outbound(*message: Union[aries_cloudagent.messaging.agent_message.AgentMessage, str, bytes], *, connection_id: str = None, reply_session_id: str = None, reply_thread_id: str = None, reply_to_verkey: str = None, target: aries_cloudagent.connections.models.connection_target.ConnectionTarget = None, target_list: Sequence[aries_cloudagent.connections.models.connection_target.ConnectionTarget] = None, to_session_only: bool = False*) → aries_cloudagent.transport.outbound.message.OutboundMessage

Create an OutboundMessage from a message payload.

send(*message: Union[aries_cloudagent.messaging.agent_message.AgentMessage, str, bytes], **kwargs*)

Convert a message to an OutboundMessage and send it.

send_outbound(*message: aries_cloudagent.transport.outbound.message.OutboundMessage*)

Send an outbound message.

Parameters **message** – The *OutboundMessage* to be sent

send_reply(*message: Union[aries_cloudagent.messaging.agent_message.AgentMessage, str, bytes], *, connection_id: str = None, target: aries_cloudagent.connections.models.connection_target.ConnectionTarget = None, target_list: Sequence[aries_cloudagent.connections.models.connection_target.ConnectionTarget] = None*)

Send a reply to an incoming message.

Parameters

- **message** – the *AgentMessage*, or pre-packed str or bytes to reply with
- **connection_id** – optionally override the target connection ID
- **target** – optionally specify a *ConnectionTarget* to send to

Raises `ResponderError` – If there is no active connection

send_webhook(*topic: str, payload: dict*)

Dispatch a webhook.

Parameters

- **topic** – the webhook topic identifier
- **payload** – the webhook payload value

class aries_cloudagent.messaging.responder.**MockResponder**

Bases: `aries_cloudagent.messaging.responder.BaseResponder`

Mock responder implementation for use by tests.

send(*message: Union[aries_cloudagent.messaging.agent_message.AgentMessage, str, bytes], **kwargs*)

Convert a message to an OutboundMessage and send it.

send_outbound(*message: aries_cloudagent.transport.outbound.message.OutboundMessage*)

Send an outbound message.

send_reply(*message: Union[aries_cloudagent.messaging.agent_message.AgentMessage, str, bytes], **kwargs*)

Send a reply to an incoming message.

send_webhook(*topic: str, payload: dict*)

Send an outbound message.

exception aries_cloudagent.messaging.responder.**ResponderError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Responder error.

aries_cloudagent.messaging.util module

Utils for messages.

aries_cloudagent.messaging.util.**canon**(*raw_attr_name: str*) → str

Canonicalize input attribute name for indy proofs and credential offers.

Parameters **raw_attr_name** – raw attribute name

Returns canonicalized attribute name

aries_cloudagent.messaging.util.**datetime_now**() → datetime.datetime

Timestamp in UTC.

aries_cloudagent.messaging.util.**datetime_to_str**(*dt: Union[str, datetime.datetime]*) → str

Convert a datetime object to an indy-standard datetime string.

Parameters **dt** – May be a string or datetime to allow automatic conversion

aries_cloudagent.messaging.util.**encode**(*orig: Any*) → str

Encode a credential value as an int.

Encode credential attribute value, purely stringifying any int32 and leaving numeric int32 strings alone, but mapping any other input to a stringified 256-bit (but not 32-bit) integer. Predicates in indy-sdk operate on int32 values properly only when their encoded values match their raw values.

Parameters **orig** – original value to encode

Returns encoded value

aries_cloudagent.messaging.util.**epoch_to_str**(*epoch: int*) → str

Convert epoch seconds to indy-standard datetime string.

Parameters **epoch** – epoch seconds

aries_cloudagent.messaging.util.**str_to_datetime**(*dt: Union[str, datetime.datetime]*) → datetime.datetime

Convert an indy-standard datetime string to a datetime.

Using a fairly lax regex pattern to match slightly different formats. In Python 3.7 datetime.fromisoformat might be used.

Parameters **dt** – May be a string or datetime to allow automatic conversion

aries_cloudagent.messaging.util.**str_to_epoch**(*dt: Union[str, datetime.datetime]*) → int

Convert an indy-standard datetime string to epoch seconds.

Parameters **dt** – May be a string or datetime to allow automatic conversion

aries_cloudagent.messaging.util.**time_now**() → str

Timestamp in ISO format.

aries_cloudagent.messaging.valid module

Validators for schema fields.

class aries_cloudagent.messaging.valid.**Base58SHA256Hash**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against base58 encoding of SHA-256 hash.

EXAMPLE = 'H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV'

PATTERN = '^[<sphinx.ext.autodoc.importer._MockObject object>]{43,44}$'

class aries_cloudagent.messaging.valid.**Base64**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate base64 value.

EXAMPLE = 'ey4uLn0='

PATTERN = '^[a-zA-Z0-9+/]*={0,2}$'

class aries_cloudagent.messaging.valid.**Base64URL**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate base64 value.

EXAMPLE = 'ey4uLn0='

PATTERN = '^[-_a-zA-Z0-9]*={0,2}$'

class aries_cloudagent.messaging.valid.**Base64URLNoPad**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate base64 value.

EXAMPLE = 'ey4uLn0'

PATTERN = '^[-_a-zA-Z0-9]*$'

class aries_cloudagent.messaging.valid.**DIDKey**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against DID key specification.

EXAMPLE = 'did:key:z6MkpTHR8VNsBxYAAWHut2Geadd9jSwuBV8xRoAnwWsdvktH'

PATTERN = '^did:key:z[<sphinx.ext.autodoc.importer._MockObject object>]+$'

class aries_cloudagent.messaging.valid.**IndyCredDefId**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against indy credential definition identifier specification.

EXAMPLE = 'WgWxqztrNooG92RXvxSTWv:3:CL:20:tag'

PATTERN = '^([<sphinx.ext.autodoc.importer._MockObject object>]{21,22}):3:CL:(([1-9][0-

class aries_cloudagent.messaging.valid.**IndyDID**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against indy DID.

EXAMPLE = 'WgWxqztrNooG92RXvxSTWv'

PATTERN = '^(did:sov:)?[<sphinx.ext.autodoc.importer._MockObject object>]{21,22}$'

class aries_cloudagent.messaging.valid.**IndyISO8601DateTime**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against ISO 8601 datetime format, indy profile.

EXAMPLE = '2020-04-23 17:31:10Z'

PATTERN = '^\\d{4}-\\d\\d-\\d\\d[T]\\d\\d:\\d\\d(?:\\:(?:\\d\\d(?:\\.\\d{1,6})?))?(?:

class aries_cloudagent.messaging.valid.**IndyPredicate**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against indy predicate.

EXAMPLE = '>='

class aries_cloudagent.messaging.valid.**IndyRawPublicKey**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against indy (Ed25519VerificationKey2018) raw public key.

EXAMPLE = 'H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV'

PATTERN = '^[<sphinx.ext.autodoc.importer._MockObject object>]{43,44}$'

class aries_cloudagent.messaging.valid.**IndyRevRegId**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against indy revocation registry identifier specification.

EXAMPLE = 'WgWxqztrNooG92RXvxSTWv:4:WgWxqztrNooG92RXvxSTWv:3:CL:20:tag:CL_ACCUM:0'

PATTERN = '^([<sphinx.ext.autodoc.importer._MockObject object>]{21,22}):4:([<sphinx.ex

class aries_cloudagent.messaging.valid.**IndySchemaId**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against indy schema identifier specification.

EXAMPLE = 'WgWxqztrNooG92RXvxSTWv:2:schema_name:1.0'

PATTERN = '^[<sphinx.ext.autodoc.importer._MockObject object>]{21,22}:2:.+:[0-9.]+$'

class aries_cloudagent.messaging.valid.**IndyVersion**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against indy version specification.

EXAMPLE = '1.0'

PATTERN = '^[0-9.]+$'

class aries_cloudagent.messaging.valid.**IntEpoch**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against (integer) epoch format.

EXAMPLE = 1587663070

class aries_cloudagent.messaging.valid.**JSONWebToken**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate JSON Web Token.

EXAMPLE = 'eyJhbGciOiJFZERTQSJ9.eyJhIjogIjAifQ.dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFO

PATTERN = '^[-_a-zA-Z0-9]*\\.[-_a-zA-Z0-9]*\\.[-_a-zA-Z0-9]*$'

class aries_cloudagent.messaging.valid.**JWSHeaderKid**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate value against JWS header kid.

EXAMPLE = 'did:sov:LjgpST2rjsoxYegQDRm7EL#keys-4'

PATTERN = '^did:(?:key:z[<sphinx.ext.autodoc.importer._MockObject object>]+|sov:[<sphi

class aries_cloudagent.messaging.valid.**SHA256Hash**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate (binhex-encoded) SHA256 value.

EXAMPLE = '617a48c7c8afe0521efdc03e5bb0ad9e655893e6b4b51f0e794d70fba132aacb'

PATTERN = '^[a-fA-F0-9+/]{64}$'

class aries_cloudagent.messaging.valid.**UUIDFour**
Bases: sphinx.ext.autodoc.importer._MockObject

Validate UUID4: 8-4-4-4-12 hex digits, the 13th of which being 4.

EXAMPLE = '3fa85f64-5717-4562-b3fc-2c963f66afa6'

PATTERN = '[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-4[a-fA-F0-9]{3}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12

1.1.10 aries_cloudagent.storage package

Submodules

aries_cloudagent.storage.base module

Abstract base classes for non-secrets storage.

class aries_cloudagent.storage.base.**BaseStorage**
Bases: abc.ABC

Abstract Non-Secrets interface.

add_record(*record: aries_cloudagent.storage.record.StorageRecord*)
Add a new record to the store.

Parameters **record** – *StorageRecord* to be stored

delete_record(*record: aries_cloudagent.storage.record.StorageRecord*)
Delete an existing record.

Parameters **record** – *StorageRecord* to delete

delete_record_tags(*record: aries_cloudagent.storage.record.StorageRecord, tags: (typing.Sequence, typing.Mapping)*)
Update an existing stored record's tags.

Parameters

- **record** – *StorageRecord* to delete
- **tags** – Tags

get_record(*record_type: str, record_id: str, options: Mapping[KT, VT_co] = None*) → aries_cloudagent.storage.record.StorageRecord
Fetch a record from the store by type and ID.

Parameters

- **record_type** – The record type
- **record_id** – The record id
- **options** – A dictionary of backend-specific options

Returns A *StorageRecord* instance

search_records(*type_filter: str, tag_query: Mapping[KT, VT_co] = None, page_size: int = None, options: Mapping[KT, VT_co] = None*) → aries_cloudagent.storage.base.BaseStorageRecordSearch

Create a new record query.

Parameters

- **type_filter** – Filter string
- **tag_query** – Tags to query
- **page_size** – Page size
- **options** – Dictionary of backend-specific options

Returns An instance of *BaseStorageRecordSearch*

update_record_tags(*record: aries_cloudagent.storage.record.StorageRecord, tags: Mapping[KT, VT_co]*)

Update an existing stored record's tags.

Parameters

- **record** – *StorageRecord* to update
- **tags** – New tags

update_record_value(*record: aries_cloudagent.storage.record.StorageRecord, value: str*)

Update an existing stored record's value.

Parameters

- **record** – *StorageRecord* to update
- **value** – The new value

class aries_cloudagent.storage.base.**BaseStorageRecordSearch**(*store: aries_cloudagent.storage.base.BaseStorage, type_filter: str, tag_query: Mapping[KT, VT_co], page_size: int = None, options: Mapping[KT, VT_co] = None*)

Bases: abc.ABC

Represent an active stored records search.

close()

Dispose of the search query.

fetch(*max_count: int*) → Sequence[aries_cloudagent.storage.record.StorageRecord]

Fetch the next list of results from the store.

Parameters **max_count** – Max number of records to return

Returns A list of *StorageRecord*

fetch_all() → Sequence[aries_cloudagent.storage.record.StorageRecord]
Fetch all records from the query.

fetch_single() → aries_cloudagent.storage.record.StorageRecord
Fetch a single query result.

handle
Handle a search request.

open()
Start the search query.

opened
Accessor for open state.

Returns True if opened, else False

option(*name: str*, *default=None*)
Fetch a named search option, if defined.

Returns The option value or default

options
Accessor for the search options.

Returns The search options

page_size
Accessor for page size.

Returns The page size

store
BaseStorage backend for this implementation.

Returns The *BaseStorage* implementation being used

tag_query
Accessor for tag query.

Returns The tag query

type_filter
Accessor for type filter.

Returns The type filter

aries_cloudagent.storage.basic module

Basic in-memory storage implementation (non-wallet).

class aries_cloudagent.storage.basic.**BasicStorage**(*_wallet: aries_cloudagent.wallet.base.BaseWallet = None*)

Bases: *aries_cloudagent.storage.base.BaseStorage*

Basic in-memory storage class.

add_record(*record: aries_cloudagent.storage.record.StorageRecord*)
Add a new record to the store.

Parameters **record** – *StorageRecord* to be stored

Raises

- `StorageError` – If no record is provided
- `StorageError` – If the record has no ID

delete_record(*record: aries_cloudagent.storage.record.StorageRecord*)

Delete a record.

Parameters **record** – *StorageRecord* to delete

Raises `StorageNotFoundError` – If record not found

delete_record_tags(*record: aries_cloudagent.storage.record.StorageRecord, tags: (typing.Sequence, typing.Mapping)*)

Update an existing stored record's tags.

Parameters

- **record** – *StorageRecord* to delete
- **tags** – Tags

Raises `StorageNotFoundError` – If record not found

get_record(*record_type: str, record_id: str, options: Mapping[KT, VT_co] = None*) → aries_cloudagent.storage.record.StorageRecord

Fetch a record from the store by type and ID.

Parameters

- **record_type** – The record type
- **record_id** – The record id
- **options** – A dictionary of backend-specific options

Returns A *StorageRecord* instance

Raises `StorageNotFoundError` – If the record is not found

search_records(*type_filter: str, tag_query: Mapping[KT, VT_co] = None, page_size: int = None, options: Mapping[KT, VT_co] = None*) → aries_cloudagent.storage.basic.BasicStorageRecordSearch

Search stored records.

Parameters

- **type_filter** – Filter string
- **tag_query** – Tags to query
- **page_size** – Page size
- **options** – Dictionary of backend-specific options

Returns An instance of *BaseStorageRecordSearch*

update_record_tags(*record: aries_cloudagent.storage.record.StorageRecord, tags: Mapping[KT, VT_co]*)

Update an existing stored record's tags.

Parameters

- **record** – *StorageRecord* to update
- **tags** – New tags

Raises `StorageNotFoundError` – If record not found

update_record_value(*record: aries_cloudagent.storage.record.StorageRecord*, *value: str*)
Update an existing stored record's value.

Parameters

- **record** – *StorageRecord* to update
- **value** – The new value

Raises `StorageNotFoundError` – If record not found

class aries_cloudagent.storage.basic.**BasicStorageRecordSearch**(*store: aries_cloudagent.storage.basic.BasicStora type_filter: str, tag_query: Mapping[KT, VT_co], page_size: int = None, options: Mapping[KT, VT_co] = None*)

Bases: *aries_cloudagent.storage.base.BaseStorageRecordSearch*

Represent an active stored records search.

close()
Dispose of the search query.

fetch(*max_count: int*) → Sequence[aries_cloudagent.storage.record.StorageRecord]
Fetch the next list of results from the store.

Parameters **max_count** – Max number of records to return

Returns A list of *StorageRecord*

Raises `StorageSearchError` – If the search query has not been opened

open()
Start the search query.

opened
Accessor for open state.

Returns True if opened, else False

aries_cloudagent.storage.basic.**basic_tag_query_match**(*tags: dict*, *tag_query: dict*) → bool

Match simple tag filters (string values).

aries_cloudagent.storage.basic.**basic_tag_value_match**(*value: str*, *match: dict*) → bool
Match a single tag against a tag subquery.

TODO: What type coercion is needed? (support int or float values?)

aries_cloudagent.storage.error module

Storage-related exceptions.

exception aries_cloudagent.storage.error.**StorageDuplicateError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.storage.error.StorageError*

Duplicate record found in storage.

exception aries_cloudagent.storage.error.**StorageError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base class for Storage errors.

exception aries_cloudagent.storage.error.**StorageNotFoundError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.storage.error.StorageError*

Record not found in storage.

exception aries_cloudagent.storage.error.**StorageSearchError**(**args*, *error_code: str = None*, ***kwargs*)

Bases: *aries_cloudagent.storage.error.StorageError*

General exception during record search.

aries_cloudagent.storage.indy module

Indy implementation of BaseStorage interface.

class aries_cloudagent.storage.indy.**IndyStorage**(*wallet: aries_cloudagent.wallet.indy.IndyWallet*)

Bases: *aries_cloudagent.storage.base.BaseStorage*

Indy Non-Secrets interface.

add_record(*record: aries_cloudagent.storage.record.StorageRecord*)

Add a new record to the store.

Parameters **record** – *StorageRecord* to be stored

delete_record(*record: aries_cloudagent.storage.record.StorageRecord*)

Delete a record.

Parameters **record** – *StorageRecord* to delete

Raises

- `StorageNotFoundError` – If record not found
- `StorageError` – If a libindy error occurs

delete_record_tags(*record: aries_cloudagent.storage.record.StorageRecord*, *tags: (typing.Sequence, typing.Mapping)*)

Update an existing stored record's tags.

Parameters

- **record** – *StorageRecord* to delete
- **tags** – Tags

get_record(*record_type: str*, *record_id: str*, *options: Mapping[KT, VT_co] = None*) → aries_cloudagent.storage.record.StorageRecord

Fetch a record from the store by type and ID.

Parameters

- **record_type** – The record type
- **record_id** – The record id

- **options** – A dictionary of backend-specific options

Returns A *StorageRecord* instance

Raises

- `StorageError` – If the record is not provided
- `StorageError` – If the record ID not provided
- `StorageNotFoundError` – If the record is not found
- `StorageError` – If record not found

search_records(*type_filter: str, tag_query: Mapping[KT, VT_co] = None, page_size: int = None, options: Mapping[KT, VT_co] = None*) → aries_cloudagent.storage.indy.IndyStorageRecordSearch

Search stored records.

Parameters

- **type_filter** – Filter string
- **tag_query** – Tags to query
- **page_size** – Page size
- **options** – Dictionary of backend-specific options

Returns An instance of *IndyStorageRecordSearch*

update_record_tags(*record: aries_cloudagent.storage.record.StorageRecord, tags: Mapping[KT, VT_co]*)

Update an existing stored record's tags.

Parameters

- **record** – *StorageRecord* to update
- **tags** – New tags

Raises

- `StorageNotFoundError` – If record not found
- `StorageError` – If a libindy error occurs

update_record_value(*record: aries_cloudagent.storage.record.StorageRecord, value: str*)

Update an existing stored record's value.

Parameters

- **record** – *StorageRecord* to update
- **value** – The new value

Raises

- `StorageNotFoundError` – If record not found
- `StorageError` – If a libindy error occurs

wallet

Accessor for IndyWallet instance.

class aries_cloudagent.storage.indy.**IndyStorageRecordSearch**(*store: aries_cloudagent.storage.indy.IndyStorage, type_filter: str, tag_query: Mapping[KT, VT_co], page_size: int = None, options: Mapping[KT, VT_co] = None*)

Bases: *aries_cloudagent.storage.base.BaseStorageRecordSearch*

Represent an active stored records search.

close()

Dispose of the search query.

fetch(*max_count: int*) → Sequence[aries_cloudagent.storage.record.StorageRecord]

Fetch the next list of results from the store.

Parameters **max_count** – Max number of records to return

Returns A list of *StorageRecord*

Raises StorageSearchError – If the search query has not been opened

handle

Accessor for search handle.

Returns The handle

open()

Start the search query.

opened

Accessor for open state.

Returns True if opened, else False

aries_cloudagent.storage.provider module

Default storage provider classes.

class aries_cloudagent.storage.provider.**StorageProvider**

Bases: *aries_cloudagent.config.base.BaseProvider*

Provider for the default configurable storage classes.

STORAGE_TYPES = {'basic': 'aries_cloudagent.storage.basic.BasicStorage', 'indy': 'ar

provide(*settings: aries_cloudagent.config.base.BaseSettings, injector: aries_cloudagent.config.base.BaseInjector*)

Create and return the storage instance.

aries_cloudagent.storage.record module

Record instance stored and searchable by BaseStorage implementation.

class aries_cloudagent.storage.record.**StorageRecord**

Bases: *aries_cloudagent.storage.record.StorageRecord*

Storage record class.

1.1.11 aries_cloudagent.transport package

Subpackages

aries_cloudagent.transport.inbound package

Submodules

aries_cloudagent.transport.inbound.base module

Base inbound transport class.

class aries_cloudagent.transport.inbound.base.**BaseInboundTransport**(*scheme: str, create_session: Callable, *, max_message_size: int = 0, wire_format: aries_cloudagent.transport.wire_fo = None*)

Bases: abc.ABC

Base inbound transport class.

create_session(*, *accept_undelivered: bool = False*, *can_respond: bool = False*, *client_info: dict = None*, *wire_format: aries_cloudagent.transport.wire_format.BaseWireFormat = None*) → Awaitable[aries_cloudagent.transport.inbound.session.InboundSession]

Create a new inbound session.

Parameters

- **accept_undelivered** – Flag for accepting undelivered messages
- **can_respond** – Flag indicating that the transport can send responses
- **client_info** – Request-specific client information
- **wire_format** – Optionally override the session wire format

max_message_size

Accessor for this transport's max message size.

scheme

Accessor for this transport's scheme.

start() → None

Start listening for on this transport.

stop() → None

Stop listening for on this transport.

class aries_cloudagent.transport.inbound.base.**InboundTransportConfiguration**(*module, host, port*)

Bases: tuple

host

Alias for field number 1

module
Alias for field number 0

port
Alias for field number 2

exception aries_cloudagent.transport.inbound.base.**InboundTransportError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.transport.error.TransportError*

Generic inbound transport error.

exception aries_cloudagent.transport.inbound.base.**InboundTransportRegistrationError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.transport.inbound.base.InboundTransportError*

Error in loading an inbound transport.

exception aries_cloudagent.transport.inbound.base.**InboundTransportSetupError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.transport.inbound.base.InboundTransportError*

Setup error for an inbound transport.

aries_cloudagent.transport.inbound.delivery_queue module

The Delivery Queue.

The delivery queue holds and manages messages that have not yet been delivered to their intended destination.

class aries_cloudagent.transport.inbound.delivery_queue.**DeliveryQueue**

Bases: object

DeliveryQueue class.

Manages undelivered messages.

add_message(*msg: aries_cloudagent.transport.outbound.message.OutboundMessage*)
Add an OutboundMessage to delivery queue.

The message is added once per recipient key

Parameters **msg** – The OutboundMessage to add

expire_messages(*ttl=None*)
Expire messages that are past the time limit.

Parameters **ttl** – Optional. Allows override of configured ttl

get_one_message_for_key(*key: str*)
Remove and return a matching message.

Parameters **key** – The key to use for lookup

has_message_for_key(*key: str*)
Check for queued messages by key.

Parameters **key** – The key to use for lookup

inspect_all_messages_for_key(*key: str*)
Return all messages for key.

Parameters **key** – The key to use for lookup

message_count_for_key(*key: str*)
Count of queued messages by key.

Parameters **key** – The key to use for lookup

remove_message_for_key(*key: str*, *msg: aries_cloudagent.transport.outbound.message.OutboundMessage*)
Remove specified message from queue for key.

Parameters

- **key** – The key to use for lookup
- **msg** – The message to remove from the queue

class aries_cloudagent.transport.inbound.delivery_queue.**QueuedMessage**(*msg: aries_cloudagent.transport.ou*

Bases: object

Wrapper Class for queued messages.

Allows tracking Metadata.

older_than(*compare_timestamp: float*) → bool
Age Comparison.

Allows you to test age as compared to the provided timestamp.

Parameters **compare_timestamp** – The timestamp to compare

aries_cloudagent.transport.inbound.http module

Http Transport classes and functions.

class aries_cloudagent.transport.inbound.http.**HttpTransport**(*host: str*, *port: int*, *create_session*, ***kwargs*)

Bases: *aries_cloudagent.transport.inbound.base.BaseInboundTransport*

Http Transport class.

inbound_message_handler(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf34c0908>*)
Message handler for inbound messages.

Parameters **request** – aiohttp request object

Returns The web response

invite_message_handler(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf34c0908>*)

Message handler for invites.

Parameters **request** – aiohttp request object

Returns The web response

make_application() → <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf34c0908>

Construct the aiohttp application.

start() → None

Start this transport.

Raises `InboundTransportSetupError` – If there was an error starting the webserver

stop() → None

Stop this transport.

aries_cloudagent.transport.inbound.manager module

Inbound transport manager.

class aries_cloudagent.transport.inbound.manager.**InboundTransportManager**(*context: aries_cloudagent.config.in receive_inbound: Coroutine[T_co, T_contra, V_co], return_inbound: Callable = None*)

Bases: `object`

Inbound transport manager class.

closed_session(*session: aries_cloudagent.transport.inbound.session.InboundSession*)

Clean up a closed session.

Returns an undelivered message to the caller if possible.

create_session(*transport_type: str, *, accept_undelivered: bool = False, can_respond: bool = False, client_info: dict = None, wire_format: aries_cloudagent.transport.wire_format.BaseWireFormat = None*)

Create a new inbound session.

Parameters

- **transport_type** – The inbound transport identifier
- **accept_undelivered** – Flag for accepting undelivered messages
- **can_respond** – Flag indicating that the transport can send responses
- **client_info** – An optional dict describing the client
- **wire_format** – Override the wire format for this session

dispatch_complete(*message: aries_cloudagent.transport.inbound.message.InboundMessage, completed: aries_cloudagent.utils.task_queue.CompletedTask*)
Handle completion of message dispatch.

get_transport_instance(*transport_id: str*) → aries_cloudagent.transport.inbound.base.BaseInboundTransport
Get an instance of a running transport by ID.

process_undelivered(*session: aries_cloudagent.transport.inbound.session.InboundSession*)
Interact with undelivered queue to find applicable messages.

Parameters **session** – The inbound session

register(*config: aries_cloudagent.transport.inbound.base.InboundTransportConfiguration*) → str
Register transport module.

Parameters **config** – The inbound transport configuration

register_transport(*transport: aries_cloudagent.transport.inbound.base.BaseInboundTransport, transport_id: str*) → str
Register a new inbound transport class.

Parameters

- **transport** – Transport instance to register
- **transport_id** – The transport ID to register

return_to_session(*outbound: aries_cloudagent.transport.outbound.message.OutboundMessage*) → bool
Return an outbound message via an open session, if possible.

return_undelivered(*outbound: aries_cloudagent.transport.outbound.message.OutboundMessage*) → bool
Add an undelivered message to the undelivered queue.

At this point the message could not be associated with an inbound session and could not be delivered via an outbound transport.

setup()
Perform setup operations.

start()
Start all registered transports.

start_transport(*transport_id: str*)
Start a registered inbound transport.

Parameters **transport_id** – ID for the inbound transport to start

stop(*wait: bool = True*)
Stop all registered transports.

aries_cloudagent.transport.inbound.message module

Classes representing inbound messages.

class aries_cloudagent.transport.inbound.message.**InboundMessage**(*payload: Union[str, bytes], receipt: aries_cloudagent.transport.inbound.rec*, *connection_id: str = None, session_id: str = None, transport_type: str = None*)

Bases: object

Container class linking a message payload with its receipt details.

aries_cloudagent.transport.inbound.receipt module

Classes for representing message receipt details.

class aries_cloudagent.transport.inbound.receipt.**MessageReceipt**(*, *connection_id: str = None, direct_response_mode: str = None, in_time: datetime.datetime = None, raw_message: str = None, recipient_verkey: str = None, recipient_did: str = None, recipient_did_public: bool = None, sender_did: str = None, sender_verkey: str = None, thread_id: str = None*)

Bases: object

Properties of an agent message's delivery.

REPLY_MODE_ALL = 'all'

REPLY_MODE_NONE = 'none'

REPLY_MODE_THREAD = 'thread'

connection_id

Accessor for the pairwise connection identifier.

Returns This context's connection identifier

direct_response_mode
Accessor for the requested direct response mode.

Returns This context's requested direct response mode

direct_response_requested
Accessor for the the state of the direct response mode.

Returns This context's requested direct response mode

in_time
Accessor for the datetime the message was received.

Returns This context's received time

raw_message
Accessor for the raw message text.

Returns The raw message text

recipient_did
Accessor for the recipient DID which corresponds with the verkey.

Returns The recipient DID

recipient_did_public
Check if the recipient did is public.

Indicates whether the message is associated with a public (ledger) recipient DID.

Returns True if the recipient's DID is public, else false

recipient_verkey
Accessor for the recipient verkey key used to pack the incoming request.

Returns The recipient verkey

sender_did
Accessor for the sender DID which corresponds with the verkey.

Returns The sender did

sender_verkey
Accessor for the sender public key used to pack the incoming request.

Returns This context's sender's verkey

thread_id
Accessor for the identifier of the message thread.

Returns The delivery thread ID

aries_cloudagent.transport.inbound.session module

Inbound connection handling classes.

class aries_cloudagent.transport.inbound.session.**AcceptResult**(*accepted: bool, retry: bool = False*)

Bases: object

Represent the result of accept_response.

class aries_cloudagent.transport.inbound.session.**InboundSession**(*, *context: aries_cloudagent.config.injection_conte in- bound_handler: Callable, ses- sion_id: str, wire_format: aries_cloudagent.transport.wire_forma ac- cept_undelivered: bool = False, can_respond: bool = False, client_info: dict = None, close_handler: Callable = None, re- ply_mode: str = None, re- ply_thread_ids: Sequence[str] = None, re- ply_verkeys: Sequence[str] = None, trans- port_type: str = None*)

Bases: object

Track an open transport connection for direct routing of outbound messages.

accept_response(*message: aries_cloudagent.transport.outbound.message.OutboundMessage*) → aries_cloudagent.transport.inbound.session.AcceptResult

Try to queue an outbound message if it applies to this session.

Returns: a tuple of (message buffered, retry later)

add_reply_thread_ids(**thids*)

Add a thread ID to the set of potential reply targets.

add_reply_verkeys(**verkeys*)

Add a verkey to the set of potential reply targets.

can_respond

Accessor for the session can-respond state.

clear_response()

Handle when the buffered response message has been delivered.

close()

Setter for the session closed state.

closed

Accessor for the session closed state.

encode_outbound(*outbound: aries_cloudagent.transport.outbound.message.OutboundMessage*) → aries_cloudagent.transport.outbound.message.OutboundMessage

Apply wire formatting to an outbound message.

parse_inbound(*payload_enc: Union[str, bytes]*) → aries_cloudagent.transport.inbound.message.InboundMessage
Convert a message payload and to an inbound message.

process_inbound(*message: aries_cloudagent.transport.inbound.message.InboundMessage*)
Process an incoming message and update the session metadata as necessary.

Parameters **message** – The inbound message instance

receive(*payload_enc: Union[str, bytes]*) → aries_cloudagent.transport.inbound.message.InboundMessage
Receive a new message payload and dispatch the message.

receive_inbound(*message: aries_cloudagent.transport.inbound.message.InboundMessage*)
Deliver the inbound message to the conductor.

reply_mode
Accessor for the session reply mode.

reply_thread_ids
Accessor for the reply thread IDs.

reply_verkeys
Accessor for the reply verkeys.

response_buffered
Check if a response is currently buffered.

select_outbound(*message: aries_cloudagent.transport.outbound.message.OutboundMessage*) → bool
Determine if an outbound message should be sent to this session.

Parameters **message** – The outbound message to be checked

set_response(*message: aries_cloudagent.transport.outbound.message.OutboundMessage*)
Set the contents of the response message buffer.

wait_response() → Union[str, bytes]
Wait for a response to be buffered and pack it.

aries_cloudagent.transport.inbound.ws module

Websockets Transport classes and functions.

class aries_cloudagent.transport.inbound.ws.**WsTransport**(*host: str, port: int, create_session, **kwargs*)
Bases: *aries_cloudagent.transport.inbound.base.BaseInboundTransport*

Websockets Transport class.

inbound_message_handler(*request*)
Message handler for inbound messages.

Parameters **request** – aiohttp request object

Returns The web response

make_application() → <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf304b4e0>
Construct the aiohttp application.

scheme
Accessor for this transport's scheme.

start() → None
Start this transport.

Raises `InboundTransportSetupError` – If there was an error starting the webserver

stop() → None
Stop this transport.

aries_cloudagent.transport.outbound package

Submodules

aries_cloudagent.transport.outbound.base module

Base outbound transport.

class aries_cloudagent.transport.outbound.base.**BaseOutboundTransport**(*wire_format: aries_cloudagent.transport.wire = None*)

Bases: `abc.ABC`

Base outbound transport class.

collector
Accessor for the stats collector instance.

handle_message(*context: aries_cloudagent.config.injection_context.InjectionContext, payload: Union[str, bytes], endpoint: str*)
Handle message from queue.

Parameters

- **context** – the context that produced the message
- **payload** – message payload in string or byte format
- **endpoint** – URI endpoint for delivery

start()
Start the transport.

stop()
Shut down the transport.

wire_format
Accessor for a custom wire format for the transport.

exception aries_cloudagent.transport.outbound.base.**OutboundDeliveryError**(**args, error_code: str = None, **kwargs*)

Bases: `aries_cloudagent.transport.outbound.base.OutboundTransportError`

Base exception when a message cannot be delivered via an outbound transport.

exception aries_cloudagent.transport.outbound.base.**OutboundTransportError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.transport.error.TransportError*

Generic outbound transport error.

exception aries_cloudagent.transport.outbound.base.**OutboundTransportRegistrationError**(**args, error_co str = None, **kwa*

Bases: *aries_cloudagent.transport.outbound.base.OutboundTransportError*

Outbound transport registration error.

aries_cloudagent.transport.outbound.http module

Http outbound transport.

class aries_cloudagent.transport.outbound.http.**HttpTransport**

Bases: *aries_cloudagent.transport.outbound.base.BaseOutboundTransport*

Http outbound transport class.

handle_message(*context: aries_cloudagent.config.injection_context.InjectionContext, payload: Union[str, bytes], endpoint: str*)

Handle message from queue.

Parameters

- **context** – the context that produced the message
- **payload** – message payload in string or byte format
- **endpoint** – URI endpoint for delivery

schemes = ('http', 'https')

start()

Start the transport.

stop()

Stop the transport.

aries_cloudagent.transport.outbound.manager module

Outbound transport manager.

class aries_cloudagent.transport.outbound.manager.**OutboundTransportManager**(*context: aries_cloudagent.confi han- dle_not_delivered: Callable = None*)

Bases: object

Outbound transport manager class.

deliver_queued_message(*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage*) → _asyncio.Task

Kick off delivery of a queued message.

encode_queued_message(*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage*) → _asyncio.Task

Kick off encoding of a queued message.

enqueue_message(*context: aries_cloudagent.config.injection_context.InjectionContext, outbound: aries_cloudagent.transport.outbound.message.OutboundMessage*)

Add an outbound message to the queue.

Parameters

- **context** – The context of the request
- **outbound** – The outbound message to deliver

enqueue_webhook(*topic: str, payload: dict, endpoint: str, max_attempts: int = None*)

Add a webhook to the queue.

Parameters

- **topic** – The webhook topic
- **payload** – The webhook payload
- **endpoint** – The webhook endpoint
- **max_attempts** – Override the maximum number of attempts

Raises OutboundDeliveryError – if the associated transport is not running

finished_deliver(*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage, completed: aries_cloudagent.utils.task_queue.CompletedTask*)

Handle completion of queued message delivery.

finished_encode(*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage, completed: aries_cloudagent.utils.task_queue.CompletedTask*)

Handle completion of queued message encoding.

flush()

Wait for any queued messages to be delivered.

get_registered_transport_for_scheme(*scheme: str*) → str

Find the registered transport ID for a given scheme.

get_running_transport_for_endpoint(*endpoint: str*)

Find the running transport ID to use for a given endpoint.

get_running_transport_for_scheme(*scheme: str*) → str

Find the running transport ID for a given scheme.

get_transport_instance(*transport_id: str*) → aries_cloudagent.transport.outbound.base.BaseOutboundTransport

Get an instance of a running transport by ID.

perform_encode(*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage*)
Perform message encoding.

process_queued() → _asyncio.Task
Start the process to deliver queued messages if necessary.

Returns: the current queue processing task or None

register(*module: str*) → str
Register a new outbound transport by module path.

Parameters **module** – Module name to register

Raises

- `OutboundTransportRegistrationError` – If the imported class cannot be located
- `OutboundTransportRegistrationError` – If the imported class does not specify a schemes attribute
- `OutboundTransportRegistrationError` – If the scheme has already been registered

register_class(*transport_class: Type[aries_cloudagent.transport.outbound.base.BaseOutboundTransport], transport_id: str = None*) → str
Register a new outbound transport class.

Parameters **transport_class** – Transport class to register

Raises

- `OutboundTransportRegistrationError` – If the imported class does not specify a schemes attribute
- `OutboundTransportRegistrationError` – If the scheme has already been registered

setup()
Perform setup operations.

start()
Start all transports and feed messages from the queue.

start_transport(*transport_id: str*)
Start a registered transport.

stop(*wait: bool = True*)
Stop all running transports.

class aries_cloudagent.transport.outbound.manager.**QueuedOutboundMessage**(*context: aries_cloudagent.config.inj mes- sage: aries_cloudagent.transport. tar- get: aries_cloudagent.connectio trans- port_id: str*)

Bases: `object`

Class representing an outbound message pending delivery.

STATE_DELIVER = 'deliver'

STATE_DONE = 'done'

STATE_ENCODE = 'encode'

STATE_NEW = 'new'

STATE_PENDING = 'pending'

STATE_RETRY = 'retry'

aries_cloudagent.transport.outbound.message module

Outbound message representation.

class aries_cloudagent.transport.outbound.message.**OutboundMessage**(*, *connection_id: str = None, enc_payload: Union[str, bytes] = None, endpoint: str = None, payload: Union[str, bytes], reply_session_id: str = None, reply_thread_id: str = None, reply_to_verkey: str = None, reply_from_verkey: str = None, target: aries_cloudagent.connections.model = None, target_list: Sequence[aries_cloudagent.connection = None, to_session_only: bool = False*)

Bases: object

Represents an outgoing message.

aries_cloudagent.transport.outbound.ws module

Websockets outbound transport.

class aries_cloudagent.transport.outbound.ws.**WsTransport**

Bases: *aries_cloudagent.transport.outbound.base.BaseOutboundTransport*

Websockets outbound transport class.

handle_message(*context: aries_cloudagent.config.injection_context.InjectionContext, payload: Union[str, bytes], endpoint: str*)

Handle message from queue.

Parameters

- **context** – the context that produced the message
- **payload** – message payload in string or byte format
- **endpoint** – URI endpoint for delivery

schemes = ('ws', 'wss')

start()

Start the outbound transport.

stop()

Stop the outbound transport.

aries_cloudagent.transport.queue package

Submodules

aries_cloudagent.transport.queue.base module

Abstract message queue.

class aries_cloudagent.transport.queue.base.**BaseMessageQueue**

Bases: abc.ABC

Abstract message queue class.

dequeue(*, *timeout: int = None*)

Dequeue a message.

Returns The dequeued message, or None if a timeout occurs

Raises

- asyncio.CancelledError if the queue has been stopped
- asyncio.TimeoutError if the timeout is reached

enqueue(*message*)

Enqueue a message.

Parameters **message** – The message to add to the end of the queue

Raises asyncio.CancelledError if the queue has been stopped

join()

Wait for the queue to empty.

reset()

Empty the queue and reset the stop event.

stop()
Cancel active iteration of the queue.

task_done()
Indicate that the current task is complete.

aries_cloudagent.transport.queue.basic module

Basic in memory queue.

class aries_cloudagent.transport.queue.basic.**BasicMessageQueue**
Bases: *aries_cloudagent.transport.queue.base.BaseMessageQueue*

Basic in memory queue implementation class.

dequeue(*, *timeout: int = None*)
Dequeue a message.

Returns The dequeued message, or None if a timeout occurs

Raises

- asyncio.CancelledError if the queue has been stopped
- asyncio.TimeoutError if the timeout is reached

enqueue(*message*)
Enqueue a message.

Parameters **message** – The message to add to the end of the queue

Raises asyncio.CancelledError if the queue has been stopped

join()
Wait for the queue to empty.

make_queue()
Create the queue instance.

reset()
Empty the queue and reset the stop event.

stop()
Cancel active iteration of the queue.

task_done()
Indicate that the current task is complete.

Submodules

aries_cloudagent.transport.error module

Transport-related error classes and codes.

exception aries_cloudagent.transport.error.**MessageEncodeError**(**args*, *error_code: str = None*, ***kwargs*)
Bases: *aries_cloudagent.transport.error.WireFormatError*

Message encoding error.

error_code = 'message_encode_error'

exception aries_cloudagent.transport.error.**MessageParseError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.transport.error.WireFormatError*

Message parse error.

error_code = 'message_parse_error'

exception aries_cloudagent.transport.error.**TransportError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base class for all transport errors.

exception aries_cloudagent.transport.error.**WireFormatError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.transport.error.TransportError*

Base class for wire-format errors.

aries_cloudagent.transport.pack_format module

Standard packed message format classes.

class aries_cloudagent.transport.pack_format.**PackWireFormat**

Bases: *aries_cloudagent.transport.wire_format.BaseWireFormat*

Standard DIDComm message parser and serializer.

encode_message(*context: aries_cloudagent.config.injection_context.InjectionContext, message_json: Union[str, bytes], recipient_keys: Sequence[str], routing_keys: Sequence[str], sender_key: str*) → Union[str, bytes]

Encode an outgoing message for transport.

Parameters

- **context** – The injection context for settings and services
- **message_json** – The message body to serialize
- **recipient_keys** – A sequence of recipient verkeys
- **routing_keys** – A sequence of routing verkeys
- **sender_key** – The verification key of the sending agent

Returns The encoded message

Raises MessageEncodeError – If the message could not be encoded

pack(*context: aries_cloudagent.config.injection_context.InjectionContext, message_json: Union[str, bytes], recipient_keys: Sequence[str], routing_keys: Sequence[str], sender_key: str*)

Look up the wallet instance and perform the message pack.

parse_message(*context: aries_cloudagent.config.injection_context.InjectionContext, message_body: Union[str, bytes]*) → Tuple[dict, aries_cloudagent.transport.inbound.receipt.MessageReceipt]

Deserialize an incoming message and further populate the request context.

Parameters

- **context** – The injection context for settings and services
- **message_body** – The body of the message

Returns A tuple of the parsed message and a message receipt instance

Raises

- `MessageParseError` – If the JSON parsing failed
- `MessageParseError` – If a wallet is required but can't be located

unpack(*context: aries_cloudagent.config.injection_context.InjectionContext, message_body: Union[str, bytes], receipt: aries_cloudagent.transport.inbound.receipt.MessageReceipt*)

Look up the wallet instance and perform the message unpack.

aries_cloudagent.transport.stats module

aiohttp stats collector support.

class aries_cloudagent.transport.stats.**StatsTracer**(*collector: aries_cloudagent.utils.stats.Collector, prefix: str*)

Bases: `sphinx.ext.autodoc.importer._MockObject`

Attach hooks to client session events and report statistics.

connection_queued_end(*session, context, params*)

Handle the end of a queued connection.

connection_queued_start(*session, context, params*)

Handle the start of a queued connection.

connection_ready(*session, context, params*)

Handle the end of connection acquisition.

dns_resolvehost_end(*session, context, params*)

Handle the end of a DNS resolution.

dns_resolvehost_start(*session, context, params*)

Handle the start of a DNS resolution.

request_end(*session, context, params*)

Handle the end of request.

request_start(*session, context, params*)

Handle the start of a request.

socket_connect_start(*session, context, params*)

Handle the start of a socket connection.

aries_cloudagent.transport.wire_format module

Abstract wire format classes.

class aries_cloudagent.transport.wire_format.**BaseWireFormat**

Bases: `object`

Abstract messaging wire format.

encode_message(*context: aries_cloudagent.config.injection_context.InjectionContext, message_json: Union[str, bytes], recipient_keys: Sequence[str], routing_keys: Sequence[str], sender_key: str*) → Union[str, bytes]

Encode an outgoing message for transport.

Parameters

- **context** – The injection context for settings and services
- **message_json** – The message body to serialize
- **recipient_keys** – A sequence of recipient verkeys
- **routing_keys** – A sequence of routing verkeys
- **sender_key** – The verification key of the sending agent

Returns The encoded message

Raises `MessageEncodeError` – If the message could not be encoded

parse_message(*context: aries_cloudagent.config.injection_context.InjectionContext, message_body: Union[str, bytes]*) → Tuple[dict, aries_cloudagent.transport.inbound.receipt.MessageReceipt]

Deserialize an incoming message and further populate the request context.

Parameters

- **context** – The injection context for settings and services
- **message_body** – The body of the message

Returns A tuple of the parsed message and a message receipt instance

Raises `MessageParseError` – If the message can't be parsed

class aries_cloudagent.transport.wire_format.**JsonWireFormat**

Bases: *aries_cloudagent.transport.wire_format.BaseWireFormat*

Unencrypted wire format.

encode_message(*context: aries_cloudagent.config.injection_context.InjectionContext, message_json: Union[str, bytes], recipient_keys: Sequence[str], routing_keys: Sequence[str], sender_key: str*) → Union[str, bytes]

Encode an outgoing message for transport.

Parameters

- **context** – The injection context for settings and services
- **message_json** – The message body to serialize
- **recipient_keys** – A sequence of recipient verkeys
- **routing_keys** – A sequence of routing verkeys
- **sender_key** – The verification key of the sending agent

Returns The encoded message

Raises `MessageEncodeError` – If the message could not be encoded

parse_message(*context: aries_cloudagent.config.injection_context.InjectionContext, message_body: Union[str, bytes]*) → Tuple[dict, aries_cloudagent.transport.inbound.receipt.MessageReceipt]

Deserialize an incoming message and further populate the request context.

Parameters

- **context** – The injection context for settings and services
- **message_body** – The body of the message

Returns A tuple of the parsed message and a message receipt instance

Raises `MessageParseError` – If the JSON parsing failed

1.1.12 aries_cloudagent.utils package

Submodules

aries_cloudagent.utils.classloader module

The classloader provides utilties to dynamically load classes and modules.

class aries_cloudagent.utils.classloader.**ClassLoader**

Bases: object

Class used to load classes from modules dynamically.

classmethod load_class(*class_name: str*, *default_module: str = None*, *package: str = None*)

Resolve a complete class path (ie. typing.Dict) to the class itself.

Parameters

- **class_name** – the class name
- **default_module** – the default module to load, if not part of in the class name
- **package** – the parent package to search for the module

Returns The resolved class

Raises

- ClassNotFoundError – If the class could not be resolved at path
- ModuleLoadError – If there was an error loading the module

classmethod load_module(*mod_path: str*, *package: str = None*) → module

Load a module by its absolute path.

Parameters

- **mod_path** – the absolute or relative module path
- **package** – the parent package to search for the module

Returns The resolved module or *None* if the module cannot be found

Raises ModuleLoadError – If there was an error loading the module

classmethod load_subclass_of(*base_class: Type[CT_co]*, *mod_path: str*, *package: str = None*)

Resolve an implementation of a base path within a module.

Parameters

- **base_class** – the base class being implemented
- **mod_path** – the absolute module path
- **package** – the parent package to search for the module

Returns The resolved class

Raises

- ClassNotFoundError – If the module or class implementation could not be found
- ModuleLoadError – If there was an error loading the module

classmethod scan_subpackages(*package: str*) → Sequence[str]

Return a list of sub-packages defined under a named package.

exception aries_cloudagent.utils.classloader.**ClassNotFoundError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Class not found error.

exception aries_cloudagent.utils.classloader.**ModuleLoadError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Module load error.

aries_cloudagent.utils.http module

HTTP utility methods.

exception aries_cloudagent.utils.http.**FetchError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Error raised when an HTTP fetch fails.

aries_cloudagent.utils.http.**fetch**(*url: str, *, headers: dict = None, retry: bool = True, max_attempts: int = 5, interval: float = 1.0, backoff: float = 0.25, request_timeout: float = 10.0, connector: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4246978> = None, session: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4246a90> = None, json: bool = False*)

Fetch from an HTTP server with automatic retries and timeouts.

Parameters

- **url** – the address to fetch
- **headers** – an optional dict of headers to send
- **retry** – flag to retry the fetch
- **max_attempts** – the maximum number of attempts to make
- **interval** – the interval between retries, in seconds
- **backoff** – the backoff interval, in seconds
- **request_timeout** – the HTTP request timeout, in seconds
- **connector** – an optional existing BaseConnector
- **session** – a shared ClientSession
- **json** – flag to parse the result as JSON

`aries_cloudagent.utils.http.fetch_stream`(*url: str, *, headers: dict = None, retry: bool = True, max_attempts: int = 5, interval: float = 1.0, backoff: float = 0.25, request_timeout: float = 10.0, connector: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4246978> = None, session: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf4246a90> = None*)

Fetch from an HTTP server with automatic retries and timeouts.

Parameters

- **url** – the address to fetch
- **headers** – an optional dict of headers to send
- **retry** – flag to retry the fetch
- **max_attempts** – the maximum number of attempts to make
- **interval** – the interval between retries, in seconds
- **backoff** – the backoff interval, in seconds
- **request_timeout** – the HTTP request timeout, in seconds
- **connector** – an optional existing BaseConnector
- **session** – a shared ClientSession
- **json** – flag to parse the result as JSON

aries_cloudagent.utils.repeat module

Utils for repeating tasks.

class `aries_cloudagent.utils.repeat.RepeatAttempt`(*seq: aries_cloudagent.utils.repeat.RepeatSequence, index: int = 1*)

Bases: `object`

Represents the current iteration in a repeat sequence.

final
Check if this is the last instance in the sequence.

next() → aries_cloudagent.utils.repeat.RepeatAttempt
Get the next attempt instance.

next_interval
Calculate the interval before the next attempt.

timeout(*interval: float = None*)
Create a context manager for timing out an attempt.

class `aries_cloudagent.utils.repeat.RepeatSequence`(*limit: int = 0, interval: float = 0.0, backoff: float = 0.0*)

Bases: `object`

Represents a repetition sequence.

next_interval(*index: int*) → float
Calculate the time before the next attempt.

start() → aries_cloudagent.utils.repeat.RepeatAttempt
Get the first attempt in the sequence.

aries_cloudagent.utils.stats module

Classes for tracking performance and timing.

class aries_cloudagent.utils.stats.**Collector**(*, *enabled: bool = True*, *log_path: str = None*)

Bases: object

Collector for a set of statistics.

enabled
Accessor for the collector's enabled property.

extract(*groups: Sequence[str] = None*) → dict
Extract statistics for a specific set of groups.

log(*name: str*, *duration: float*, *start: float = None*)
Log an entry in the statistics if the collector is enabled.

mark(**names*)
Make a custom decorator function for adding to the set of groups.

reset()
Reset the collector's statistics.

results
Accessor for the current set of collected statistics.

timer(**groups*)
Create a new timer attached to this collector.

wrap(*obj, prop_name: Union[str, Sequence[str]], groups: Sequence[str] = None, *, ignore_missing: bool = False*)
Wrap a method on a class or class instance.

wrap_coro(*fn, groups: Sequence[str]*)
Wrap a coroutine instance to collect timing statistics on execution.

wrap_fn(*fn, groups: Sequence[str]*)
Wrap a function instance to collect timing statistics on execution.

class aries_cloudagent.utils.stats.**Stats**

Bases: object

A collection of statistics.

extract(*names: Sequence[str] = None*) → dict
Summarize the stats in a dictionary.

log(*name: str*, *duration: float*)
Log an entry in the stats.

class aries_cloudagent.utils.stats.**Timer**(*collector: aries_cloudagent.utils.stats.Collector, groups: Sequence[str]*)

Bases: object

Timer instance for a running task.

classmethod now()
Fetch a standard timer value.

start() → aries_cloudagent.utils.stats.Timer
Start the timer.

stop()
Stop the timer.

aries_cloudagent.utils.task_queue module

Classes for managing a set of asyncio tasks.

class aries_cloudagent.utils.task_queue.**CompletedTask**(*task: _asyncio.Task, exc_info: Tuple, ident: str = None, timing: dict = None*)

Bases: object

Represent the result of a queued task.

class aries_cloudagent.utils.task_queue.**PendingTask**(*coro: Coroutine[T_co, T_contra, V_co], complete_hook: Callable = None, ident: str = None, task_future: _asyncio.Future = None, queued_time: float = None*)

Bases: object

Represent a task in the queue.

cancel()
Cancel the pending task.

cancelled
Accessor for the cancelled property.

task
Accessor for the task.

class aries_cloudagent.utils.task_queue.**TaskQueue**(*max_active: int = 0, timed: bool = False, trace_fn: Callable = None*)

Bases: object

A class for managing a set of asyncio tasks.

add_active(*task: _asyncio.Task, task_complete: Callable = None, ident: str = None, timing: dict = None*) → _asyncio.Task
Register an active async task with an optional completion callback.

Parameters

- **task** – The asyncio task instance
- **task_complete** – An optional callback to run on completion
- **ident** – A string identifer for the task
- **timing** – An optional dictionary of timing information

add_pending(*pending: aries_cloudagent.utils.task_queue.PendingTask*)
Add a task to the pending queue.

Parameters **pending** – The *PendingTask* to add to the task queue

cancel()
Cancel any pending or active tasks in the queue.

cancel_pending()
Cancel any pending tasks in the queue.

cancelled
Accessor for the cancelled property of the queue.

complete(*timeout: float = None*, *cleanup: bool = True*)
Cancel any pending tasks and wait for, or cancel active tasks.

completed_task(*task: _asyncio.Task*, *task_complete: Callable*, *ident: str*, *timing: dict = None*)
Clean up after a task has completed and run callbacks.

current_active
Accessor for the current number of active tasks in the queue.

current_pending
Accessor for the current number of pending tasks in the queue.

current_size
Accessor for the total number of tasks in the queue.

drain() → _asyncio.Task
Start the process to run queued tasks.

flush()
Wait for any active or pending tasks to be completed.

max_active
Accessor for the maximum number of active tasks in the queue.

put(*coro: Coroutine[T_co, T_contra, V_co], task_complete: Callable = None, ident: str = None*) → aries_cloudagent.utils.task_queue.PendingTask
Add a new task to the queue, delaying execution if busy.

Parameters

- **coro** – The coroutine to run
- **task_complete** – A callback to run on completion
- **ident** – A string identifier for the task

Returns: a future resolving to the asyncio task instance once queued

ready
Accessor for the ready property of the queue.

run(*coro: Coroutine[T_co, T_contra, V_co], task_complete: Callable = None, ident: str = None, timing: dict = None*) → _asyncio.Task
Start executing a coroutine as an async task, bypassing the pending queue.

Parameters

- **coro** – The coroutine to run
- **task_complete** – An optional callback to run on completion
- **ident** – A string identifier for the task
- **timing** – An optional dictionary of timing information

Returns: the new asyncio task instance

wait_for(*timeout: float*)
Wait for all queued tasks to complete with a timeout.

`aries_cloudagent.utils.task_queue.coro_ident`(*coro: Coroutine[T_co, T_contra, V_co]*)
Extract an identifier for a coroutine.

`aries_cloudagent.utils.task_queue.coro_timed`(*coro: Coroutine[T_co, T_contra, V_co], timing: dict*)
Capture timing for a coroutine.

`aries_cloudagent.utils.task_queue.task_exc_info`(*task: _asyncio.Task*)
Extract exception info from an asyncio task.

1.1.13 aries_cloudagent.verifier package

Submodules

aries_cloudagent.verifier.base module

Base Verifier class.

class `aries_cloudagent.verifier.base.BaseVerifier`
Bases: `abc.ABC`

Base class for verifier.

verify_presentation(*presentation_request, presentation, schemas, credential_definitions, rev_reg_defs, rev_reg_entries*)
Verify a presentation.

Parameters

- **presentation_request** – Presentation request data
- **presentation** – Presentation data
- **schemas** – Schema data
- **credential_definitions** – credential definition data
- **rev_reg_defs** – revocation registry definitions
- **rev_reg_entries** – revocation registry entries

aries_cloudagent.verifier.indy module

Indy verifier implementation.

class `aries_cloudagent.verifier.indy.IndyVerifier`(*wallet*)
Bases: *aries_cloudagent.verifier.base.BaseVerifier*

Indy verifier class.

static pre_verify(*pres_req: dict, pres: dict) -> (<enum 'PreVerifyResult'>, <class 'str'>*)
Check for essential components and tampering in presentation.

Visit encoded attribute values against raw, and predicate bounds, in presentation, cross-reference against presentation request.

Parameters

- **pres_req** – presentation request
- **pres** – corresponding presentation

Returns An instance of *PreVerifyResult* representing the validation result

verify_presentation(*presentation_request*, *presentation*, *schemas*, *credential_definitions*, *rev_reg_defs*, *rev_reg_entries*) → bool

Verify a presentation.

Parameters

- **presentation_request** – Presentation request data
- **presentation** – Presentation data
- **schemas** – Schema data
- **credential_definitions** – credential definition data
- **rev_reg_defs** – revocation registry definitions
- **rev_reg_entries** – revocation registry entries

class aries_cloudagent.verifier.indy.**PreVerifyResult**

Bases: enum.Enum

Represent the result of IndyVerifier.pre_verify.

ENCODING_MISMATCH = 'demonstrates tampering with raw values'

INCOMPLETE = 'missing essential components'

OK = 'ok'

1.1.14 aries_cloudagent.wallet package

Abstract and Indy wallet handling.

Submodules

aries_cloudagent.wallet.base module

Wallet base class.

class aries_cloudagent.wallet.base.**BaseWallet**(*config: dict*)

Bases: abc.ABC

Abstract wallet interface.

close()

Close previously-opened wallet, removing it if so configured.

create_local_did(*seed: str = None*, *did: str = None*, *metadata: dict = None*) → aries_cloudagent.wallet.base.DIDInfo

Create and store a new local DID.

Parameters

- **seed** – Optional seed to use for did
- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns The created *DIDInfo*

create_public_did(*seed: str = None, did: str = None, metadata: dict = {}*) → aries_cloudagent.wallet.base.DIDInfo

Create and store a new public DID.

Implicitly flags all other dids as not public.

Parameters

- **seed** – Optional seed to use for did
- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns The created *DIDInfo*

create_signing_key(*seed: str = None, metadata: dict = None*) → aries_cloudagent.wallet.base.KeyInfo

Create a new public/private signing keypair.

Parameters

- **seed** – Optional seed allowing deterministic key creation
- **metadata** – Optional metadata to store with the keypair

Returns A *KeyInfo* representing the new record

created

Check whether the wallet was created on the last open call.

get_local_did(*did: str*) → aries_cloudagent.wallet.base.DIDInfo

Find info for a local DID.

Parameters **did** – The DID to get info for

Returns A *DIDInfo* instance for the DID

get_local_did_for_verkey(*verkey: str*) → aries_cloudagent.wallet.base.DIDInfo

Resolve a local DID from a verkey.

Parameters **verkey** – Verkey to get DID info for

Returns A *DIDInfo* instance for the DID

get_local_dids() → Sequence[aries_cloudagent.wallet.base.DIDInfo]

Get list of defined local DIDs.

Returns A list of *DIDInfo* instances

get_public_did() → aries_cloudagent.wallet.base.DIDInfo

Retrieve the public did.

Returns The created *DIDInfo*

get_signing_key(*verkey: str*) → aries_cloudagent.wallet.base.KeyInfo

Fetch info for a signing keypair.

Parameters **verkey** – The verification key of the keypair

Returns A *KeyInfo* representing the keypair

handle

Get internal wallet reference.

Returns Defaults to None

name
Accessor for the wallet name.

open()
Open wallet, removing and/or creating it if so configured.

opened
Check whether wallet is currently open.

pack_message(*message: str, to_verkeys: Sequence[str], from_verkey: str = None*) → bytes
Pack a message for one or more recipients.

Parameters

- **message** – The message to pack
- **to_verkeys** – The verkeys to pack the message for
- **from_verkey** – The sender verkey

Returns The packed message

replace_local_did_metadata(*did: str*, *metadata: dict*)
Replace the metadata associated with a local DID.

Parameters

- **did** – DID to replace metadata for
- **metadata** – The new metadata

replace_signing_key_metadata(*verkey: str*, *metadata: dict*)
Replace the metadata associated with a signing keypair.

Parameters

- **verkey** – The verification key of the keypair
- **metadata** – The new metadata to store

set_public_did(*did: str*) → aries_cloudagent.wallet.base.DIDInfo
Assign the public did.

Returns The created *DIDInfo*

sign_message(*message: bytes*, *from_verkey: str*) → bytes
Sign a message using the private key associated with a given verkey.

Parameters

- **message** – The message to sign
- **from_verkey** – Sign using the private key related to this verkey

Returns The signature

type
Accessor for the wallet type.

unpack_message(*enc_message: bytes) -> (<class 'str'>, <class 'str'>, <class 'str'>*)
Unpack a message.

Parameters **enc_message** – The encrypted message

Returns (message, from_verkey, to_verkey)

Return type A tuple

verify_message(*message: bytes*, *signature: bytes*, *from_verkey: str*) → bool
Verify a signature against the public key of the signer.

Parameters

- **message** – The message to verify
- **signature** – The signature to verify
- **from_verkey** – Verkey to use in verification

Returns True if verified, else False

class aries_cloudagent.wallet.base.**DIDInfo**(*did*, *verkey*, *metadata*)
Bases: tuple

did
Alias for field number 0

metadata
Alias for field number 2

verkey
Alias for field number 1

class aries_cloudagent.wallet.base.**KeyInfo**(*verkey*, *metadata*)
Bases: tuple

metadata
Alias for field number 1

verkey
Alias for field number 0

aries_cloudagent.wallet.basic module

In-memory implementation of BaseWallet interface.

class aries_cloudagent.wallet.basic.**BasicWallet**(*config: dict = None*)
Bases: *aries_cloudagent.wallet.base.BaseWallet*

In-memory wallet implementation.

WALLET_TYPE = 'basic'

close()
Not applicable to in-memory wallet.

create_local_did(*seed: str = None*, *did: str = None*, *metadata: dict = None*) → aries_cloudagent.wallet.base.DIDInfo
Create and store a new local DID.

Parameters

- **seed** – Optional seed to use for did
- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns A *DIDInfo* instance representing the created DID

Raises WalletDuplicateError – If the DID already exists in the wallet

create_signing_key(*seed: str = None, metadata: dict = None*) → aries_cloudagent.wallet.base.KeyInfo

Create a new public/private signing keypair.

Parameters

- **seed** – Seed to use for signing key
- **metadata** – Optional metadata to store with the keypair

Returns A *KeyInfo* representing the new record

Raises `WalletDuplicateError` – If the resulting verkey already exists in the wallet

created

Check whether the wallet was created on the last open call.

get_local_did(*did: str*) → aries_cloudagent.wallet.base.DIDInfo

Find info for a local DID.

Parameters **did** – The DID to get info for

Returns A *DIDInfo* instance representing the found DID

Raises `WalletNotFoundError` – If the DID is not found

get_local_did_for_verkey(*verkey: str*) → aries_cloudagent.wallet.base.DIDInfo

Resolve a local DID from a verkey.

Parameters **verkey** – The verkey to get the local DID for

Returns A *DIDInfo* instance representing the found DID

Raises `WalletNotFoundError` – If the verkey is not found

get_local_dids() → Sequence[aries_cloudagent.wallet.base.DIDInfo]

Get list of defined local DIDs.

Returns A list of locally stored DIDs as *DIDInfo* instances

get_signing_key(*verkey: str*) → aries_cloudagent.wallet.base.KeyInfo

Fetch info for a signing keypair.

Parameters **verkey** – The verification key of the keypair

Returns A *KeyInfo* representing the keypair

Raises `WalletNotFoundError` – if no keypair is associated with the verification key

name

Accessor for the wallet name.

open()

Not applicable to in-memory wallet.

opened

Check whether wallet is currently open.

Returns True

pack_message(*message: str, to_verkeys: Sequence[str], from_verkey: str = None*) → bytes

Pack a message for one or more recipients.

Parameters

- **message** – The message to pack
- **to_verkeys** – List of verkeys to pack for

- **from_verkey** – Sender verkey to pack from

Returns The resulting packed message bytes

Raises `WalletError` – If the message is not provided

replace_local_did_metadata(*did: str*, *metadata: dict*)
Replace metadata for a local DID.

Parameters

- **did** – The DID to replace metadata for
- **metadata** – The new metadata

Raises `WalletNotFoundError` – If the DID doesn't exist

replace_signing_key_metadata(*verkey: str*, *metadata: dict*)
Replace the metadata associated with a signing keypair.

Parameters

- **verkey** – The verification key of the keypair
- **metadata** – The new metadata to store

Raises `WalletNotFoundError` – if no keypair is associated with the verification key

sign_message(*message: bytes*, *from_verkey: str*) → bytes
Sign a message using the private key associated with a given verkey.

Parameters

- **message** – Message bytes to sign
- **from_verkey** – The verkey to use to sign

Returns A signature

Raises

- `WalletError` – If the message is not provided
- `WalletError` – If the verkey is not provided

type
Accessor for the wallet type.

unpack_message(*enc_message: bytes) -> (<class 'str'>, <class 'str'>, <class 'str'>*)
Unpack a message.

Parameters **enc_message** – The packed message bytes

Returns (message, from_verkey, to_verkey)

Return type A tuple

Raises

- `WalletError` – If the message is not provided
- `WalletError` – If there is a problem unpacking the message

verify_message(*message: bytes*, *signature: bytes*, *from_verkey: str*) → bool
Verify a signature against the public key of the signer.

Parameters

- **message** – Message to verify

- **signature** – Signature to verify
- **from_verkey** – Verkey to use in verification

Returns True if verified, else False

Raises

- `WalletError` – If the verkey is not provided
- `WalletError` – If the signature is not provided
- `WalletError` – If the message is not provided

aries_cloudagent.wallet.crypto module

Cryptography functions used by BasicWallet.

class aries_cloudagent.wallet.crypto.**PackMessageSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: `marshmallow.schema.Schema`

Packed message schema.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.wallet.crypto.**PackRecipientHeaderSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: `marshmallow.schema.Schema`

Packed recipient header schema.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.wallet.crypto.**PackRecipientSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: `marshmallow.schema.Schema`

Packed recipient schema.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.wallet.crypto.**PackRecipientsSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Packed recipients schema.

opts = <marshmallow.schema.SchemaOpts object>

aries_cloudagent.wallet.crypto.**create_keypair**(*seed: bytes = None*) → Tuple[bytes, bytes]

Create a public and private signing keypair from a seed value.

Parameters **seed** – Seed for keypair

Returns A tuple of (public key, secret key)

aries_cloudagent.wallet.crypto.**decode_pack_message**(*enc_message: bytes*, *find_key: Callable*) → Tuple[str, Optional[str], str]

Decode a packed message.

Disassemble and unencrypt a packed message, returning the message content, verification key of the sender (if available), and verification key of the recipient.

Parameters

- **enc_message** – The encrypted message
- **find_key** – Function to retrieve private key

Returns A tuple of (message, sender_vk, recip_vk)

Raises

- ValueError – If the packed message is invalid
- ValueError – If the packed message reipients are invalid
- ValueError – If the pack algorithm is unsupported
- ValueError – If the sender's public key was not provided

aries_cloudagent.wallet.crypto.**decode_pack_message_outer**(*enc_message: bytes*) → Tuple[dict, dict, bool]

Decode the outer wrapper of a packed message and extract the recipients.

Parameters **enc_message** – The encrypted message

Returns: a tuple of the decoded wrapper, recipients, and authcrypt flag

aries_cloudagent.wallet.crypto.**decode_pack_message_payload**(*wrapper: dict*, *payload_key: bytes*) → str

Decode the payload of a packed message once the CEK is known.

Parameters

- **wrapper** – The decoded message wrapper
- **payload_key** – The decrypted payload key

`aries_cloudagent.wallet.crypto.`**`decrypt_plaintext`**(*ciphertext: bytes, recips_bin: bytes, nonce: bytes, key: bytes*) → str

Decrypt the payload of a packed message.

Parameters

- **ciphertext** –
- **recips_bin** –
- **nonce** –
- **key** –

Returns The decrypted string

`aries_cloudagent.wallet.crypto.`**`encode_pack_message`**(*message: str, to_verkeys: Sequence[bytes], from_secret: bytes = None*) → bytes

Assemble a packed message for a set of recipients, optionally including the sender.

Parameters

- **message** – The message to pack
- **to_verkeys** – The verkeys to pack the message for
- **from_secret** – The sender secret

Returns The encoded message

`aries_cloudagent.wallet.crypto.`**`encrypt_plaintext`**(*message: str, add_data: bytes, key: bytes*) → Tuple[bytes, bytes, bytes]

Encrypt the payload of a packed message.

Parameters

- **message** – Message to encrypt
- **add_data** –
- **key** – Key used for encryption

Returns A tuple of (ciphertext, nonce, tag)

`aries_cloudagent.wallet.crypto.`**`extract_pack_recipients`**(*recipients: Sequence[dict]*) → dict

Extract the pack message recipients into a dict indexed by verkey.

Parameters **recipients** – Recipients to locate

Raises `ValueError` – If the recipients block is mal-formatted

`aries_cloudagent.wallet.crypto.`**`extract_payload_key`**(*sender_cek: dict, recip_secret: bytes*) → Tuple[bytes, str]

Extract the payload key from pack recipient details.

Returns: A tuple of the CEK and sender verkey

`aries_cloudagent.wallet.crypto.`**`prepare_pack_recipient_keys`**(*to_verkeys: Sequence[bytes], from_secret: bytes = None*) → Tuple[str, bytes]

Assemble the recipients block of a packed message.

Parameters

- **to_verkeys** – Verkeys of recipients
- **from_secret** – Secret to use for signing keys

Returns A tuple of (json result, key)

`aries_cloudagent.wallet.crypto.random_seed()` → bytes

Generate a random seed value.

Returns A new random seed

`aries_cloudagent.wallet.crypto.seed_to_did(seed: str)` → str

Derive a DID from a seed value.

Parameters **seed** – The seed to derive

Returns The DID derived from the seed

`aries_cloudagent.wallet.crypto.sign_message(message: bytes, secret: bytes)` → bytes

Sign a message using a private signing key.

Parameters

- **message** – The message to sign
- **secret** – The private signing key

Returns The signature

`aries_cloudagent.wallet.crypto.sign_pk_from_sk(secret: bytes)` → bytes

Extract the verkey from a secret signing key.

`aries_cloudagent.wallet.crypto.validate_seed(seed: (<class 'str'>, <class 'bytes'>))` → bytes

Convert a seed parameter to standard format and check length.

Parameters **seed** – The seed to validate

Returns The validated and encoded seed

`aries_cloudagent.wallet.crypto.verify_signed_message(signed: bytes, verkey: bytes)` → bool

Verify a signed message according to a public verification key.

Parameters

- **signed** – The signed message
- **verkey** – The verkey to use in verification

Returns True if verified, else False

aries_cloudagent.wallet.error module

Wallet-related exceptions.

exception `aries_cloudagent.wallet.error.WalletDuplicateError(*args, error_code: str = None, **kwargs)`

Bases: *aries_cloudagent.wallet.error.WalletError*

Duplicate record exception.

exception `aries_cloudagent.wallet.error.WalletError(*args, error_code: str = None, **kwargs)`

Bases: *aries_cloudagent.core.error.BaseError*

General wallet exception.

exception aries_cloudagent.wallet.error.**WalletNotFoundError**(**args, error_code: str = None, **kwargs*)

Bases: *aries_cloudagent.wallet.error.WalletError*

Record not found exception.

aries_cloudagent.wallet.indy module

Indy implementation of BaseWallet interface.

class aries_cloudagent.wallet.indy.**IndyWallet**(*config: dict = None*)

Bases: *aries_cloudagent.wallet.base.BaseWallet*

Indy wallet implementation.

DEFAULT_FRESHNESS = 0

DEFAULT_KEY = ''

DEFAULT_KEY_DERIVIATION = 'ARGON2I_MOD'

DEFAULT_NAME = 'default'

DEFAULT_STORAGE_TYPE = None

KEY_DERIVATION_ARGON2I_INT = 'ARGON2I_INT'

KEY_DERIVATION_ARGON2I_MOD = 'ARGON2I_MOD'

KEY_DERIVATION_RAW = 'RAW'

WALLET_TYPE = 'indy'

close()

Close previously-opened wallet, removing it if so configured.

create(*replace: bool = False*)

Create a new wallet.

Parameters **replace** – Removes the old wallet if True

Raises

- WalletError – If there was a problem removing the wallet
- WalletError – IF there was a libindy error

create_local_did(*seed: str = None, did: str = None, metadata: dict = None*) → aries_cloudagent.wallet.base.DIDInfo

Create and store a new local DID.

Parameters

- **seed** – Optional seed to use for did
- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns A *DIDInfo* instance representing the created DID

Raises

- WalletDuplicateError – If the DID already exists in the wallet

- `WalletError` – If there is a libindy error

create_signing_key(*seed: str = None, metadata: dict = None*) → aries_cloudagent.wallet.base.KeyInfo

Create a new public/private signing keypair.

Parameters

- **seed** – Seed for key
- **metadata** – Optional metadata to store with the keypair

Returns A *KeyInfo* representing the new record

Raises

- `WalletDuplicateError` – If the resulting verkey already exists in the wallet
- `WalletError` – If there is a libindy error

created

Check whether the wallet was created on the last open call.

classmethod generate_wallet_key(*seed: str = None*) → str

Generate a raw Indy wallet key.

get_credential_definition_tag_policy(*credential_definition_id: str*)

Return the tag policy for a given credential definition ID.

get_local_did(*did: str*) → aries_cloudagent.wallet.base.DIDInfo

Find info for a local DID.

Parameters **did** – The DID to get info for

Returns A *DIDInfo* instance representing the found DID

Raises

- `WalletNotFoundError` – If the DID is not found
- `WalletError` – If there is a libindy error

get_local_did_for_verkey(*verkey: str*) → aries_cloudagent.wallet.base.DIDInfo

Resolve a local DID from a verkey.

Parameters **verkey** – The verkey to get the local DID for

Returns A *DIDInfo* instance representing the found DID

Raises `WalletNotFoundError` – If the verkey is not found

get_local_dids() → Sequence[aries_cloudagent.wallet.base.DIDInfo]

Get list of defined local DIDs.

Returns A list of locally stored DIDs as *DIDInfo* instances

get_signing_key(*verkey: str*) → aries_cloudagent.wallet.base.KeyInfo

Fetch info for a signing keypair.

Parameters **verkey** – The verification key of the keypair

Returns A *KeyInfo* representing the keypair

Raises

- `WalletNotFoundError` – If no keypair is associated with the verification key
- `WalletError` – If there is a libindy error

handle
Get internal wallet reference.

Returns A handle to the wallet

master_secret_id
Accessor for the master secret id.

Returns The master secret id

name
Accessor for the wallet name.

Returns The wallet name

open()
Open wallet, removing and/or creating it if so configured.

Raises

- `WalletError` – If wallet not found after creation
- `WalletNotFoundError` – If the wallet is not found
- `WalletError` – If the wallet is already open
- `WalletError` – If there is a libindy error

opened
Check whether wallet is currently open.

Returns True if open, else False

pack_message(*message: str, to_verkeys: Sequence[str], from_verkey: str = None*) → bytes
Pack a message for one or more recipients.

Parameters

- **message** – The message to pack
- **to_verkeys** – List of verkeys to pack for
- **from_verkey** – Sender verkey to pack from

Returns The resulting packed message bytes

Raises

- `WalletError` – If no message is provided
- `WalletError` – If a libindy error occurs

remove()
Remove an existing wallet.

Raises

- `WalletNotFoundError` – If the wallet could not be found
- `WalletError` – If there was an libindy error

replace_local_did_metadata(*did: str, metadata: dict*)
Replace metadata for a local DID.

Parameters

- **did** – The DID to replace metadata for
- **metadata** – The new metadata

replace_signing_key_metadata(*verkey: str*, *metadata: dict*)
Replace the metadata associated with a signing keypair.

Parameters

- **verkey** – The verification key of the keypair
- **metadata** – The new metadata to store

Raises `WalletNotFoundError` – if no keypair is associated with the verification key

set_credential_definition_tag_policy(*credential_definition_id: str*, *taggables: Sequence[str] = None*, *retroactive: bool = True*)
Set the tag policy for a given credential definition ID.

Parameters

- **credential_definition_id** – The ID of the credential definition
- **taggables** – A sequence of string values representing attribute names; empty array for none, None for all
- **retroactive** – Whether to apply the policy to previously-stored credentials

sign_message(*message: bytes*, *from_verkey: str*) → bytes
Sign a message using the private key associated with a given verkey.

Parameters

- **message** – Message bytes to sign
- **from_verkey** – The verkey to use to sign

Returns A signature

Raises

- `WalletError` – If the message is not provided
- `WalletError` – If the verkey is not provided
- `WalletError` – If a libindy error occurs

type
Accessor for the wallet type.

unpack_message(*enc_message: bytes) -> (<class 'str'>, <class 'str'>, <class 'str'>*)
Unpack a message.

Parameters **enc_message** – The packed message bytes

Returns (message, from_verkey, to_verkey)

Return type A tuple

Raises

- `WalletError` – If the message is not provided
- `WalletError` – If a libindy error occurs

verify_message(*message: bytes*, *signature: bytes*, *from_verkey: str*) → bool
Verify a signature against the public key of the signer.

Parameters

- **message** – Message to verify
- **signature** – Signature to verify

- **from_verkey** – Verkey to use in verification

Returns True if verified, else False

Raises

- `WalletError` – If the verkey is not provided
- `WalletError` – If the signature is not provided
- `WalletError` – If the message is not provided
- `WalletError` – If a libindy error occurs

aries_cloudagent.wallet.plugin module

Utility for loading Postgres wallet plug-in.

`aries_cloudagent.wallet.plugin.`**`file_ext`**`()`

Determine file extension based on platform.

`aries_cloudagent.wallet.plugin.`**`load_postgres_plugin`**`(`*storage_config*, *storage_creds*, *raise_exc=False*`)`

Load postgres dll and configure postgres wallet.

aries_cloudagent.wallet.provider module

Default wallet provider classes.

class `aries_cloudagent.wallet.provider.`**`WalletProvider`**

Bases: *`aries_cloudagent.config.base.BaseProvider`*

Provider for the default configurable wallet classes.

`WALLET_TYPES = {'basic': 'aries_cloudagent.wallet.basic.BasicWallet', 'indy': 'aries_`

`provide``(`*settings: aries_cloudagent.config.base.BaseSettings*, *injector: aries_cloudagent.config.base.BaseInjector*`)`

Create and open the wallet instance.

aries_cloudagent.wallet.routes module

Wallet admin routes.

class `aries_cloudagent.wallet.routes.`**`DIDListSchema`**`(*`, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*`)`

Bases: `marshmallow.schema.Schema`

Result schema for connection list.

`opts = <marshmallow.schema.SchemaOpts object>`

class `aries_cloudagent.wallet.routes.`**`DIDResultSchema`**`(*`, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*`)`

Bases: `marshmallow.schema.Schema`

Result schema for a DID.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.wallet.routes.**DIDSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Result schema for a DID.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.wallet.routes.**GetTagPolicyResultSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Result schema for tagging policy get request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.wallet.routes.**SetTagPolicyRequestSchema**(*, *only=None*, *exclude=()*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*, *unknown=None*)

Bases: marshmallow.schema.Schema

Request schema for tagging policy set request.

opts = <marshmallow.schema.SchemaOpts object>

aries_cloudagent.wallet.routes.**format_did_info**(*info: aries_cloudagent.wallet.base.DIDInfo*)

Serialize a DIDInfo object.

aries_cloudagent.wallet.routes.**register**(*app: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3d87588>*)

Register routes.

aries_cloudagent.wallet.routes.**wallet_create_did**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3d87588>*)

Request handler for creating a new wallet DID.

Parameters **request** – aiohttp request object

Returns The DID info

aries_cloudagent.wallet.routes.**wallet_did_list**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3d87588>*)

Request handler for searching wallet DIDs.

Parameters **request** – aiohttp request object

Returns The DID list response

`aries_cloudagent.wallet.routes.`**`wallet_get_public_did`**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3d87588>*)

Request handler for fetching the current public DID.

Parameters **request** – aiohttp request object

Returns The DID info

`aries_cloudagent.wallet.routes.`**`wallet_get_tagging_policy`**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3d87588>*)

Request handler for getting the tag policy associated with a cred def.

Parameters **request** – aiohttp request object

Returns A JSON object containing the tagging policy

`aries_cloudagent.wallet.routes.`**`wallet_set_public_did`**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3d87588>*)

Request handler for setting the current public DID.

Parameters **request** – aiohttp request object

Returns The updated DID info

`aries_cloudagent.wallet.routes.`**`wallet_set_tagging_policy`**(*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7fcaf3d87588>*)

Request handler for setting the tag policy associated with a cred def.

Parameters **request** – aiohttp request object

Returns An empty JSON response

aries_cloudagent.wallet.util module

Wallet utility functions.

`aries_cloudagent.wallet.util.`**`b58_to_bytes`**(*val: str*) → bytes

Convert a base 58 string to bytes.

`aries_cloudagent.wallet.util.`**`b64_to_bytes`**(*val: str, urlsafe=False*) → bytes

Convert a base 64 string to bytes.

`aries_cloudagent.wallet.util.`**`b64_to_str`**(*val: str, urlsafe=False, encoding=None*) → str

Convert a base 64 string to string on input encoding (default utf-8).

`aries_cloudagent.wallet.util.`**`bytes_to_b58`**(*val: bytes*) → str

Convert a byte string to base 58.

`aries_cloudagent.wallet.util.`**`bytes_to_b64`**(*val: bytes, urlsafe=False, pad=True*) → str

Convert a byte string to base 64.

`aries_cloudagent.wallet.util.`**`pad`**(*val: str*) → str

Pad base64 values if need be: JWT calls to omit trailing padding.

`aries_cloudagent.wallet.util.`**`set_urlsafe_b64`**(*val: str*, *urlsafe: bool = True*) → str

Set URL safety in base64 encoding.

`aries_cloudagent.wallet.util.`**`str_to_b64`**(*val: str*, *urlsafe=False*, *encoding=None*, *pad=True*) → str

Convert a string to base64 string on input encoding (default utf-8).

`aries_cloudagent.wallet.util.`**`unpad`**(*val: str*) → str

Remove padding from base64 values if need be.

1.2 Submodules

1.3 aries_cloudagent.version module

Library version information.

CHAPTER 2

aries_cloudagent.connections package

2.1 Subpackages

2.1.1 aries_cloudagent.connections.models package

Subpackages

aries_cloudagent.connections.models.diddoc package

DID Document model support.

class `aries_cloudagent.connections.models.diddoc.`**`DIDDoc`**(*did: str = None*)

Bases: `object`

DID document, grouping a DID with verification keys and services.

Retains DIDs as raw values (orientated toward indy-facing operations), everything else as URIs (oriented toward W3C-facing operations).

`CONTEXT = 'https://w3id.org/did/v1'`

`add_service_pubkeys`(*service: dict, tags: Union[Sequence[str], str]*) → List[aries_cloudagent.connections.models.diddoc.publickey.PublicKey]

Add public keys specified in service. Return public keys so discovered.

Parameters

- **service** – service from DID document
- **tags** – potential tags marking public keys of type of interest (the standard is still coalescing)

Raises ValueError – for public key reference not present in DID document.

Returns: list of public keys from the document service specification

authnkey
Accessor for public keys marked as authentication keys, by identifier.

classmethod deserialize(*did_doc: dict*) → aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc
Construct DIDDoc object from dict representation.

Parameters **did_doc** – DIDDoc dict representation

Raises ValueError – for bad DID or missing mandatory item.

Returns: DIDDoc from input json

did
Accessor for DID.

classmethod from_json(*did_doc_json: str*) → aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc
Construct DIDDoc object from json representation.

Parameters **did_doc_json** – DIDDoc json representation

Returns: DIDDoc from input json

pubkey
Accessor for public keys by identifier.

serialize() → str
Dump current object to a JSON-compatible dictionary.

Returns dict representation of current DIDDoc

service
Accessor for services by identifier.

set(*item: Union[aries_cloudagent.connections.models.diddoc.service.Service, aries_cloudagent.connections.models.diddoc.publickey.PublicKey]*) → aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc
Add or replace service or public key; return current DIDDoc.

Raises ValueError – if input item is neither service nor public key.

Parameters **item** – service or public key to set

Returns: the current DIDDoc

to_json() → str
Dump current object as json (JSON-LD).

Returns json representation of current DIDDoc

class aries_cloudagent.connections.models.diddoc.**LinkedDataKeySpec**(*ver_type, authn_type, specifier*)

Bases: tuple

authn_type
Alias for field number 1

specifier
Alias for field number 2

ver_type
Alias for field number 0

class aries_cloudagent.connections.models.diddoc.**PublicKey**(*did: str, ident: str, value: str, pk_type: aries_cloudagent.connections.models.diddoc.p = None, controller: str = None, authn: bool = False*)

Bases: object

Public key specification to embed in DID document.

Retains DIDs as raw values (orientated toward indy-facing operations), everything else as URIs (oriented toward W3C-facing operations).

authn
Accessor for the authentication marker.

Returns: whether public key is marked as having DID authentication privilege

controller
Accessor for the controller DID.

did
Accessor for the DID.

id
Accessor for the public key identifier.

to_dict() → dict
Return dict representation of public key to embed in DID document.

type
Accessor for the public key type.

value
Accessor for the public key value.

class aries_cloudagent.connections.models.diddoc.**PublicKeyType**

Bases: enum.Enum

Class encapsulating public key types.

ED25519_SIG_2018 = LinkedDataKeySpec(ver_type='Ed25519VerificationKey2018', authn_type=

EDDSA_SA_SIG_SECP256K1 = LinkedDataKeySpec(ver_type='Secp256k1VerificationKey2018', au

RSA_SIG_2018 = LinkedDataKeySpec(ver_type='RsaVerificationKey2018', authn_type='RsaSig

authn_type
Accessor for the authentication type identifier.

get = <function PublicKeyType.get>

specification(*val: str*) → str
Return specifier and input value for use in public key specification.

Parameters **val** – value of public key

Returns: dict mapping applicable specifier to input value

specifier
Accessor for the value specifier.

ver_type
Accessor for the verification type identifier.

class aries_cloudagent.connections.models.diddoc.**Service**(*did: str, ident: str, typ: str, recip_keys: Union[Sequence[T_co], aries_cloudagent.connections.models.diddoc.publ routing_keys: Union[Sequence[T_co], aries_cloudagent.connections.models.diddoc.publ endpoint: str, priority: int = 0*)

Bases: object

Service specification to embed in DID document.

Retains DIDs as raw values (orientated toward indy-facing operations), everything else as URIs (oriented toward W3C-facing operations).

did
Accessor for the DID value.

endpoint
Accessor for the endpoint value.

id
Accessor for the service identifier.

priority
Accessor for the priority value.

recip_keys
Accessor for the recipient keys.

routing_keys
Accessor for the routing keys.

to_dict() → dict
Return dict representation of service to embed in DID document.

type
Accessor for the service type.

Submodules

aries_cloudagent.connections.models.diddoc.diddoc module

DID Document classes.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

class aries_cloudagent.connections.models.diddoc.diddoc.**DIDDoc**(*did: str = None*)

Bases: object

DID document, grouping a DID with verification keys and services.

Retains DIDs as raw values (orientated toward indy-facing operations), everything else as URIs (oriented toward W3C-facing operations).

CONTEXT = 'https://w3id.org/did/v1'

add_service_pubkeys(*service: dict, tags: Union[Sequence[str], str]*) → List[aries_cloudagent.connections.models.diddoc.publickey.PublicKey]

Add public keys specified in service. Return public keys so discovered.

Parameters

- **service** – service from DID document
- **tags** – potential tags marking public keys of type of interest (the standard is still coalescing)

Raises ValueError – for public key reference not present in DID document.

Returns: list of public keys from the document service specification

authnkey

Accessor for public keys marked as authentication keys, by identifier.

classmethod deserialize(*did_doc: dict*) → aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc

Construct DIDDoc object from dict representation.

Parameters **did_doc** – DIDDoc dict representation

Raises ValueError – for bad DID or missing mandatory item.

Returns: DIDDoc from input json

did

Accessor for DID.

classmethod from_json(*did_doc_json: str*) → aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc

Construct DIDDoc object from json representation.

Parameters **did_doc_json** – DIDDoc json representation

Returns: DIDDoc from input json

pubkey

Accessor for public keys by identifier.

serialize() → str

Dump current object to a JSON-compatible dictionary.

Returns dict representation of current DIDDoc

service

Accessor for services by identifier.

set(*item: Union[aries_cloudagent.connections.models.diddoc.service.Service, aries_cloudagent.connections.models.diddoc.publickey.PublicKey]*) → aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc

Add or replace service or public key; return current DIDDoc.

Raises `ValueError` – if input item is neither service nor public key.

Parameters **item** – service or public key to set

Returns: the current DIDDoc

to_json() → str

Dump current object as json (JSON-LD).

Returns json representation of current DIDDoc

aries_cloudagent.connections.models.diddoc.publickey module

DID Document Public Key classes.

class aries_cloudagent.connections.models.diddoc.publickey.**LinkedDataKeySpec**(*ver_type, authn_type, specifier*)

Bases: `tuple`

authn_type

Alias for field number 1

specifier

Alias for field number 2

ver_type

Alias for field number 0

class aries_cloudagent.connections.models.diddoc.publickey.**PublicKey**(*did: str, ident: str, value: str, pk_type: aries_cloudagent.connections.m = None, controller: str = None, authn: bool = False*)

Bases: object

Public key specification to embed in DID document.

Retains DIDs as raw values (orientated toward indy-facing operations), everything else as URIs (oriented toward W3C-facing operations).

authn
Accessor for the authentication marker.

Returns: whether public key is marked as having DID authentication privilege

controller
Accessor for the controller DID.

did
Accessor for the DID.

id
Accessor for the public key identifier.

to_dict() → dict
Return dict representation of public key to embed in DID document.

type
Accessor for the public key type.

value
Accessor for the public key value.

class aries_cloudagent.connections.models.diddoc.publickey.**PublicKeyType**

Bases: enum.Enum

Class encapsulating public key types.

ED25519_SIG_2018 = LinkedDataKeySpec(ver_type='Ed25519VerificationKey2018', authn_type=

EDDSA_SA_SIG_SECP256K1 = LinkedDataKeySpec(ver_type='Secp256k1VerificationKey2018', au

RSA_SIG_2018 = LinkedDataKeySpec(ver_type='RsaVerificationKey2018', authn_type='RsaSig

authn_type
Accessor for the authentication type identifier.

get = <function PublicKeyType.get>

specification(*val: str*) → str
Return specifier and input value for use in public key specification.

Parameters **val** – value of public key

Returns: dict mapping applicable specifier to input value

specifier
Accessor for the value specifier.

ver_type
Accessor for the verification type identifier.

aries_cloudagent.connections.models.diddoc.service module

DID Document Service classes.

class aries_cloudagent.connections.models.diddoc.service.**Service**(*did: str, ident: str, typ: str, recip_keys: Union[Sequence[T_co], aries_cloudagent.connections.models. routing_keys: Union[Sequence[T_co], aries_cloudagent.connections.models. endpoint: str, priority: int = 0*)

Bases: object

Service specification to embed in DID document.

Retains DIDs as raw values (orientated toward indy-facing operations), everything else as URIs (oriented toward W3C-facing operations).

did
Accessor for the DID value.

endpoint
Accessor for the endpoint value.

id
Accessor for the service identifier.

priority
Accessor for the priority value.

recip_keys
Accessor for the recipient keys.

routing_keys
Accessor for the routing keys.

to_dict() → dict
Return dict representation of service to embed in DID document.

type
Accessor for the service type.

aries_cloudagent.connections.models.diddoc.util module

DIDDoc utility methods.

aries_cloudagent.connections.models.diddoc.util.**canon_did**(*uri: str*) → str
Convert a URI into a DID if need be, left-stripping 'did:sov:' if present.

Parameters **uri** – input URI or DID

Raises ValueError – for invalid input.

aries_cloudagent.connections.models.diddoc.util.**canon_ref**(*did: str*, *ref: str*, *delimiter: str = None*)
Given a reference in a DID document, return it in its canonical form of a URI.

Parameters

- **did** – DID acting as the identifier of the DID document
- **ref** – reference to canonicalize, either a DID or a fragment pointing to a location in the DID doc
- **delimiter** – delimiter character marking fragment (default '#') or introducing identifier (';') against DID resource

aries_cloudagent.connections.models.diddoc.util.**ok_did**(*token: str*) → bool
Whether input token looks like a valid decentralized identifier.

Parameters **token** – candidate string

Returns: whether input token looks like a valid schema identifier

aries_cloudagent.connections.models.diddoc.util.**resource**(*ref: str*, *delimiter: str = None*) → str
Extract the resource for an identifier.

Given a (URI) reference, return up to its delimiter (exclusively), or all of it if there is none.

Parameters

- **ref** – reference
- **delimiter** – delimiter character (default None maps to '#', or ';' introduces identifiers)

Submodules

aries_cloudagent.connections.models.connection_record module

Handle connection information interface with non-secrets storage.

class aries_cloudagent.connections.models.connection_record.**ConnectionRecord**(*, *connection_id: str = None, my_did: str = None, their_did: str = None, their_label: str = None, their_role: str = None, initiator: str = None, invitation_key: str = None, request_id: str = None, state: str = None, inbound_connection_i*
str = None, error_msg: str = None, routing_state: str = None,

Bases: *aries_cloudagent.messaging.models.base_record.BaseRecord*

Represents a single pairwise connection.

ACCEPT_AUTO = 'auto'

ACCEPT_MANUAL = 'manual'

CACHE_ENABLED = True

DIRECTION_RECEIVED = 'received'

DIRECTION_SENT = 'sent'

INITIATOR_EXTERNAL = 'external'

INITIATOR_MULTIUSE = 'multiuse'

INITIATOR_SELF = 'self'

INVITATION_MODE_MULTI = 'multi'

INVITATION_MODE_ONCE = 'once'

INVITATION_MODE_STATIC = 'static'

LOG_STATE_FLAG = 'debug.connections'

class Meta

Bases: object

ConnectionRecord metadata.

schema_class = 'ConnectionRecordSchema'

RECORD_ID_NAME = 'connection_id'

RECORD_TYPE = 'connection'

RECORD_TYPE_INVITATION = 'connection_invitation'

RECORD_TYPE_REQUEST = 'connection_request'

ROUTING_STATE_ACTIVE = 'active'

ROUTING_STATE_ERROR = 'error'

ROUTING_STATE_NONE = 'none'

ROUTING_STATE_REQUEST = 'request'

STATE_ACTIVE = 'active'

STATE_ERROR = 'error'

STATE_INACTIVE = 'inactive'

STATE_INIT = 'init'

STATE_INVITATION = 'invitation'

STATE_REQUEST = 'request'

STATE_RESPONSE = 'response'

TAG_NAMES = {'invitation_key', 'my_did', 'request_id', 'their_did'}

WEBHOOK_TOPIC = 'connections'

attach_invitation(*context: aries_cloudagent.config.injection_context.InjectionContext, invitation: aries_cloudagent.protocols.connections.v1_0.messages.connection_invitation.ConnectionInvitati*

Persist the related connection invitation to storage.

Parameters

- **context** – The injection context to use
- **invitation** – The invitation to relate to this connection record

attach_request(*context: aries_cloudagent.config.injection_context.InjectionContext, request: aries_cloudagent.protocols.connections.v1_0.messages.connection_request.ConnectionRequest*)

Persist the related connection request to storage.

Parameters

- **context** – The injection context to use
- **request** – The request to relate to this connection record

connection_id

Accessor for the ID associated with this connection.

is_multiuse_invitation

Accessor for multi use invitation mode.

is_ready

Accessor for connection readiness.

post_save(*context: aries_cloudagent.config.injection_context.InjectionContext, *args, **kwargs*)

Perform post-save actions.

Parameters **context** – The injection context to use

record_value

Accessor to for the JSON record value properties for this connection.

classmethod retrieve_by_did(*context: aries_cloudagent.config.injection_context.InjectionContext, their_did: str = None, my_did: str = None, initiator: str = None*) → aries_cloudagent.connections.models.connection_record.ConnectionRecord

Retrieve a connection record by target DID.

Parameters

- **context** – The injection context to use
- **their_did** – The target DID to filter by
- **my_did** – One of our DIDs to filter by
- **initiator** – Filter connections by the initiator value

classmethod retrieve_by_invitation_key(*context: aries_cloudagent.config.injection_context.InjectionContext, invitation_key: str, initiator: str = None*) → aries_cloudagent.connections.models.connection_record.ConnectionRe

Retrieve a connection record by invitation key.

Parameters

- **context** – The injection context to use
- **invitation_key** – The key on the originating invitation
- **initiator** – Filter by the initiator value

classmethod retrieve_by_request_id(*context: aries_cloudagent.config.injection_context.InjectionContext*, *request_id: str*) → aries_cloudagent.connections.models.connection_record.ConnectionRecord

Retrieve a connection record from our previous request ID.

Parameters

- **context** – The injection context to use
- **request_id** – The ID of the originating connection request

retrieve_invitation(*context: aries_cloudagent.config.injection_context.InjectionContext*) → aries_cloudagent.protocols.connections.v1_0.messages.connection_invitation.ConnectionInvitation

Retrieve the related connection invitation.

Parameters **context** – The injection context to use

retrieve_request(*context: aries_cloudagent.config.injection_context.InjectionContext*) → aries_cloudagent.protocols.connections.v1_0.messages.connection_request.ConnectionRequest

Retrieve the related connection invitation.

Parameters **context** – The injection context to use

class aries_cloudagent.connections.models.connection_record.**ConnectionRecordSchema**(**args*, ***kwargs*)

Bases: *aries_cloudagent.messaging.models.base_record.BaseRecordSchema*

Schema to allow serialization/deserialization of connection records.

class Meta

Bases: object

ConnectionRecordSchema metadata.

model_class

alias of *ConnectionRecord*

accept = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<OneOf

alias = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, r

connection_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=

error_msg = <fields.String(default=<marshmallow.missing>, attribute=None, validate=Non

inbound_connection_id = <fields.String(default=<marshmallow.missing>, attribute=None,

initiator = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<On

invitation_key = <fields.String(default=<marshmallow.missing>, attribute=None, validat

invitation_mode = <fields.String(default=<marshmallow.missing>, attribute=None, valida

my_did = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<aries_

request_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=No

routing_state = <fields.String(default=<marshmallow.missing>, attribute=None, validate=

their_did = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<ar

their_label = <fields.String(default=<marshmallow.missing>, attribute=None, validate=N

their_role = <fields.String(default=<marshmallow.missing>, attribute=None, validate=No

aries_cloudagent.connections.models.connection_target module

Record used to handle routing of messages to another agent.

class aries_cloudagent.connections.models.connection_target.**ConnectionTarget**(*, *did: str = None, endpoint: str = None, label: str = None, recipient_keys: Sequence[str] = None, routing_keys: Sequence[str] = None, sender_key: str = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Record used to handle routing of messages to another agent.

class Meta

Bases: object

ConnectionTarget metadata.

schema_class = 'ConnectionTargetSchema'

class aries_cloudagent.connections.models.connection_target.**ConnectionTargetSchema**(**args, **kwargs*)

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

ConnectionTarget schema.

class Meta

Bases: object

ConnectionTargetSchema metadata.

model_class
alias of *ConnectionTarget*

did
Used by autodoc_mock_imports.

endpoint
Used by autodoc_mock_imports.

label
Used by autodoc_mock_imports.

recipient_keys
Used by autodoc_mock_imports.

routing_keys
Used by autodoc_mock_imports.

sender_key
Used by autodoc_mock_imports.

CHAPTER 3

aries_cloudagent.protocols package

3.1 Subpackages

3.1.1 aries_cloudagent.protocols.actionmenu package

Subpackages

aries_cloudagent.protocols.actionmenu.handlers package

Submodules

aries_cloudagent.protocols.actionmenu.handlers.menu_handler module

aries_cloudagent.protocols.actionmenu.handlers.menu_request_handler module

aries_cloudagent.protocols.actionmenu.handlers.perform_handler module

aries_cloudagent.protocols.actionmenu.messages package

Submodules

aries_cloudagent.protocols.actionmenu.messages.menu module

aries_cloudagent.protocols.actionmenu.messages.menu_request module

aries_cloudagent.protocols.actionmenu.messages.perform module

aries_cloudagent.protocols.actionmenu.models package

Submodules

aries_cloudagent.protocols.actionmenu.models.menu_form module

aries_cloudagent.protocols.actionmenu.models.menu_form_param module

aries_cloudagent.protocols.actionmenu.models.menu_option module

Submodules

aries_cloudagent.protocols.actionmenu.base_service module

aries_cloudagent.protocols.actionmenu.controller module

aries_cloudagent.protocols.actionmenu.driver_service module

aries_cloudagent.protocols.actionmenu.message_types module

aries_cloudagent.protocols.actionmenu.routes module

aries_cloudagent.protocols.actionmenu.util module

3.1.2 aries_cloudagent.protocols.basicmessage package

Subpackages

aries_cloudagent.protocols.basicmessage.handlers package

Submodules

aries_cloudagent.protocols.basicmessage.handlers.basicmessage_handler module

aries_cloudagent.protocols.basicmessage.messages package

Submodules

aries_cloudagent.protocols.basicmessage.messages.basicmessage module

Submodules

aries_cloudagent.protocols.basicmessage.message_types module

aries_cloudagent.protocols.basicmessage.routes module

3.1.3 aries_cloudagent.protocols.connections package

Subpackages

aries_cloudagent.protocols.connections.handlers package

Submodules

aries_cloudagent.protocols.connections.handlers.connection_invitation_handler module

aries_cloudagent.protocols.connections.handlers.connection_request_handler module

aries_cloudagent.protocols.connections.handlers.connection_response_handler module

aries_cloudagent.protocols.connections.messages package

Submodules

aries_cloudagent.protocols.connections.messages.connection_invitation module

aries_cloudagent.protocols.connections.messages.connection_request module

aries_cloudagent.protocols.connections.messages.connection_response module

aries_cloudagent.protocols.connections.messages.problem_report module

aries_cloudagent.protocols.connections.models package

Submodules

aries_cloudagent.protocols.connections.models.connection_detail module

Submodules

aries_cloudagent.protocols.connections.manager module

aries_cloudagent.protocols.connections.message_types module

aries_cloudagent.protocols.connections.routes module

3.1.4 aries_cloudagent.protocols.credentials package

Subpackages

aries_cloudagent.protocols.credentials.handlers package

Submodules

aries_cloudagent.protocols.credentials.handlers.credential_issue_handler module

aries_cloudagent.protocols.credentials.handlers.credential_offer_handler module

aries_cloudagent.protocols.credentials.handlers.credential_request_handler module

aries_cloudagent.protocols.credentials.handlers.credential_stored_handler module

aries_cloudagent.protocols.credentials.messages package

Submodules

aries_cloudagent.protocols.credentials.messages.credential_issue module

aries_cloudagent.protocols.credentials.messages.credential_offer module

aries_cloudagent.protocols.credentials.messages.credential_request module

aries_cloudagent.protocols.credentials.messages.credential_stored module

aries_cloudagent.protocols.credentials.models package

Submodules

aries_cloudagent.protocols.credentials.models.credential_exchange module

Submodules

aries_cloudagent.protocols.credentials.manager module

aries_cloudagent.protocols.credentials.message_types module

aries_cloudagent.protocols.credentials.routes module

3.1.5 aries_cloudagent.protocols.discovery package

Subpackages

aries_cloudagent.protocols.discovery.handlers package

Submodules

aries_cloudagent.protocols.discovery.handlers.disclose_handler module

aries_cloudagent.protocols.discovery.handlers.query_handler module

aries_cloudagent.protocols.discovery.messages package

Submodules

aries_cloudagent.protocols.discovery.messages.disclose module

aries_cloudagent.protocols.discovery.messages.query module

Submodules

aries_cloudagent.protocols.discovery.message_types module

aries_cloudagent.protocols.discovery.routes module

3.1.6 aries_cloudagent.protocols.introduction package

Subpackages

aries_cloudagent.protocols.introduction.handlers package

Submodules

aries_cloudagent.protocols.introduction.handlers.forward_invitation_handler module

aries_cloudagent.protocols.introduction.handlers.invitation_handler module

aries_cloudagent.protocols.introduction.handlers.invitation_request_handler module

aries_cloudagent.protocols.introduction.messages package

Submodules

aries_cloudagent.protocols.introduction.messages.forward_invitation module

aries_cloudagent.protocols.introduction.messages.invitation module

aries_cloudagent.protocols.introduction.messages.invitation_request module

Submodules

aries_cloudagent.protocols.introduction.base_service module

aries_cloudagent.protocols.introduction.demo_service module

aries_cloudagent.protocols.introduction.message_types module

aries_cloudagent.protocols.introduction.routes module

3.1.7 aries_cloudagent.protocols.issue_credential package

Subpackages

aries_cloudagent.protocols.issue_credential.v1_0 package

Subpackages

aries_cloudagent.protocols.issue_credential.v1_0.handlers package

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_ack_handler module

Credential ack message handler.

class aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_ack_handler.**Cre**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential acks.

handle(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Message handler logic for credential acks.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_issue_handler module

Credential issue message handler.

class aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_issue_handler.**C**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential offers.

handle(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Message handler logic for credential offers.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_offer_handler module

Credential offer message handler.

class aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_offer_handler.**C**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential offers.

handle(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Message handler logic for credential offers.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_proposal_handler module

Credential proposal message handler.

class aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_proposal_handler

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential proposals.

handle(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Message handler logic for credential proposals.

Parameters

- **context** – proposal context
- **responder** – responder callback

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_request_handler module

Credential request message handler.

class aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_request_handler

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential requests.

handle(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Message handler logic for credential requests.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.issue_credential.v1_0.messages package

Subpackages

aries_cloudagent.protocols.issue_credential.v1_0.messages.inner package

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.messages.inner.credential_preview module

A credential preview inner object.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.inner.credential_preview.Cr

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing a preview of an attibute.

class Meta

Bases: object

Attribute preview metadata.

schema_class = 'CredAttrSpecSchema'

b64_decoded_value() → str

Value, base64-decoded if applicable.

static list_plain(*plain: dict*)

Return a list of *CredAttrSpec* without MIME types from names/values.

Parameters **plain** – dict mapping names to values

Returns List of CredAttrSpecs with no MIME types

class aries_cloudagent.protocols.issue_credential.v1_0.messages.inner.credential_preview.Cr

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Attribute preview schema.

class Meta

Bases: object

Attribute preview schema metadata.

model_class

alias of *CredAttrSpec*

mime_type = <fields.String(default=<marshmallow.missing>, attribute=None, validate=Non

name = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, re

value = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, r

class aries_cloudagent.protocols.issue_credential.v1_0.messages.inner.credential_preview.Cr

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing a credential preview inner object.

class Meta

Bases: object

Credential preview metadata.

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/issue-credential/1.0/credential

schema_class = 'CredentialPreviewSchema'

attr_dict (*decode: bool = False*)

Return name:value pair per attribute.

Parameters **decode** – whether first to decode attributes with MIME type

mime_types()

Return per-attribute mapping from name to MIME type.

Return empty dict if no attribute has MIME type.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.inner.credential_preview.**Cr**

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Credential preview schema.

class Meta

Bases: object

Credential preview schema metadata.

model_class

alias of *CredentialPreview*

attributes = <fields.Nested(default=<marshmallow.missing>, attribute=None, validate=No

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_ack module

A credential ack message.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_ack.**CredentialAc**

Bases: *aries_cloudagent.messaging.ack.message.Ack*

Class representing a credential ack message.

class Meta

Bases: object

Credential metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v1_0.handlers.credenti

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/issue-credential/1.0/ack'

schema_class = 'CredentialAckSchema'

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_ack.**CredentialAc**

Bases: *aries_cloudagent.messaging.ack.message.AckSchema*

Credential ack schema.

class Meta

Bases: object

Schema metadata.

model_class

alias of *CredentialAck*

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_issue module

A credential content message.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_issue.**Credentia**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a credential.

class Meta

Bases: object

Credential metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v1_0.handlers.credenti

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/issue-credential/1.0/issue-cred

schema_class = 'CredentialIssueSchema'

indy_credential(*index: int = 0*)

Retrieve and decode indy credential from attachment.

Parameters **index** – ordinal in attachment list to decode and return (typically, list has length 1)

classmethod wrap_indy_credential(*indy_cred: dict*) → aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator

Convert an indy credential offer to an attachment decorator.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_issue.**Credentia**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessageSchema*

Credential schema.

class Meta

Bases: object

Credential schema metadata.

model_class

alias of *CredentialIssue*

comment = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None,

credentials_attach = <fields.Nested(default=<marshmallow.missing>, attribute=None, val

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_offer module

A credential offer content message.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_offer.**Credentia**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a credential offer.

class Meta

Bases: object

CredentialOffer metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v1_0.handlers.credenti

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/issue-credential/1.0/offer-cred

schema_class = 'CredentialOfferSchema'

indy_offer(*index: int = 0*) → dict

Retrieve and decode indy offer from attachment.

Parameters **index** – ordinal in attachment list to decode and return (typically, list has length 1)

classmethod wrap_indy_offer(*indy_offer: dict*) → aries_cloudagent.messaging.decorators.attach_decorator.AttachDec

Convert an indy credential offer to an attachment decorator.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_offer.**Credentia**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessageSchema*

Credential offer schema.

class Meta

Bases: object

Credential offer schema metadata.

model_class

alias of *CredentialOffer*

comment = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None,

credential_preview = <fields.Nested(default=<marshmallow.missing>, attribute=None, val

offers_attach = <fields.Nested(default=<marshmallow.missing>, attribute=None, validate=

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_proposal module

A credential proposal content message.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_proposal.**Credent**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a credential proposal.

class Meta

Bases: object

CredentialProposal metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v1_0.handlers.credenti

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/issue-credential/1.0/propose-cr

schema_class = 'CredentialProposalSchema'

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_proposal.**Credent**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessageSchema*

Credential proposal schema.

class Meta

Bases: object

Credential proposal schema metadata.

model_class

alias of *CredentialProposal*

comment = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None,

cred_def_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<a

credential_proposal = <fields.Nested(default=<marshmallow.missing>, attribute=None, va

issuer_did = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<a

schema_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<ar

schema_issuer_did = <fields.String(default=<marshmallow.missing>, attribute=None, vali

schema_name = <fields.String(default=<marshmallow.missing>, attribute=None, validate=N

schema_version = <fields.String(default=<marshmallow.missing>, attribute=None, validat

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_request module

A credential request content message.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_request.**Credent**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a credential request.

class Meta

Bases: object

CredentialRequest metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v1_0.handlers.credenti

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/issue-credential/1.0/request-cr

schema_class = 'CredentialRequestSchema'

indy_cred_req(*index: int = 0*)

Retrieve and decode indy credential request from attachment.

Parameters index – ordinal in attachment list to decode and return (typically, list has length 1)

classmethod wrap_indy_cred_req(*indy_cred_req: dict*) → aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator

Convert an indy credential request to an attachment decorator.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_request.**Credent**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessageSchema*

Credential request schema.

class Meta

Bases: object

Credential request schema metadata.

model_class

alias of *CredentialRequest*

comment = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None,

requests_attach = <fields.Nested(default=<marshmallow.missing>, attribute=None, valida

aries_cloudagent.protocols.issue_credential.v1_0.models package

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange module

Aries#0036 v1.0 credential exchange information with non-secrets storage.

class aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.**V10Creden**

Bases: *aries_cloudagent.messaging.models.base_record.BaseExchangeRecord*

Represents an Aries#0036 credential exchange.

INITIATOR_EXTERNAL = 'external'

INITIATOR_SELF = 'self'

class Meta

Bases: object

CredentialExchange metadata.

schema_class = 'V10CredentialExchangeSchema'

RECORD_ID_NAME = 'credential_exchange_id'

RECORD_TYPE = 'credential_exchange_v10'

ROLE_HOLDER = 'holder'

ROLE_ISSUER = 'issuer'

STATE_ACKED = 'credential_acked'

STATE_CREDENTIAL_RECEIVED = 'credential_received'

STATE_ISSUED = 'credential_issued'

STATE_OFFER_RECEIVED = 'offer_received'

STATE_OFFER_SENT = 'offer_sent'

STATE_PROPOSAL_RECEIVED = 'proposal_received'

STATE_PROPOSAL_SENT = 'proposal_sent'

STATE_REQUEST_RECEIVED = 'request_received'

STATE_REQUEST_SENT = 'request_sent'

TAG_NAMES = {'thread_id'}

WEBHOOK_TOPIC = 'issue_credential'

credential_exchange_id

Accessor for the ID associated with this exchange.

record_value

Accessor for the JSON record value generated for this credential exchange.

classmethod retrieve_by_connection_and_thread(*context: aries_cloudagent.config.injection_context.InjectionContext, connection_id: str, thread_id: str*) → aries_cloudagent.protocols.issue_credential.v1_0.models.cre

Retrieve a credential exchange record by connection and thread ID.

class aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.**V10Creden**

Bases: *aries_cloudagent.messaging.models.base_record.BaseExchangeSchema*

Schema to allow serialization/deserialization of credential exchange records.

class Meta

Bases: object

V10CredentialExchangeSchema metadata.

model_class
alias of *V10CredentialExchange*

auto_issue = <fields.Boolean(default=<marshmallow.missing>, attribute=None, validate=N

auto_offer = <fields.Boolean(default=<marshmallow.missing>, attribute=None, validate=N

auto_remove = <fields.Boolean(default=True, attribute=None, validate=None, required=Fa

connection_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate

credential = <fields.Dict(default=<marshmallow.missing>, attribute=None, validate=None

credential_definition_id = <fields.String(default=<marshmallow.missing>, attribute=Non

credential_exchange_id = <fields.String(default=<marshmallow.missing>, attribute=None,

credential_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate

credential_offer = <fields.Dict(default=<marshmallow.missing>, attribute=None, validat

credential_proposal_dict = <fields.Dict(default=<marshmallow.missing>, attribute=None,

credential_request = <fields.Dict(default=<marshmallow.missing>, attribute=None, valid

credential_request_metadata = <fields.Dict(default=<marshmallow.missing>, attribute=No

error_msg = <fields.String(default=<marshmallow.missing>, attribute=None, validate=Non

initiator = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<On

parent_thread_id = <fields.String(default=<marshmallow.missing>, attribute=None, valid

raw_credential = <fields.Dict(default=<marshmallow.missing>, attribute=None, validate=

revoc_reg_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=

revocation_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate

role = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<OneOf(c

schema_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<ar

state = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, r

thread_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=Non

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.manager module

Classes to manage credentials.

class aries_cloudagent.protocols.issue_credential.v1_0.manager.**CredentialManager**(*context: aries_cloudag*

Bases: object

Class for managing credentials.

context
Accessor for the current request context.

Returns The request context for this connection

create_offer(*credential_exchange_record: aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.V*
comment: str = None) → Tuple[aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchang
aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_offer.CredentialOffer]

Create a credential offer, update credential exchange record.

Parameters

- **credential_exchange_record** – Credential exchange to create offer for
- **comment** – optional human-readable comment to set in offer message

Returns A tuple (credential exchange record, credential offer message)

create_proposal(*connection_id: str, *, auto_offer: bool = None, auto_remove: bool = None, comment: str = None, credential_preview: aries_cloudagent.protocols.issue_credential.v1_0.messages.inner.credential_preview.CredentialPreview = None, schema_id: str = None, schema_issuer_did: str = None, schema_name: str = None, schema_version: str = None, cred_def_id: str = None, issuer_did: str = None, trace: bool = False*) → aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.V10CredentialExchange

Create a credential proposal.

Parameters

- **connection_id** – Connection to create proposal for
- **auto_offer** – Should this proposal request automatically be handled to offer a credential
- **auto_remove** – Should the record be automatically removed on completion
- **comment** – Optional human-readable comment to include in proposal
- **credential_preview** – The credential preview to use to create the credential proposal
- **schema_id** – Schema id for credential proposal
- **schema_issuer_did** – Schema issuer DID for credential proposal
- **schema_name** – Schema name for credential proposal
- **schema_version** – Schema version for credential proposal
- **cred_def_id** – Credential definition id for credential proposal
- **issuer_did** – Issuer DID for credential proposal

Returns Resulting credential exchange record including credential proposal

create_request(*credential_exchange_record: aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchang*
holder_did: str) → Tuple[aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.V
aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_request.CredentialRequest]

Create a credential request.

Parameters

- **credential_exchange_record** – Credential exchange record for which to create request
- **holder_did** – holder DID

Returns A tuple (credential exchange record, credential request message)

issue_credential(*credential_exchange_record: aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exch*
**, comment: str = None, credential_values: dict*) → Tuple[aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.V10CredentialExcha
aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_issue.CredentialIssue]

Issue a credential.

Parameters

- **credential_exchange_record** – The credential exchange record for which to issue a credential
- **comment** – optional human-readable comment pertaining to credential issue
- **credential_values** – dict of credential attribute {name: value} pairs

Returns (Updated credential exchange record, credential message)

Return type Tuple

prepare_send(*connection_id: str, credential_proposal: aries_cloudagent.protocols.issue_credential.v1_0.messages.credenti*
auto_remove: bool = None) → Tuple[aries_cloudagent.protocols.issue_credential.v1_0.models.credential_ex
aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_offer.CredentialOffer]

Set up a new credential exchange for an automated send.

Parameters

- **connection_id** – Connection to create offer for
- **credential_proposal** – The credential proposal with preview on attribute values to use if auto_issue is enabled
- **auto_remove** – Flag to automatically remove the record on completion

Returns A tuple of the new credential exchange record and credential offer message

publish_pending_revocations() → Mapping[str, Sequence[str]]

Publish pending revocations to the ledger.

Returns: mapping from each revocation registry id to its cred rev ids published.

receive_credential() → aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.V10CredentialEx

Receive a credential from an issuer.

Hold in storage potentially to be processed by controller before storing.

Returns Credential exchange record, retrieved and updated

receive_credential_ack() → aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.V10Creder

Receive credential ack from holder.

Returns credential exchange record, retrieved and updated

receive_offer() → aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.V10CredentialExchange

Receive a credential offer.

Returns The credential exchange record, updated

receive_proposal() → aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.V10CredentialExch

Receive a credential proposal from message in context on manager creation.

Returns The resulting credential exchange record, created

receive_request()

Receive a credential request.

Parameters **credential_request_message** – Credential request to receive

Returns credential exchange record, retrieved and updated

revoke_credential(*rev_reg_id: str*, *cred_rev_id: str*, *publish: bool = False*)

Revoke a previously-issued credential.

Optionally, publish the corresponding revocation registry delta to the ledger.

Parameters

- **rev_reg_id** – revocation registry id
- **cred_rev_id** – credential revocation id
- **publish** – whether to publish the resulting revocation registry delta

store_credential(*credential_exchange_record: aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exch*
credential_id: str = None) → Tuple[aries_cloudagent.protocols.issue_credential.v1_0.models.credentia
aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_ack.CredentialAck]

Store a credential in holder wallet; send ack to issuer.

Parameters

- **credential_exchange_record** – credential exchange record with credential to store and ack
- **credential_id** – optional credential identifier to override default on storage

Returns (Updated credential exchange record, credential ack message)

Return type Tuple

exception aries_cloudagent.protocols.issue_credential.v1_0.manager.**CredentialManagerError**(

Bases: *aries_cloudagent.core.error.BaseError*

Credential error.

aries_cloudagent.protocols.issue_credential.v1_0.message_types module

Message and inner object type identifiers for Connections.

aries_cloudagent.protocols.issue_credential.v1_0.routes module

Credential exchange admin routes.

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10AttributeMimeTypesResultSc**

Bases: marshmallow.schema.Schema

Result schema for credential attribute MIME types by credential definition.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10CredentialExchangeListResu**

Bases: marshmallow.schema.Schema

Result schema for Aries#0036 v1.0 credential exchange query.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10CredentialIssueRequestSche**

Bases: marshmallow.schema.Schema

Request schema for sending credential issue admin message.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10CredentialOfferRequestSche**

Bases: aries_cloudagent.utils.tracing.AdminAPIMessageTracingSchema

Request schema for sending credential offer admin message.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10CredentialProblemReportRe**

Bases: marshmallow.schema.Schema

Request schema for sending problem report.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10CredentialProposalRequestM**

Bases: *aries_cloudagent.protocols.issue_credential.v1_0.routes.V10CredentialProposalRequestSchemaBase*

Request schema for sending credential proposal on mandatory proposal preview.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10CredentialProposalRequestC**

Bases: *aries_cloudagent.protocols.issue_credential.v1_0.routes.V10CredentialProposalRequestSchemaBase*

Request schema for sending credential proposal on optional proposal preview.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10CredentialProposalRequestS**

Bases: aries_cloudagent.utils.tracing.AdminAPIMessageTracingSchema

Base class for request schema for sending credential proposal admin message.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10CredentialStoreRequestSche**

Bases: marshmallow.schema.Schema

Request schema for sending a credential store admin message.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.issue_credential.v1_0.routes.**V10PublishRevocationsResultSc**

Bases: marshmallow.schema.Schema

Result schema for revocation publication API call.

opts = <marshmallow.schema.SchemaOpts object>

aries_cloudagent.protocols.issue_credential.v1_0.routes.**attribute_mime_types_get**(*request: <sphinx.ext.a ob- ject at 0x7fcaf4bce94*

Request handler for getting credential attribute MIME types.

Parameters **request** – aiohttp request object

Returns The MIME types response

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_issue**(*request: <sphinx.ext. ob- ject at 0x7fcaf4bce*

Request handler for sending credential.

Parameters **request** – aiohttp request object

Returns The credential exchange record

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_list**(*request: <sphinx.ext.a ob- ject at 0x7fcaf4bce94*

Request handler for searching connection records.

Parameters **request** – aiohttp request object

Returns The connection list response

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_problem_report**

Request handler for sending problem report.

Parameters **request** – aiohttp request object

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_publish_revocat**

Request handler for publishing pending revocations to the ledger.

Parameters **request** – aiohttp request object

Returns Credential revocation ids published as revoked by revocation registry id.

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_remove**(*request: <sphinx.ex ob- ject at 0x7fcaf4bc*

Request handler for removing a credential exchange record.

Parameters **request** – aiohttp request object

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_retrieve**(*request: <sphinx ob- ject at 0x7fcaf4*

Request handler for fetching single connection record.

Parameters **request** – aiohttp request object

Returns The credential exchange record

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_revoke**(*request: <sphinx.ex ob- ject at 0x7fcaf4bc*

Request handler for storing a credential request.

Parameters **request** – aiohttp request object

Returns The credential request details.

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_send**(*request: <sphinx.ext.a ob- ject at 0x7fcaf4bce94*

Request handler for sending credential from issuer to holder from attr values.

If both issuer and holder are configured for automatic responses, the operation ultimately results in credential issue; otherwise, the result waits on the first response not automated; the credential exchange record retains state regardless.

Parameters **request** – aiohttp request object

Returns The credential exchange record

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_send_bound_offe**

Request handler for sending bound credential offer.

A holder initiates this sequence with a credential proposal; this message responds with an offer bound to the proposal.

Parameters **request** – aiohttp request object

Returns The credential exchange record

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_send_free_offer**

Request handler for sending free credential offer.

An issuer initiates a such a credential offer, free from any holder-initiated corresponding credential proposal with preview.

Parameters **request** – aiohttp request object

Returns The credential exchange record

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_send_proposal**(*r*
<
c
j
c
0

Request handler for sending credential proposal.

Parameters **request** – aiohttp request object

Returns The credential exchange record

aries_cloudagent.protocols.issue_credential.v1_0.routes.**credential_exchange_send_request**(*re*
<s
ob
je
at
0x

Request handler for sending credential request.

Parameters **request** – aiohttp request object

Returns The credential exchange record

`aries_cloudagent.protocols.issue_credential.v1_0.routes.`**`credential_exchange_store`**(*request: <sphinx.ext.a object at 0x7fcaf4bce9*

Request handler for storing credential.

Parameters **`request`** – aiohttp request object

Returns The credential exchange record

`aries_cloudagent.protocols.issue_credential.v1_0.routes.`**`register`**(*app: <sphinx.ext.autodoc.importer._MockO object at 0x7fcaf4bce940>*)

Register routes.

Submodules

aries_cloudagent.protocols.issue_credential.message_types module

aries_cloudagent.protocols.issue_credential.routes module

3.1.8 aries_cloudagent.protocols.present_proof package

Subpackages

aries_cloudagent.protocols.present_proof.v1_0 package

Subpackages

aries_cloudagent.protocols.present_proof.v1_0.handlers package

Submodules

aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_ack_handler module

Presentation ack message handler.

`class` `aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_ack_handler.`**`Prese`**

Bases: `aries_cloudagent.messaging.base_handler.BaseHandler`

Message handler class for presentation acks.

`handle`(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Message handler logic for presentation acks.

Parameters

- **`context`** – request context
- **`responder`** – responder callback

aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_handler module

Presentation message handler.

class aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_handler.**Presentat**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for presentations.

handle(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Message handler logic for presentations.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_proposal_handler module

Presentation proposal message handler.

class aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_proposal_handler

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for presentation proposals.

handle(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Message handler logic for presentation proposals.

Parameters

- **context** – proposal context
- **responder** – responder callback

aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_request_handler module

Presentation request message handler.

class aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_request_handler.

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for Aries#0037 v1.0 presentation requests.

handle(*context: aries_cloudagent.messaging.request_context.RequestContext, responder: aries_cloudagent.messaging.responder.BaseResponder*)

Message handler logic for Aries#0037 v1.0 presentation requests.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.present_proof.v1_0.messages package

Subpackages

aries_cloudagent.protocols.present_proof.v1_0.messages.inner package

Submodules

aries_cloudagent.protocols.present_proof.v1_0.messages.inner.presentation_preview module

A presentation preview inner object.

class aries_cloudagent.protocols.present_proof.v1_0.messages.inner.presentation_preview.**Pre**

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing an attibute specification within a presentation preview.

class Meta

Bases: object

Attr spec metadata.

schema_class = 'PresAttrSpecSchema'

class Posture

Bases: enum.Enum

Attribute posture: self-attested, revealed claim or unrevealed claim.

REVEALED_CLAIM = 1

SELF_ATTESTED = 0

UNREVEALED_CLAIM = 2

b64_decoded_value() → str

Value, base64-decoded if applicable.

static list_plain(*plain: dict*, *cred_def_id: str*, *referent: str = None*)
Return a list of *PresAttrSpec* on input cred def id.

Parameters

- **plain** – dict mapping names to values
- **cred_def_id** – credential definition identifier to specify
- **referent** – single referent to use, omit for none

Returns List of PresAttrSpec on input cred def id with no MIME types

posture
self-attested, revealed claim, or unrevealed claim.

Type Attribute posture

satisfies(*pred_spec: aries_cloudagent.protocols.present_proof.v1_0.messages.inner.presentation_preview.PresPredSpec*)
Whether current specified attribute satisfied input specified predicate.

class aries_cloudagent.protocols.present_proof.v1_0.messages.inner.presentation_preview.**Pre**

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Attribute specifiation schema.

class Meta
Bases: object

Attribute specifiation schema metadata.

model_class
alias of *PresAttrSpec*

cred_def_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<

mime_type = <fields.String(default=<marshmallow.missing>, attribute=None, validate=Non

name = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, re

referent = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None

value = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, r

class aries_cloudagent.protocols.present_proof.v1_0.messages.inner.presentation_preview.**Pre**

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing a predicate specification within a presentation preview.

class Meta
Bases: object

Pred spec metadata.

schema_class = 'PresPredSpecSchema'

class aries_cloudagent.protocols.present_proof.v1_0.messages.inner.presentation_preview.**Pre**

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Predicate specifiation schema.

class Meta

Bases: object

Predicate specifiation schema metadata.

model_class

alias of *PresPredSpec*

cred_def_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<

name = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, re

predicate = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<ar

threshold = <fields.Integer(default=<marshmallow.missing>, attribute=None, validate=No

class aries_cloudagent.protocols.present_proof.v1_0.messages.inner.presentation_preview.**Pre**

Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Class representing presentation preview.

class Meta

Bases: object

Presentation preview metadata.

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/present-proof/1.0/presentation-

schema_class = 'PresentationPreviewSchema'

has_attr_spec(*cred_def_id: str*, *name: str*, *value: str*) → bool

Return whether preview contains given attribute specification.

Parameters

- **cred_def_id** – credential definition identifier

- **name** – attribute name
- **value** – attribute value

Returns Whether preview contains matching attribute specification.

indy_proof_request(*name: str = None, version: str = None, nonce: str = None, ledger: aries_cloudagent.ledger.indy.IndyLedger = None, non_revoc_intervals: Mapping[str, aries_cloudagent.revocation.models.indy.NonRevocationInterval] = None*) → dict

Return indy proof request corresponding to presentation preview.

Typically the verifier turns the proof preview into a proof request.

Parameters

- **name** – for proof request
- **version** – version for proof request
- **nonce** – nonce for proof request
- **ledger** – ledger with credential definitions, to check for revocation support
- **non_revoc_intervals** – non-revocation interval to use per cred def id where applicable (default from and to the current time if applicable)

Returns Indy proof request dict.

class aries_cloudagent.protocols.present_proof.v1_0.messages.inner.presentation_preview.**Pre**

Bases: *aries_cloudagent.messaging.models.base.BaseModelSchema*

Presentation preview schema.

class Meta

Bases: object

Presentation preview schema metadata.

model_class

alias of *PresentationPreview*

attributes = <fields.Nested(default=<marshmallow.missing>, attribute=None, validate=No

predicates = <fields.Nested(default=<marshmallow.missing>, attribute=None, validate=No

Submodules

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation module

A (proof) presentation content message.

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation.**Presentation**(*_id: str = None, *, comment: str = None, presentations_attach: Sequence = None, **kw*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a (proof) presentation.

class Meta

Bases: object

Presentation metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v1_0.handlers.presentatio

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/present-proof/1.0/presentation'

schema_class = 'PresentationSchema'

indy_proof(*index: int = 0*)

Retrieve and decode indy proof from attachment.

Parameters **index** – ordinal in attachment list to decode and return (typically, list has length 1)

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation.**PresentationSche**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessageSchema*

(Proof) presentation schema.

class Meta

Bases: object

Presentation schema metadata.

model_class

alias of *Presentation*

comment = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None,

presentations_attach = <fields.Nested(default=<marshmallow.missing>, attribute=None, va

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_ack module

Represents an explicit RFC 15 ack message, adopted into present-proof protocol.

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_ack.**PresentationA**

Bases: *aries_cloudagent.messaging.ack.message.Ack*

Base class representing an explicit ack message for present-proof protocol.

class Meta

Bases: object

PresentationAck metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v1_0.handlers.presentatio

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/present-proof/1.0/ack'

schema_class = 'PresentationAckSchema'

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_ack.**PresentationA**

Bases: *aries_cloudagent.messaging.ack.message.AckSchema*

Schema for PresentationAck class.

class Meta

Bases: object

PresentationAck schema metadata.

model_class

alias of *PresentationAck*

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_proposal module

A presentation proposal content message.

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_proposal.**Presenta**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a presentation proposal.

class Meta

Bases: object

PresentationProposal metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v1_0.handlers.presentatio

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/present-proof/1.0/propose-prese

schema_class = 'PresentationProposalSchema'

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_proposal.**Presenta**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessageSchema*

Presentation proposal schema.

class Meta

Bases: object

Presentation proposal schema metadata.

model_class

alias of *PresentationProposal*

comment = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None,

presentation_proposal = <fields.Nested(default=<marshmallow.missing>, attribute=None,

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_request module

A presentation request content message.

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_request.**Presentat**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a presentation request.

class Meta

Bases: object

PresentationRequest metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v1_0.handlers.presentatio

message_type = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/present-proof/1.0/request-prese

schema_class = 'PresentationRequestSchema'

indy_proof_request (*index: int = 0*)

Retrieve and decode indy proof request from attachment.

Parameters **index** – ordinal in attachment list to decode and return (typically, list has length 1)

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_request.**Presentat**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessageSchema*

Presentation request schema.

class Meta

Bases: object

Presentation request schema metadata.

model_class

alias of *PresentationRequest*

comment = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None,

request_presentations_attach = <fields.Nested(default=<marshmallow.missing>, attribute=

aries_cloudagent.protocols.present_proof.v1_0.models package

Submodules

aries_cloudagent.protocols.present_proof.v1_0.models.presentation_exchange module

Aries#0037 v1.0 presentation exchange information with non-secrets storage.

class aries_cloudagent.protocols.present_proof.v1_0.models.presentation_exchange.**V10Present**

Bases: *aries_cloudagent.messaging.models.base_record.BaseExchangeRecord*

Represents an Aries#0037 v1.0 presentation exchange.

INITIATOR_EXTERNAL = 'external'

INITIATOR_SELF = 'self'

class Meta

Bases: object

V10PresentationExchange metadata.

schema_class = 'V10PresentationExchangeSchema'

RECORD_ID_NAME = 'presentation_exchange_id'

RECORD_TYPE = 'presentation_exchange_v10'

ROLE_PROVER = 'prover'

ROLE_VERIFIER = 'verifier'

STATE_PRESENTATION_ACKED = 'presentation_acked'

STATE_PRESENTATION_RECEIVED = 'presentation_received'

STATE_PRESENTATION_SENT = 'presentation_sent'

STATE_PROPOSAL_RECEIVED = 'proposal_received'

STATE_PROPOSAL_SENT = 'proposal_sent'

STATE_REQUEST_RECEIVED = 'request_received'

STATE_REQUEST_SENT = 'request_sent'

STATE_VERIFIED = 'verified'

TAG_NAMES = {'thread_id'}

WEBHOOK_TOPIC = 'present_proof'

presentation_exchange_id

Accessor for the ID associated with this exchange.

record_value

Accessor for JSON record value generated for this presentation exchange.

class aries_cloudagent.protocols.present_proof.v1_0.models.presentation_exchange.**V10Present**

Bases: *aries_cloudagent.messaging.models.base_record.BaseExchangeSchema*

Schema for de/serialization of v1.0 presentation exchange records.

class Meta

Bases: object

V10PresentationExchangeSchema metadata.

model_class

alias of *V10PresentationExchange*

auto_present = <fields.Boolean(default=<marshmallow.missing>, attribute=None, validate=

connection_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=

error_msg = <fields.String(default=<marshmallow.missing>, attribute=None, validate=Non

initiator = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<On

```
presentation = <fields.Dict(default=<marshmallow.missing>, attribute=None, validate=No
presentation_exchange_id = <fields.String(default=<marshmallow.missing>, attribute=Non
presentation_proposal_dict = <fields.Dict(default=<marshmallow.missing>, attribute=Non
presentation_request = <fields.Dict(default=<marshmallow.missing>, attribute=None, val
role = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<OneOf(c
state = <fields.String(default=<marshmallow.missing>, attribute=None, validate=None, r
thread_id = <fields.String(default=<marshmallow.missing>, attribute=None, validate=Non
verified = <fields.String(default=<marshmallow.missing>, attribute=None, validate=<One
```

aries_cloudagent.protocols.present_proof.v1_0.util package

Submodules

aries_cloudagent.protocols.present_proof.v1_0.util.indy module

Utilities for dealing with indy conventions.

`aries_cloudagent.protocols.present_proof.v1_0.util.indy.`**`indy_proof_req_preview2indy_request`**

Build indy requested-credentials structure.

Given input proof request and presentation preview, use credentials in holder's wallet to build indy requested credentials structure for input to proof creation.

Parameters

- **`indy_proof_request`** – indy proof request
- **`pres_preview`** – preview from presentation proposal, if applicable
- **`holder`** – holder injected into current context

Submodules

aries_cloudagent.protocols.present_proof.v1_0.manager module

Classes to manage presentations.

class `aries_cloudagent.protocols.present_proof.v1_0.manager.`**`PresentationManager`**(*context: aries_cloudage*

Bases: `object`

Class for managing presentations.

context

Accessor for the current request context.

Returns The injection context for this presentation manager

create_bound_request(*presentation_exchange_record: aries_cloudagent.protocols.present_proof.v1_0.models.presentati name: str = None, version: str = None, nonce: str = None, comment: str = None*)

Create a presentation request bound to a proposal.

Parameters

- **presentation_exchange_record** – Presentation exchange record for which to create presentation request
- **name** – name to use in presentation request (None for default)
- **version** – version to use in presentation request (None for default)
- **nonce** – nonce to use in presentation request (None to generate)
- **comment** – Optional human-readable comment pertaining to request creation

Returns A tuple (updated presentation exchange record, presentation request message)

create_exchange_for_proposal(*connection_id: str, presentation_proposal_message: aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_proposal.Prese auto_present: bool = None*)

Create a presentation exchange record for input presentation proposal.

Parameters

- **connection_id** – connection identifier
- **presentation_proposal_message** – presentation proposal to serialize to exchange record
- **auto_present** – whether to present proof upon receiving proof request (default to configuration setting)

Returns Presentation exchange record, created

create_exchange_for_request(*connection_id: str, presentation_request_message: aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_request.Present*

Create a presentation exchange record for input presentation request.

Parameters

- **connection_id** – connection identifier
- **presentation_request_message** – presentation request to use in creating exchange record, extracting indy proof request and thread id

Returns Presentation exchange record, updated

create_presentation(*presentation_exchange_record: aries_cloudagent.protocols.present_proof.v1_0.models.presentatio requested_credentials: dict, comment: str = None*)

Create a presentation.

Parameters

- **presentation_exchange_record** – Record to update
- **requested_credentials** – Indy formatted requested_credentials
- **comment** – optional human-readable comment

Example *requested_credentials* format:

```
{
    "self_attested_attributes": {
        "j233ffbc-bd35-49b1-934f-51e083106f6d": "value"
    },
    "requested_attributes": {
        "6253ffbb-bd35-49b3-934f-46e083106f6c": {
            "cred_id": "5bfa40b7-062b-4ae0-a251-a86c87922c0e",
            "revealed": true
        }
    },
    "requested_predicates": {
        "bfc8a97d-60d3-4f21-b998-85eeabe5c8c0": {
            "cred_id": "5bfa40b7-062b-4ae0-a251-a86c87922c0e"
        }
    }
}
```

Returns A tuple (updated presentation exchange record, presentation message)

receive_presentation()

Receive a presentation, from message in context on manager creation.

Returns presentation exchange record, retrieved and updated

receive_presentation_ack()

Receive a presentation ack, from message in context on manager creation.

Returns presentation exchange record, retrieved and updated

receive_proposal()

Receive a presentation proposal from message in context on manager creation.

Returns Presentation exchange record, created

receive_request(*presentation_exchange_record: aries_cloudagent.protocols.present_proof.v1_0.models.presentation_excl*

Receive a presentation request.

Parameters **presentation_exchange_record** – presentation exchange record with request to receive

Returns The presentation_exchange_record, updated

send_presentation_ack(*presentation_exchange_record: aries_cloudagent.protocols.present_proof.v1_0.models.presenta*

Send acknowledgement of presentation receipt.

Parameters **presentation_exchange_record** – presentation exchange record with thread id

verify_presentation(*presentation_exchange_record: aries_cloudagent.protocols.present_proof.v1_0.models.presentatio*

Verify a presentation.

Parameters **presentation_exchange_record** – presentation exchange record with presentation request and presentation to verify

Returns presentation record, updated

exception aries_cloudagent.protocols.present_proof.v1_0.manager.**PresentationManagerError**(*a
er
ro
st
=
N
**

Bases: *aries_cloudagent.core.error.BaseError*

Presentation error.

aries_cloudagent.protocols.present_proof.v1_0.message_types module

Message and inner object type identifiers for Connections.

aries_cloudagent.protocols.present_proof.v1_0.routes module

Admin routes for presentations.

class aries_cloudagent.protocols.present_proof.v1_0.routes.**IndyProofReqAttrSpecSchema**(*, *only=N ex-clude= many= con-text=N load_o dump_ par-tial=F un-known*

Bases: marshmallow.schema.Schema

Schema for attribute specification in indy proof request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**IndyProofReqNonRevoked**(*, *only=None, ex-clude=(), many=False con-text=None, load_only=(dump_only= par-tial=False, un-known=Non*

Bases: marshmallow.schema.Schema

Non-revocation times specification in indy proof request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**IndyProofReqPredSpecSchema**(*, *only=N ex- clude= many= con- text=N load_o dump_ par- tial=F un- known*

Bases: marshmallow.schema.Schema

Schema for predicate specification in indy proof request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**IndyProofReqSpecRestrictionsSche**

Bases: marshmallow.schema.Schema

Schema for restrictions in attr or pred specifier indy proof request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**IndyProofRequestSchema**(*, *only=None, ex- clude=(), many=False con- text=None, load_only=(dump_only= par- tial=False, un- known=Non*

Bases: marshmallow.schema.Schema

Schema for indy proof request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**IndyRequestedCredsRequestedAttrS**

Bases: marshmallow.schema.Schema

Schema for requested attributes within indy requested credentials structure.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**IndyRequestedCredsRequestedPredS**

Bases: marshmallow.schema.Schema

Schema for requested predicates within indy requested credentials structure.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**V10PresentationExchangeListSchem**

Bases: marshmallow.schema.Schema

Result schema for an Aries#0037 v1.0 presentation exchange query.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**V10PresentationProposalRequestSc**

Bases: aries_cloudagent.utils.tracing.AdminAPIMessageTracingSchema

Request schema for sending a presentation proposal admin message.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**V10PresentationRequestRequestSch**

Bases: aries_cloudagent.utils.tracing.AdminAPIMessageTracingSchema

Request schema for sending a proof request.

opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.protocols.present_proof.v1_0.routes.**V10PresentationRequestSchema**(*, *only ex- clud mar con text load dum par tial un- kno*

Bases: aries_cloudagent.utils.tracing.AdminAPIMessageTracingSchema

Request schema for sending a presentation.

opts = <marshmallow.schema.SchemaOpts object>

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_create_request**(*r*
<
c
j
c
(

Request handler for creating a free presentation request.

The presentation request will not be bound to any proposal or existing connection.

Parameters **request** – aiohttp request object

Returns The presentation exchange details

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_credentials_list**

Request handler for searching applicable credential records.

Parameters **request** – aiohttp request object

Returns The credential list response

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_list**(*request: <sphinx.ext.aut ob- ject at 0x7fcaf3e659e8*

Request handler for searching presentation exchange records.

Parameters **request** – aiohttp request object

Returns The presentation exchange list response

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_remove**(*request: <sphinx.ext. ob- ject at 0x7fcaf3e65*

Request handler for removing a presentation exchange record.

Parameters **request** – aiohttp request object

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_retrieve**(*request: <sphinx.e ob- ject at 0x7fcaf3e*

Request handler for fetching a single presentation exchange record.

Parameters **request** – aiohttp request object

Returns The presentation exchange record response

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_send_bound_reque**

Request handler for sending a presentation request free from any proposal.

Parameters **request** – aiohttp request object

Returns The presentation exchange details

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_send_free_reques**

Request handler for sending a presentation request free from any proposal.

Parameters **request** – aiohttp request object

Returns The presentation exchange details

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_send_presentatio**

Request handler for sending a presentation.

Parameters **request** – aiohttp request object

Returns The presentation exchange details

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_send_proposal**(*re*
<s
ob
je
at
0x

Request handler for sending a presentation proposal.

Parameters **request** – aiohttp request object

Returns The presentation exchange details

aries_cloudagent.protocols.present_proof.v1_0.routes.**presentation_exchange_verify_presentat**

Request handler for verifying a presentation request.

Parameters **request** – aiohttp request object

Returns The presentation exchange details

`aries_cloudagent.protocols.present_proof.v1_0.routes.register`(*app: <sphinx.ext.autodoc.importer._MockObjec object at 0x7fcaf3e659e8>*)

Register routes.

Submodules

aries_cloudagent.protocols.present_proof.message_types module

aries_cloudagent.protocols.present_proof.routes module

3.1.9 aries_cloudagent.protocols.presentations package

Subpackages

aries_cloudagent.protocols.presentations.handlers package

Submodules

aries_cloudagent.protocols.presentations.handlers.credential_presentation_handler module

aries_cloudagent.protocols.presentations.handlers.presentation_request_handler module

aries_cloudagent.protocols.presentations.messages package

Submodules

aries_cloudagent.protocols.presentations.messages.credential_presentation module

aries_cloudagent.protocols.presentations.messages.presentation_request module

aries_cloudagent.protocols.presentations.models package

Submodules

aries_cloudagent.protocols.presentations.models.presentation_exchange module

Submodules

aries_cloudagent.protocols.presentations.manager module

aries_cloudagent.protocols.presentations.message_types module

aries_cloudagent.protocols.presentations.routes module

3.1.10 aries_cloudagent.protocols.problem_report package

Submodules

aries_cloudagent.protocols.problem_report.handler module

aries_cloudagent.protocols.problem_report.message module

aries_cloudagent.protocols.problem_report.message_types module

3.1.11 aries_cloudagent.protocols.routing package

Subpackages

aries_cloudagent.protocols.routing.handlers package

Submodules

aries_cloudagent.protocols.routing.handlers.forward_handler module

aries_cloudagent.protocols.routing.handlers.route_query_request_handler module

aries_cloudagent.protocols.routing.handlers.route_query_response_handler module

aries_cloudagent.protocols.routing.handlers.route_update_request_handler module

aries_cloudagent.protocols.routing.handlers.route_update_response_handler module

aries_cloudagent.protocols.routing.messages package

Submodules

aries_cloudagent.protocols.routing.messages.forward module

aries_cloudagent.protocols.routing.messages.route_query_request module

aries_cloudagent.protocols.routing.messages.route_query_response module

aries_cloudagent.protocols.routing.messages.route_update_request module

aries_cloudagent.protocols.routing.messages.route_update_response module

aries_cloudagent.protocols.routing.models package

Submodules

aries_cloudagent.protocols.routing.models.paginate module

aries_cloudagent.protocols.routing.models.paginated module

aries_cloudagent.protocols.routing.models.route_query_result module

aries_cloudagent.protocols.routing.models.route_record module

aries_cloudagent.protocols.routing.models.route_update module

aries_cloudagent.protocols.routing.models.route_updated module

Submodules

aries_cloudagent.protocols.routing.manager module

aries_cloudagent.protocols.routing.message_types module

3.1.12 aries_cloudagent.protocols.trustping package

Subpackages

aries_cloudagent.protocols.trustping.handlers package

Submodules

aries_cloudagent.protocols.trustping.handlers.ping_handler module

aries_cloudagent.protocols.trustping.handlers.ping_response_handler module

aries_cloudagent.protocols.trustping.messages package

Submodules

aries_cloudagent.protocols.trustping.messages.ping module

aries_cloudagent.protocols.trustping.messages.ping_response module

Submodules

aries_cloudagent.protocols.trustping.message_types module

aries_cloudagent.protocols.trustping.routes module

CHAPTER 4

Indices and tables

- genindex

Python Module Index

a

Index

A

B

C

D

E

F

G

H

J

M

N

O

P

Q

R

T

U

V

W